A Pumping Lemma for Ordered Restarting Automata

Kent Kwee, Friedrich Otto

A Pumping Lemma for Ordered Restarting Automata

Kent Kwee and Friedrich Otto

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

Abstract. While stateless ordered restarting automata accept exactly the regular languages, it is known that ordered restarting automata with states accept some languages that are not even growing context-sensitive. In fact, the class of languages accepted by these automata is an abstract family of languages that is incomparable to the (deterministic) linear languages, the (deterministic) context-free languages, and the growing context-sensitive languages with respect to inclusion, and the emptiness problem is decidable for these automata. These results were derived using a Cut-and-Paste Lemma for ordered restarting automata that is based on Higman's theorem. Here we extend the arguments used in that proof to actually derive a real Pumping Lemma for these automata. Based on this Pumping Lemma, we then prove that the finiteness problem is also decidable for these automata, and that the only unary languages these automata accept are the regular ones.

Keywords: restarting automaton, ordered rewriting, pumping lemma, finiteness problem

1 Introduction

The *ordered restarting automaton* (ORWW-automaton for short) was introduced in [9], where its deterministic variant was extended into a device for recognizing picture languages. An ORWW-automaton (for words) has a finite-state control, a tape with end markers that initially contains the input, and a window of size three. Based on its state and the content of its window, the automaton can either perform a *move-right step*, a *rewrite/restart step*, or an *accept step*. While the deterministic variant of the ORWW-automaton characterizes the regular languages, it has been observed that the nondeterministic variant is more expressive. In fact, the nondeterministic ORWW-automaton and the languages it accepts have been studied in some detail in [6], where it is shown that the class of languages accepted by ORWW-automata forms an abstract family of languages, that is, it is closed under union, intersection (with regular sets), product, Kleene star, inverse morphisms, and non-erasing morphisms (see, e.g., [3]). However, it is neither closed under complementation nor under reversal. Further, it is incomparable to the (deterministic) linear, the (deterministic) context-free, and

the growing context-sensitive languages with respect to inclusion, as it contains a language that is not even growing context-sensitive, while on the other hand, it does not even include the deterministic linear language $\{ a^m b^m \mid m \geq 1 \}$. In addition, it was shown that the emptiness problem is decidable for ORWW-automata. Several of these results were derived from a Cut-and-Paste Lemma for ORWW-automata that is based on Higman's Theorem [2].

Here we continue the study of nondeterministic ORWW-automata, where we are particularly interested in the expressive capability of ORWW-automata and their algorithmic properties. The Cut-and-Paste Lemma of [6] states that, for each ORWW-automaton M, a non-empty factor can be cut from the *suffix* of each sufficiently long word accepted by M such that the resulting shorter word is accepted by M, too. Thus, in comparison to the Pumping Lemma for regular languages (see, e.g., [3]), the Cut-and-Paste Lemma just covers the case of pumping with exponent zero. Here we also present a real Pumping Lemma for ORWW-automata that takes care of the case of pumping with positive exponents. However, in contrast to the Cut-and-Paste Lemma, which applies to the suffix of a sufficiently long word, the Pumping Lemma applies to the *prefix* of a sufficiently long word. Then, based on both these lemmas, we show that finiteness is decidable for ORWW-automata, and we show that each unary language that is accepted by an ORWW-automaton is necessarily regular.

This paper is structured as follows. In Section 2, we introduce the ORWW-automaton and restate the known results on the class of languages it accepts. Then, in Section 3, we present the announced Pumping Lemma, which is derived from Higman's theorem similar to the Cut-and-Paste Lemma. In Section 4, we give two applications of this lemma by showing that finiteness is decidable for ORWW-automata and that all unary languages that are accepted by ORWW-automata are necessarily regular. The paper closes with Section 5, which summarizes our results in short and states a number of open problems.

2 Definitions and Known Results

An *ordered restarting automaton* (ORWW-automaton) is a one-tape machine that is described by an 8-tuple $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, \delta, >)$, where Q is a finite set of states containing the initial state q_0, Σ is a finite input alphabet, Γ is a finite tape alphabet such that $\Sigma \subseteq \Gamma$, the symbols $\rhd, \lhd \notin \Gamma$ serve as markers for the left and right border of the work space, respectively,

$$\delta : (Q \times ((\Gamma \cup \{\rhd\}) \cdot \Gamma \cdot (\Gamma \cup \{\lhd\}) \cup \{\rhd\lhd\})) \to 2^{(Q \times \mathsf{MVR}) \cup \Gamma} \cup \{\mathsf{Accept}\}$$

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

(1) A *move-right step* has the form $(q', \mathsf{MVR}) \in \delta(q, a_1 a_2 a_3)$, where $q, q' \in Q$, $a_1 \in \Gamma \cup \{\rhd\}$, and $a_2, a_3 \in \Gamma$. It causes M to shift the window one position to the right and to change from state q to state q'. Observe that no move-right step is possible, if the window contains the symbol $\lhd$.

