# On the Effects of Nondeterminism on Ordered Restarting Automata

**Friedrich Otto** and **Kent Kwee**

Fachbereich Elektrotechnik/Informatik
Universität Kassel
34109 Kassel, Germany

{otto,kwee}@theory.informatik.uni-kassel.de

**SOFSEM 2016**

Harrachov, Czech Republic

January 23–28, 2016

Outline:


1. Introduction
2. Ordered Restarting Automata
3. Closure Properties
4. Stateless Ordered Restarting Automata
5. Conclusion

# 1. Introduction

## Characterizations of REG

Many different types of automata characterize the class REG:

- the deterministic finite-state acceptor (DFA)
- the nondeterministic finite-state acceptor (NFA)
- the alternating finite-state acceptor (AFA)
- the two-way finite-state acceptor (2NFA)
- the deterministic ordered restarting automaton (det-ORWW)
- the stateless deterministic ordered restarting automaton (stl-det-ORWW)

How easy are these automata to explain and to understand?

# 1. Introduction

## Characterizations of REG

Many different types of automata characterize the class REG:

- the deterministic finite-state acceptor (DFA)
- the nondeterministic finite-state acceptor (NFA)
- the alternating finite-state acceptor (AFA)
- the two-way finite-state acceptor (2NFA)
- the deterministic ordered restarting automaton (det-ORWW)
- the stateless deterministic ordered restarting automaton (stl-det-ORWW)

How easy are these automata to explain and to understand?

## Descriptional Complexity

How succinctly do these automata represent a particular language, when we take the number of states (or the number of letters) as a measure for their size?

Various trade-offs have been established:

- NFA to DFA: $2^n$
- AFA to DFA: $2^{2^n}$ (Chandra, Kozen, Stockmeyer, 1981)
- stateless det-ORWW to DFA: $2^{2^{O(n)}}$ (O, 2014)
- stl-det-ORWW to NFA: $2^{O(n)}$ (KO, 2015)

Here we are interested in the

nondeterministic ORWW-automaton and its stateless variant!

## Descriptional Complexity

How succinctly do these automata represent a particular language, when we take the number of states (or the number of letters) as a measure for their size?

Various trade-offs have been established:

- NFA to DFA: $2^n$
- AFA to DFA: $2^{2^n}$ (Chandra, Kozen, Stockmeyer, 1981)
- stateless det-ORWW to DFA: $2^{2^{O(n)}}$ (O, 2014)
- stl-det-ORWW to NFA: $2^{O(n)}$ (KO, 2015)

Here we are interested in the

nondeterministic ORWW-automaton and its stateless variant!

## Descriptional Complexity

How succinctly do these automata represent a particular language, when we take the number of states (or the number of letters) as a measure for their size?

Various trade-offs have been established:

- NFA to DFA: $2^n$
- AFA to DFA: $2^{2^n}$ (Chandra, Kozen, Stockmeyer, 1981)
- stateless det-ORWW to DFA: $2^{2^{O(n)}}$ (O, 2014)
- stl-det-ORWW to NFA: $2^{O(n)}$ (KO, 2015)

Here we are interested in the

nondeterministic ORWW-automaton and its stateless variant!

## Descriptional Complexity

How succinctly do these automata represent a particular language, when we take the number of states (or the number of letters) as a measure for their size?

Various trade-offs have been established:

- NFA to DFA: $2^n$
- AFA to DFA: $2^{2^n}$ (Chandra, Kozen, Stockmeyer, 1981)
- stateless det-ORWW to DFA: $2^{2^{O(n)}}$ (O, 2014)
- stl-det-ORWW to NFA: $2^{O(n)}$ (KO, 2015)

Here we are interested in the

nondeterministic ORWW-automaton and its stateless variant!

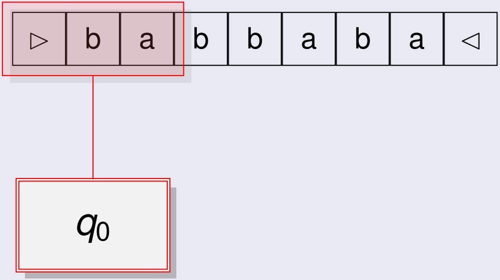
# 2. Ordered Restarting Automata

## Definition 1

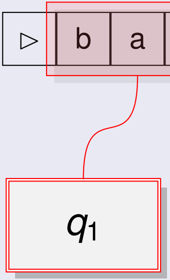
*A (nondeterministic) ORWW-automaton (ORWW) is a one-tape machine that is described by an 8-tuple* $M = (Q, \Sigma, \Gamma, \triangleright, \triangleleft, q_0, \delta, >)$*:*

- $Q$ *is a finite set of states,*
- $\Sigma$ *is a finite input alphabet,*
- $\Gamma$ *is a finite tape alphabet containing* $\Sigma$*,*
- *the symbols* $\triangleright, \triangleleft \notin \Gamma$ *serve as markers for the left and right border of the work space, respectively,*
- $q_0 \in Q$ *is the initial state,*
- $\delta : Q \times (((\Gamma \cup \{\triangleright\}) \cdot \Gamma \cdot (\Gamma \cup \{\triangleleft\})) \cup \{\triangleright\triangleleft\}) \to 2^{(Q \times \{\mathsf{MVR}\}) \cup \Gamma \cup \{\mathsf{Accept}\}}$ *is the transition relation, and*
- $>$ *is a partial ordering on* $\Gamma$*.*

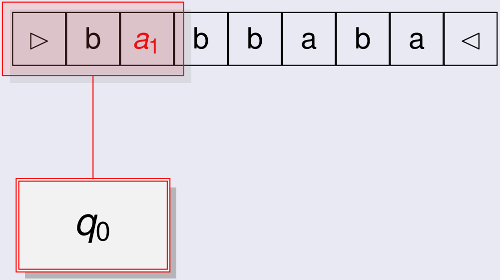
## Definition 1 (cont.)



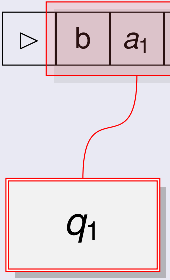
## Definition 1 (cont.)



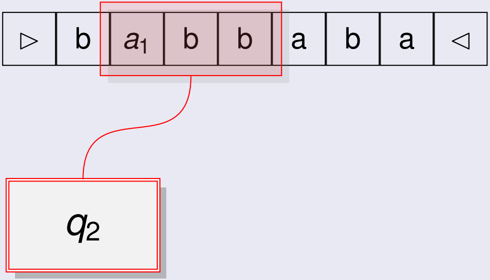
## Definition 1 (cont.)



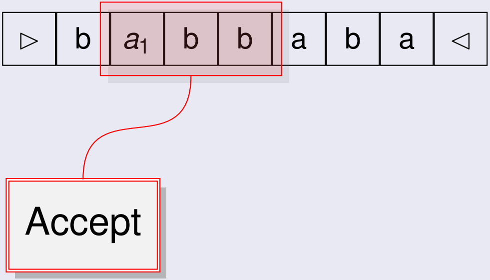
## Definition 1 (cont.)



## Definition 1 (cont.)



## Definition 1 (cont.)



## Definition 1 (cont.)

An input $w \in \Sigma^*$ is accepted by $M$, if there exists a computation of $M$ which starts with the initial configuration $q_0 \rhd w \lhd$, and which finally ends with executing an Accept instruction.

$L(M)$ is the language consisting of all words that are accepted by $M$.

The ORWW-automaton $M$ is called stateless if $Q = \{q_0\}$.
By stl-ORWW we denote the stateless ORWW-automata.

As each cycle ends with a rewrite operation, which replaces a symbol $a$ by a symbol $b$ that is strictly smaller with respect to $>$, each computation of $M$ on input $w$ consists of $\leq |w| \cdot (|\Gamma| - 1)$ cycles.

## Definition 1 (cont.)

An input $w \in \Sigma^*$ is accepted by $M$, if there exists a computation of $M$ which starts with the initial configuration $q_0 \rhd w \lhd$, and which finally ends with executing an Accept instruction.

$L(M)$ is the language consisting of all words that are accepted by $M$.

The ORWW-automaton $M$ is called stateless if $Q = \{q_0\}$.
By stl-ORWW we denote the stateless ORWW-automata.

As each cycle ends with a rewrite operation, which replaces a symbol $a$ by a symbol $b$ that is strictly smaller with respect to $>$, each computation of $M$ on input $w$ consists of $\leq |w| \cdot (|\Gamma| - 1)$ cycles.

## Definition 1 (cont.)

An input $w \in \Sigma^*$ is accepted by $M$, if there exists a computation of $M$ which starts with the initial configuration $q_0 \rhd w \lhd$, and which finally ends with executing an Accept instruction.

$L(M)$ is the language consisting of all words that are accepted by $M$.

The ORWW-automaton $M$ is called stateless if $Q = \{q_0\}$.
By stl-ORWW we denote the stateless ORWW-automata.