(2) A *rewrite/restart step* has the form $b \in \delta(q, a_1a_2a_3)$, where $q \in Q$, $a_1 \in \Gamma \cup \{\rhd\}$, $a_2, b \in \Gamma$, and $a_3 \in \Gamma \cup \{\lhd\}$ 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, that is, the window is moved back to the left end of the tape, and M reenters its initial state q_0.

(3) An *accept step* has the form $\delta(q, a_1a_2a_3) = \mathsf{Accept}$, where $q \in Q$, $a_1 \in \Gamma \cup \{\rhd\}$, $a_2 \in \Gamma$, and $a_3 \in \Gamma \cup \{\lhd\}$. It causes M to halt and accept. In addition, we allow an accept step of the form $\delta(q_0, \rhd\lhd) = \mathsf{Accept}$.

If $\delta(q, u) = \emptyset$ for some state q and a word u, then M necessarily halts, when it is in state q with u in its window, and we say that M *rejects* in this situation. Further, the letters in $\Gamma \setminus \Sigma$ are called *auxiliary symbols*.

If $|\delta(q, u)| \leq 1$ for all q and u, then M is a *deterministic ORWW-automaton* (det-ORWW-automaton), and if $Q = \{q_0\}$, that is, if the initial state is the only state of M, then we call M a *stateless* ORWW-automaton (stl-ORWW-automaton) or a *stateless deterministic ORWW-automaton* (stl-det-ORWW-automaton), as in this case the state is actually not needed.

A *configuration* of an ORWW-automaton M is a word $\alpha q \beta$, where $q \in Q$ is the current state, $|\beta| \geq 3$, and either $\alpha = \lambda$ (the empty word) and $\beta \in \{\rhd\} \cdot \Gamma^+ \cdot \{\lhd\}$ 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 $q_0 \rhd \lhd$. By $\vdash_M$ we denote binary relation that M induces on the set of configurations, and $\vdash_M^*$ is the reflexive transitive closure of this relation. A *restarting configuration* has the form $q_0 \rhd w \lhd$; if $w \in \Sigma^*$, then $q_0 \rhd w \lhd$ is also called an *initial configuration*. Further, we use Accept to denote the *accepting configurations*, which are those configurations that M reaches by an accept step.

Any computation of an 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 is realized by M on the set of restarting configurations.

An input $w \in \Sigma^*$ is accepted by M, if there is a computation of M which starts with the initial configuration $q_0 \rhd w \lhd$ and which ends with an accept step. The language consisting of all input words that are accepted by M is denoted by $L(M)$. Further, by $\mathcal{L}(\mathsf{ORWW})$ we denote the class of all languages that are accepted by ORWW-automata.

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 $>$, each computation of M on an input of length n consists of at most $(|\Gamma| - 1) \cdot n$ cycles. Thus, M can be simulated by a nondeterministic single-tape Turing machine in time $O(n^2)$.

The following technical result has already been used in [6] without stating or proving it explicitly. As below we will use it again, we present it in some detail.

Lemma 1. *For each ORWW-automaton M, there exists an ORWW-automaton M' that accepts the same language as M, but that performs accept steps only at the left sentinel.*

Proof. Let $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, \delta, >)$ be an ORWW-automaton. To obtain the automaton $M' = (Q', \Sigma, \Gamma', \rhd, \lhd, q_0', \delta', >')$, we take $Q' = Q$, $q_0' = q_0$, and $\Gamma' = \Gamma \cup \{*\}$, where $*$ is a new symbol. Further, we extend $>$ to $>'$ by taking $a >' *$ for all $a \in \Gamma$. Finally, we define the transition relation δ' of M' as follows, where $a \in \Gamma \cup \{\rhd\}$, $b \in \Gamma$, $c \in \Gamma \cup \{\lhd\}$, and $q \in Q$:

$$\begin{array}{lll}
\delta'(q_0, \rhd\lhd) & = \delta(q_0, \rhd\lhd), & \\
\delta'(q, abc) & = \delta(q, abc), & \text{if } \delta(q, abc) \neq \mathsf{Accept}, \\
\delta'(q, abc) & = \{*\}, & \text{if } \delta(q, abc) = \mathsf{Accept}, \\
\delta'(q, ab*) & = \{*\}, & \\
\delta'(q_0, \rhd * d) & = \mathsf{Accept} & \text{for all } d \in \Gamma \cup \{\lhd, *\}.
\end{array}$$

Obviously, M' performs an accept step only at its left sentinel. The automaton M' can simulate M step by step until M accepts, in which case M' writes the letter $*$. In the following cycles, whenever M' detects an occurrence of the symbol $*$, it copies this symbol to its left-hand neighbour. It follows that $L(M) \subseteq L(M')$. On the other hand, if M' accepts on input w, then it can do so only because it has been able to simulate an accepting computation of M on input w, as the first $*$-symbol can only be produced by M' on reaching a configuration in which M would accept. Thus, $L(M) = L(M')$ holds. □

While nondeterministic ORWW-automata are quite expressive as we will see below, the deterministic variants are fairly weak.

Theorem 2. [5, 11]

(a) *For each det-ORWW-automaton $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, \delta, >)$, there exists a stateless det-ORWW-automaton $M' = (\Sigma, \Gamma', \rhd, \lhd, \delta', >')$ such that $L(M') = L(M)$ and $|\Gamma'| = |Q| \cdot |\Gamma|^2 + 2 \cdot |\Gamma|$.*

(b) *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 $|\Gamma| = |Q| + |\Sigma|$.*

(c) *For each stl-det-ORWW-automaton M with an alphabet of size n, there exists an NFA A of size $2^{O(n)}$ such that $L(A) = L(M)$ holds.*

(d) *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 NFA for accepting B_n has at least 2^n states.*

Lemma 3. (Cut-and-Paste Lemma) [6]
For each ORWW-automaton M, there exists a constant $N_c(M) > 0$ such that each word $w \in L(M)$, $|w| \geq N_c(M)$, has a factorization $w = xyz$ satisfying all of the following conditions:

$$\text{(a) } |yz| \leq N_c(M), \text{ (b) } |y| > 0, \text{ and (c) } xz \in L(M).$$

In addition, the constant N_c can be determined from M effectively.

Theorem 4. [6] $\mathcal{L}(\mathsf{ORWW})$ *is closed under union, intersection, product, Kleene star, inverse morphisms, and non-erasing morphisms, but it is neither closed under the operation of reversal nor under complementation.*

Using the Cut-and-Paste Lemma it is easily seen that the deterministic linear language $\{a^m b^m \mid m \geq 1\}$ is not accepted by any ORWW-automaton. On the other hand, there exists a language that is accepted by an ORWW-automaton, but that is not even growing context-sensitive. Thus, we have the following incomparability results, where DLIN denotes the *deterministic linear languages*, that is, those languages that are accepted by deterministic one-turn pushdown automata, LIN is the class of *linear languages*, CFL and DCFL are the classes of context-free and deterministic context-free languages, CRL is the class of *Church-Rosser languages* [8], and GCSL is the class of *growing context-sensitive languages* [1].

Corollary 5. *The language class* $\mathcal{L}(\mathsf{ORWW})$ *is incomparable to the language classes* DLIN, LIN, DCFL, CFL, CRL, *and* GCSL *with respect to inclusion.*

Also from the Cut-and-Paste Lemma the following decidability result follows.

Theorem 6. [6] *The emptiness problem for ORWW-automata is decidable.*

The following result was given without proof in [6], pointing out that the construction for the deterministic case (see Theorem 2 (c)) can be extended accordingly. In fact, a simpler construction is presented in [7].

Theorem 7. [6] *Let* $M = (\Sigma, \Gamma, \rhd, \lhd, \delta_M, >)$ *be a stl-ORWW-automaton. Then* $L(M)$ *is a regular language.*

3 A Pumping Lemma for ORWW-Automata

Here we derive our main result, the Pumping Lemma for ORWW-automata.

Definition 8. *Let* $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, \delta, >)$ *be an ORWW-automaton. The transition relation* δ *can be presented by a set of five-tuples of the form* (q, a_1, a_2, a_3, o)*, where* $q \in Q$*,* $a_1 \in \Gamma \cup \{\rhd\}$*,* $a_2 \in \Gamma$*,* $a_3 \in \Gamma \cup \{\lhd\}$*, and* $o \in \Gamma \cup Q \cup \{\mathsf{Accept}\}$*. Here a tuple* (q, a_1, a_2, a_3, q') *with* $q' \in Q$ *represents the move-right transition* $(q', \mathsf{MVR}) \in \delta(q, a_1, a_2, a_3)$*, a tuple* (q, a_1, a_2, a_3, b) *with* $b \in \Gamma$ *represents the rewrite/restart transition* $b \in \delta(q, a_1, a_2, a_3)$*, and a tuple* $(q, a_1, a_2, a_3, \mathsf{Accept})$ *represents the accept transition* $\delta(q, a_1, a_2, a_3) = \mathsf{Accept}$*. We introduce an alphabet* Ω *the letters of which are in 1-to-1 correspondence to these five-tuples.*

Let $w \in L(M)$ *and let* C *be an accepting computation of* M *on input* w*. With each integer* i*,* $1 \leq i \leq |w|$*, we associate a word* $\sigma_i^C \in \Omega^*$ *that corresponds to the sequence of operations that* M *executes within the computation* C

at position i, that is, when the i-th letter is in the middle of the window. Let $\sigma_i^C = t_{j_1} t_{j_2} \dots t_{j_s}$, where $t_{j_r} \in \Omega$ for all $1 \leq r \leq s$. If $t_{j_r} = (q_1, a_1, a_2, a_3, o_1)$ and $t_{j_{r+1}} = (q_2, b_1, b_2, b_3, o_2)$, then $a_1 \geq b_1$, $a_2 \geq b_2$, and $a_3 \geq b_3$. In addition, if $o_1 = q' \in Q$, that is, it represents a move-right operation, then $a_2 = b_2$, and if $o_1 = b \in \Gamma$, that is, it represents a rewrite/restart operation, then $a_2 > b = b_2$. Now the pattern *$\tau_i^C \in \Omega^*$ is the word that is obtained from σ_i^C by condensing consecutive identical letters into a single letter.*

Observe that it is only MVR operations that may be condensed.

Example 9. Consider the following accepting computation C of an ORWW-automaton M:

$$\begin{aligned} q_0 \rhd aaa \lhd &\vdash_M q_0 \rhd a_1 aa \lhd \vdash_M \rhd q_0 a_1 aa \lhd \vdash_M \rhd a_1 q_0 aa \lhd \\ &\vdash_M q_0 \rhd a_1 aa_1 \lhd \vdash_M \rhd q_0 a_1 aa_1 \lhd \vdash_M \rhd a_1 q_0 aa_1 \lhd \vdash_M \mathsf{Accept}. \end{aligned}$$

This computation consists of two cycles and an accepting tail that are described by the following sequences of operations:

$$\begin{aligned} c_1 &= (q_0, \rhd, a, a, a_1), \\ c_2 &= (q_0, \rhd, a_1, a, q_0), (q_0, a_1, a, a, q_0), (q_0, a, a, \lhd, a_1), \\ c_3 &= (q_0, \rhd, a_1, a, q_0), (q_0, a_1, a, a_1, q_0), (q_0, a, a_1, \lhd, \mathsf{Accept}). \end{aligned}$$

For the first position, we thus get the sequence of operations

$$\sigma_1^C = (q_0, \rhd, a, a, a_1)(q_0, \rhd, a_1, a, q_0)(q_0, \rhd, a_1, a, q_0),$$

which yields the pattern $\tau_1^C = (q_0, \rhd, a, a, a_1)(q_0, \rhd, a_1, a, q_0)$, while for the second position we get the sequence of operations

$$\sigma_2^C = (q_0, a_1, a, a, q_0)(q_0, a_1, a, a_1, q_0) = \tau_2^C.$$

For the third position we have $\sigma_3^C = (q_0, a, a, \lhd, a_1)(q_0, a, a_1, \lhd, \mathsf{Accept}) = \tau_3^C$.

For two patterns τ_1^C and τ_2^C, we write $\tau_1^C \sqsubseteq \tau_2^C$ if τ_1^C is a *scattered subword* of τ_2^C, that is, if $\tau_1^C = \omega_1 \omega_2 \dots \omega_m$ for some $\omega_1, \omega_2, \dots, \omega_m \in \Omega$, then there are words $y_0, y_1, \dots, y_m \in \Omega^*$ such that $\tau_2^C = y_0 \omega_1 y_1 \omega_2 y_2 \dots y_{m-1} \omega_m y_m$. The next lemma is the main step towards the proof of the Pumping Lemma.

Lemma 10. *Let $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, \delta, >)$ be an ORWW-automaton that accepts at the left sentinel, let C_{xz} be an accepting computation of M for the input xz, and let C_{uv} be an accepting computation of M for the input uv. If the pattern $\tau_{|u|}^{C_{uv}}$ of the computation C_{uv} at position $|u|$ is a scattered subword of the pattern $\tau_{|x|}^{C_{xz}}$ of the computation C_{xz} at position $|x|$, that is, $\tau_{|u|}^{C_{uv}} \sqsubseteq \tau_{|x|}^{C_{xz}}$, and if these two patterns contain the same rewrite operations, then $xv \in L(M)$.*

Proof. We construct an accepting computation C' for the input xv from the given computations C_{xz} and C_{uv}. The sequences of cycles $(C_1, C_2, \dots, C_\alpha)$ of C_{xz} and $(D_1, D_2, \dots, D_\beta)$ of C_{uv} are considered as working lists that are used for

constructing the cycles of C' that have their rewrite operations in the prefix x or in the suffix v of the input xv, respectively. As $\tau_{|u|}^{C_{uv}} \sqsubseteq \tau_{|x|}^{C_{xz}}$, these patterns can be written as $\tau_{|u|}^{C_{uv}} = t_1 t_2 \dots t_r$ with $t_1, t_2, \dots, t_r \in \Omega$ and $\tau_{|x|}^{C_{xz}} = y_0 t_1 y_1 \dots y_{r-1} t_r y_r$ for some $y_0, y_1, \dots, y_r \in \Omega^*$ (see Fig. 1). As both patterns contain the same rewrite operations, the factors $y_0, y_1, \dots y_r$ only consist of MVR operations.

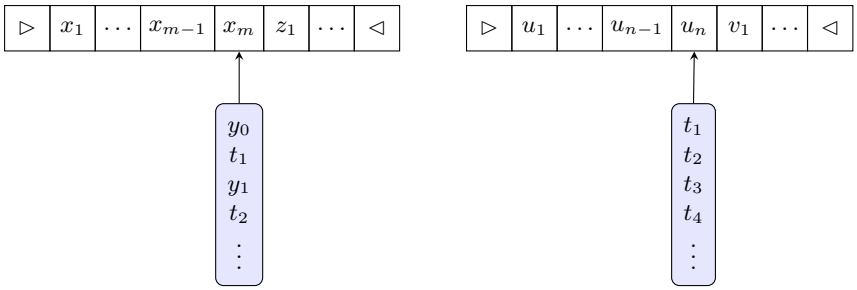

Fig. 1: The inputs xz and uv with the patterns $\tau_{|x|}^{C_{xz}}$ (left) and $\tau_{|u|}^{C_{uv}}$ (right)

For constructing the computation C' on input xv, we start by taking C' to be the empty sequence of cycles. Now we consider the cycles of C_{xz} one after another (see Fig. 2).

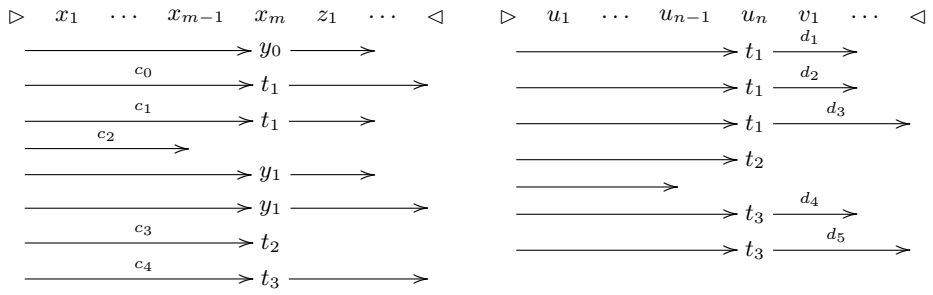

Fig. 2: The cycles of the computations C_{xz} (left) and C_{uv} (right). Each line represents a cycle, where the operation executed at the last position of x (left) or u (right) is displayed. The arrows labelled c_i represent initial parts of cycles executed within the prefix of the tape initially containing x (left), and the arrows labelled d_j represent final parts of cycles executed within the suffix of the tape initially containing v (right).

Let C_i be the cycle currently considered.

- If C_i is a *short cycle*, that is, a cycle that executes a rewrite step within a proper prefix of x, then we just append it to C' (see the cycle c_2 in Fig. 2).
- If C_i contains a rewrite operation at position $|x|$, then this operation corresponds to the letter t_l for some $1 \leq l \leq r$. Again we append this cycle

to C' (see the cycle c_3). As both patterns contain the same rewrite operations, which must occur in the same relative order in both patterns, we see that the rewrite operation t_l can also be executed at this point in the computation C'.

- If C_i is a cycle that executes a rewrite step within the suffix z of xz, then this cycle contains a MVR operation at position $|x|$. If this operation does not correspond to one of the letters $t_1, t_2, \ldots, t_r$ in the pattern $\tau_{|x|}^{C_{xz}}$, we skip this cycle without appending it to C'.
- Finally, let C_i be a cycle that executes a rewrite step within the suffix z of xz, but the MVR operation executed at position $|x|$ corresponds to the letter t_l for some $1 \leq l \leq r$. By c_0 we denote the prefix of the cycle C_i up to position $|x|-1$. Further, let $D_{i_1}, D_{i_2}, \ldots, D_{i_\nu}$ be all those cycles of C_{uv} that contain the MVR operation t_l at position $|u|$, and for all $1 \leq j \leq \nu$, let d_j be the suffix of the cycle D_{i_j} that starts with the operation t_l at position $|u|$. We now combine the prefix c_0 of C_i with the suffix d_j of D_{i_j} for all $1 \leq j \leq \nu$ (see c_0 and d_1, d_2, d_3 in Fig. 2). As the same operation t_l is applied in the cycle C_i at position $|x|$ as in the cycles $D_{i_1}, D_{i_2}, \ldots, D_{i_\nu}$ at position $|u|$, we see that $c_0d_1, c_0d_2, \ldots, c_0d_\nu$ is a sequence of possible cycles of M. We append this sequence of cycles to C'.
- Any further cycle C_{i+s}, $s \geq 1$, that also executes a MVR operation at position $|x|$ which corresponds to the letter t_l of the pattern $\tau_{|x|}^{C_{xz}}$, is skipped (see c_1 in Fig. 2).

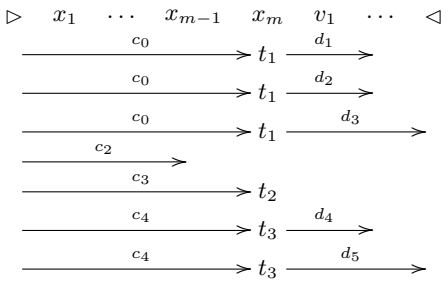

Fig. 3: The computation C' for input xv

Fig. 3 illustrates the result of the construction above. Finally, the computation C' is completed by attaching the accepting tail computation from C_{xz} to it. Recall that M accepts with the left sentinel in its window. It is now easily seen that C' is an accepting computation of M for the input xv. □

Next we consider a special case of the above lemma.

Lemma 11. *Let $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, \delta, >)$ be an ORWW-automaton that accepts at the left sentinel, let $w \in L(M)$, let C be an accepting computation of M for the input w, and let $1 \leq i < j \leq |w|$ be indices such that $\tau_i^C(w) \sqsubseteq \tau_j^C(w)$ and these two patterns contain the same rewrite operations. Then w can be written as $w = xyz$, where $|x| = i$ and $|y| = j - i$, such that $xyyz \in L(M)$. In fact, there exists an accepting computation C' for $xyyz$ satisfying $\tau_i^{C'}(xyyz) = \tau_j^{C'}(xyyz)$.*

Proof. If we choose $x_1 = xy$, $y_1 = z$, $u_1 = x$, and $v_1 = yz$, we can apply Lemma 10 to the factorizations $w = xyz = x_1y_1$ and $w = xyz = u_1v_1$. Thus, we obtain an accepting computation C' of M for the input $x_1v_1 = xyyz$. From the construction of C' in the proof of the above lemma we see that the patterns $\tau_i^{C'}(xyyz)$ and $\tau_j^{C'}(xyyz)$ coincide. □

Finally, we need the following notion that has already been considered in [10] under the name of *det-MVR$_1$-form* for general restarting automata.

Definition 12. *An ORWW-automaton $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, \delta, >)$ is said to have* deterministic MVR operations *if, for all $q \in Q$ and all $a, b, c \in \Gamma \cup \{\rhd, \lhd\}$, $\delta(q, abc)$ contains at most a single MVR operation.*

Lemma 13. *For each ORWW-automaton $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, \delta, >)$, there exists an ORWW automaton M' with deterministic MVR operations that accepts the same language as M. If M accepts at the left sentinel, then so does M'.*

Proof. Using a variant of the well-known powerset construction, the ORWW-automaton M' can be defined as $M' = (2^Q, \Sigma, \Gamma, \rhd, \lhd, \{q_0\}, \delta', >)$, where, for all $\emptyset \neq S \subseteq Q$ and all $a, b, c \in \Gamma \cup \{\rhd, \lhd\}$,

$$T_{(S,abc)} = \{\, q \in Q \mid \exists s \in S : (q, \mathsf{MVR}) \in \delta(s, abc) \,\}, \text{ and}$$

$$\delta'(S, abc) = \begin{cases} \mathsf{Accept}, & \text{if } \exists s \in S : \delta(s, abc) = \mathsf{Accept}, \\ \left(\bigcup_{s \in S} \delta(s, abc) \cap \Gamma\right) \cup \{(T_{(S,abc)}, \mathsf{MVR})\}, & \text{if } T_{(S,abc)} \neq \emptyset, \\ \left(\bigcup_{s \in S} \delta(s, abc) \cap \Gamma\right), & \text{if } T_{(S,abc)} = \emptyset. \end{cases}$$ □

The next lemma is the second technical main result.

Lemma 14. *Let M be an ORWW-automaton with deterministic MVR operations that accepts at the left sentinel. From M a constant $N(M) > 0$ can be computed such that, for each $w \in L(M)$ satisfying $|w| \geq N(M)$ and each accepting computation C of M on input w, there are indices $1 \leq i < j \leq |w|$ such that $\tau_i^C(w) \sqsubseteq \tau_j^C(w)$ and these patterns contain the same rewrite operations.*

Proof. Let $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, \delta, >)$ be an ORWW-automaton with deterministic MVR operations that accepts at the left sentinel, and let $n = |\Gamma|$. Further, let $w \in L(M)$ and let C be an accepting computation of M on input w. The MVR operations executed at a position $1 \leq k \leq |w| - 1$ only depend on the prefix of length $k + 1$ of w. As M has deterministic MVR operations, the MVR operation that can be executed at position k is uniquely determined by that prefix, if it exists at all. For this reason a different MVR operation can become applicable at position k only if that prefix has been modified by a rewrite operation. This, however, can happen at most $(k + 1) \cdot (n - 1)$ times. Therefore, the pattern $\tau_k^C(w)$ contains at most $(k+1) \cdot (n-1) + 1$ MVR operations. Additionally, it contains at most $n - 1$ rewrite operations. Therefore, $\tau_k^C(w)$ has length at most $(k+1) \cdot (n-1) + n + 1 = k \cdot (n-1) + 2n$. Finally, we extend each pattern $\tau_k^C(w)$ into the word $\eta_k^C(w) = a_k \tau_k^C(w) s_k$ where a_k is the input letter at position k and s_k is the final letter produced by C at position k. Higman's theorem [2]

(see, also [4] and [12]) implies there exists a computable constant $N(M)$ such that, if $|w| \geq N(M)$, then there are indices $1 \leq i < j \leq N(M)$ such that $\eta_i^C(w)$ is a scattered subsequence of $\eta_j^C(w)$. This means that $a_i = a_j$ and $s_i = s_j$, and that $\tau_i^C(w)$ is a scattered subsequence of $\tau_j^C(w)$. As in both positions the letter $a_i = a_j$ is rewritten into the letter $s_i = s_j$, and as each rewrite operation at position i occurs in $\tau_i^C(w)$ and therewith also in $\tau_j^C(w)$, we see that $\tau_i^C(w)$ and $\tau_j^C(w)$ contain exactly the same rewrite operations. □

Now we can state and prove the announced Pumping Lemma.

Theorem 15 (Pumping Lemma). *For each ORWW-automaton M there exists a computable constant $N_p(M) > 0$ such that each word $w \in L(M)$, $|w| \geq N_p(M)$, has a factorization $w = xyz$ satisfying all of the following conditions:*

$$\text{(a) } |xy| \leq N_p(M),\ \text{(b) } |y| > 0,\ \textit{and}\ \text{(c) } xy^m z \in L(M)\ \textit{for all}\ m \geq 1.$$

Proof. Let M be an ORWW automaton. By Lemma 1 we may assume that M only accepts at the left sentinel. Further, by Lemma 13, we can convert M into an equivalent ORWW-automaton M_1 that is MVR-deterministic and that only accepts at the left sentinel. Then Lemma 14 implies that a constant $N_p(M)$ can be computed such that, for each $w \in L(M_1) = L(M)$ satisfying $|w| \geq N_p(M)$, and each accepting computation C of M_1 on input w, there are indices $1 \leq i < j \leq N_p(M)$ such that $\tau_i^C(w) \sqsubseteq \tau_j^C(w)$ and these patterns contain the same rewrite operations. Hence, by Lemma 11, w can be factored as $w = xyz$ such that $|xy| \leq N_p(M)$, $|y| > 0$, $xyyz \in L(M_1) = L(M)$, and $\tau_{|x|}^{C'}(xyyz) = \tau_{|xy|}^{C'}(xyyz)$, where C' is the accepting computation of M_1 for input $xyyz$ that is obtained from the computation C. Using Lemma 11 repeatedly we obtain that $xy^m z \in L(M_1) = L(M)$ holds for all $m \geq 1$. □

4 Applications of the Pumping Lemma

In [6] we have used the Cut-and-Paste Lemma to prove that emptiness is decidable for ORWW-automata. Here we show that also finiteness is decidable for ORWW-automata using both, the Cut-and-Paste Lemma and the Pumping Lemma.

Theorem 16. *The following* finiteness problem *is decidable:*
INSTANCE: An ORWW-automaton M.
QUESTION: Is the language $L(M)$ finite?

Proof. Let $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, \delta, >)$ be an ORWW-automaton, let $N_c(M)$ be the corresponding constant from the Cut-and-Paste Lemma for M, and $N_p(M)$ be the corresponding constant from the Pumping Lemma for M. We claim that $L(M)$ is finite iff it does not contain any word w such that $N_p(M) \leq |w| \leq N_p(M) + N_c(M)$.

Indeed, if $L(M)$ contains a word w such that $N_p(M) \leq |w| \leq N_p(M) + N_c(M)$, then the Pumping Lemma tells us that $L(M)$ is infinite. Conversely, if $L(M)$ is infinite, then it contains a word w of length at least $N_p(M)$. Assume that w is the shortest word with these properties. If $|w| \leq N_p(M) + N_c(M)$, then there is nothing to prove. On the other hand, if $|w| > N_p(M) + N_c(M)$, then we can apply the Cut-and-Paste Lemma to w, which yields a factorization $w = xyz$ such that $|yz| \leq N_c(M)$, $|y| > 0$, and $xz \in L(M)$. Thus, $|w| > |xz| = |w| - |y| \geq |w| - N_c(M) > N_p(M)$, which contradicts our choice of w. Hence, we see that $L(M)$ is infinite iff it contains a word w such that $N_p(M) \leq |w| \leq N_p(M) + N_c(M)$. □

The next result, which is also derived from the Pumping Lemma, shows that ORWW-automata only accept unary languages that are regular.

Theorem 17. *For each ORWW-automaton M, if the language $L(M)$ is unary, then it is already regular.*

Proof. Let M be an ORWW-automaton with input alphabet $\Sigma = \{a\}$, and let $\alpha = N_p(M)$ be the constant from the Pumping Lemma for M. For all integers c and d satisfying $0 \leq d < \alpha!$ and $0 < c \leq \alpha$, we let $S_{d,c} \subseteq \mathbb{N}$ be defined as follows:

$$S_{d,c} := \{\, n \geq \alpha \mid n \equiv d \mod \alpha! \text{ and } a^{n+c\cdot i} \in L(M) \text{ for all } i \in \mathbb{N} \,\}.$$

By definition $\{\, a^n \mid n \in S_{d,c} \,\} \subseteq L(M)$ for all pairs (d, c). On the other hand, if $a^n \in L(M)$ for some $n \geq \alpha$, then there exists an integer d, $0 \leq d < \alpha!$, such that $n \equiv d \mod \alpha!$. By the Pumping Lemma there also exists an integer c, $0 < c \leq \alpha$, such that $a^{n+c\cdot i} \in L(M)$ for all $i \in \mathbb{N}$. Hence, it follows that $n \in S_{d,c}$.

If $S_{d,c} \neq \emptyset$, it can be represented as the linear set $S_{d,c} = \{\, \min(S_{d,c}) + i \cdot \alpha! \mid i \in \mathbb{N} \,\}$. Therefore, if $\psi : \Sigma^* \to \mathbb{N}$ denotes the Parikh mapping defined by $a^n \mapsto n$ $(n \geq 0)$, then

$$\psi(L(M)) = \{\, n < \alpha \mid a^n \in L(M) \,\} \cup \bigcup_{d,c} S_{d,c},$$

which shows that $\psi(L(M))$ is a semi-linear subset of $\mathbb{N}$. Thus, it follows that $L(M)$ is indeed a regular language. □

Actually, it can be shown that a regular expression can be determined for the language $L(M)$ of an ORWW-automaton M that has a unary input alphabet.

5 Concluding Remarks

We have established a Pumping Lemma for ORWW-automata that nicely complements the Cut-and-Paste Lemma for these automata presented in [6]. Observe that the Cut-and-Paste Lemma tells us that we can cut from the suffix of a sufficiently long word, while the Pumping Lemma tells us that we can pump within the prefix of a sufficiently long word. This effect is clearly demonstrated by the

language $L = \{ a^m b^n \mid m \geq n \} \in \mathcal{L}(\mathsf{ORWW})$ [6], as from a word $a^m b^m \in L$, where m is a sufficiently large integer, the Cut-and-Paste Lemma yields a word of the form $a^m b^{m-i}$, and the Pumping Lemma gives words of the form $a^{m+c \cdot i} b^m$. From the Pumping Lemma we have then derived the solvability of the finiteness problem for ORWW-automata and the fact that the only unary languages accepted by these automata are the regular ones.

However, there still remain many open questions. For example, is it true that ORWW-automata only accept languages that are semi-linear? Further, given an ORWW-automaton M and a regular language R (for example, through a DFA), it can be checked whether $L(M)$ is contained in R, as this is the case iff $L(M) \cap R^c$ is empty. However, it is still open whether the converse inclusion (that is, is R contained in $L(M)$) can be checked. A special case is the *universality problem*, that is, given an ORWW-automaton M with input alphabet Σ, is $L(M)$ all of Σ^*? Finally, one may ask whether ORWW-automata yield more succinct representations for unary languages than deterministic ORWW-automata.

References

1. E. Dahlhaus and M. Warmuth. Membership for growing context-sensitive grammars is polynomial. *Journal of Computer and System Sciences*, 33:456–472, 1986.
2. G. Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 2:326–336, 1952.
3. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, M.A., 1979.
4. P. Karandikar and Ph. Schnoebelen. Generalized Post embedding problems. *Theory Comput. Syst.*, 56:697–716, 2015.
5. K. Kwee and F. Otto. On some decision problems for stateless deterministic ordered restarting automata. In J. Shallit and A. Okhotin, editors, *DCFS 2015, Proc.*, *LNCS 9118*, pages 165–176. Springer, Heidelberg, 2015.
6. K. Kwee and F. Otto. On the effects of nondeterminism on ordered restarting automata. In R.M. Freivalds, G. Engels, and B. Catania, editors, *SOFSEM 2016, Proc.*, Lecture Notes in Computer Science 9587, pages 369–380. Springer, Heidelberg, 2016.
7. K. Kwee and F. Otto. Nondeterministic ordered restarting automata. 2017. Submitted.
8. R. McNaughton, P. Narendran, and F. Otto. Church-Rosser Thue systems and formal languages. *Journal of the Association for Computing Machinery*, 35:324–344, 1988.
9. F. Mráz and F. Otto. Ordered restarting automata for picture languages. In V. Geffert, B. Preneel, B. Rovan, J. Štuller, and A. Min Tjoa, editors, *SOFSEM 2014, Proc.*, *LNCS 8327*, pages 431–442. Springer, Heidelberg, 2014.
10. F. Mráz, M. Plátek, and Procházka. M. On special forms of restarting automata. *Grammars*, 2:223–233, 1999.
11. F. Otto. On the descriptional complexity of deterministic ordered restarting automata. In H. Jürgensen, J. Karhumäki, and A. Okhotin, editors, *DCFS 2014, Proc.*, *LNCS 8614*, pages 318–329. Springer, Heidelberg, 2014.
12. S. Schmitz and Ph. Schnoebelen. Multiply-recursive upper bounds with Higman's lemma. In L. Aceto, M. Henzinger, and J. Sgall, editors, *ICALP 2011, Proc., Part II*, *LNCS 6756*, pages 441–452. Springer, Heidelberg, 2011.