As each cycle ends with a rewrite operation, which replaces a symbol $a$ by a symbol $b$ that is strictly smaller with respect to $>$, each computation of $M$ on input $w$ consists of $\leq |w| \cdot (|\Gamma| - 1)$ cycles.

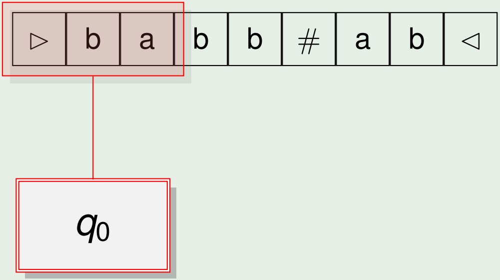
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



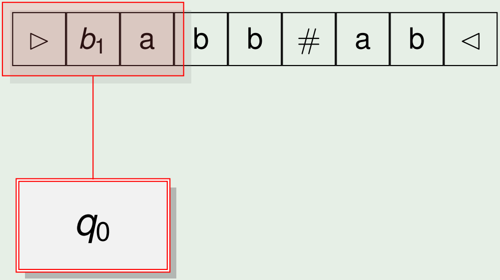
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



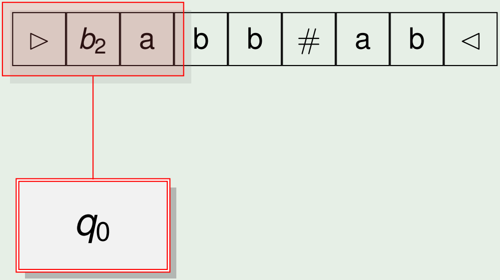
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\}:$$



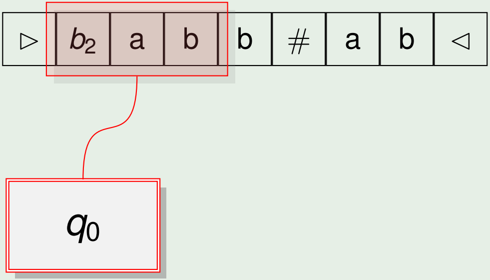
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



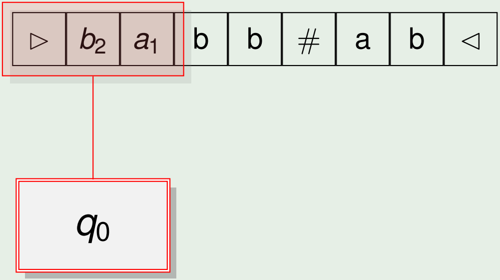
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



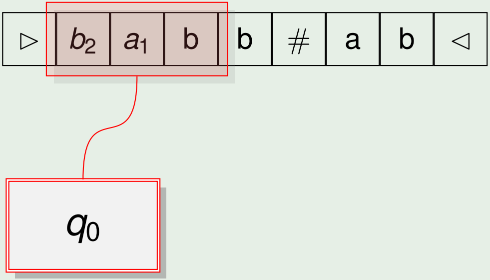
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



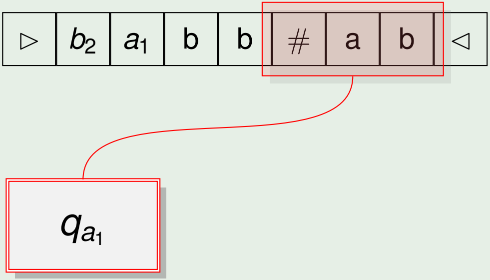
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a,b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



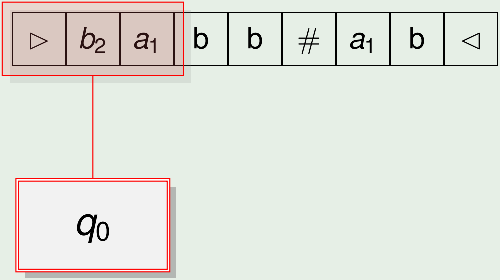
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



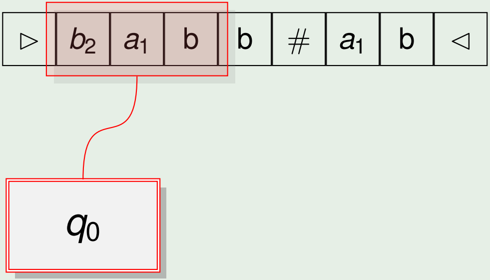
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a,b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



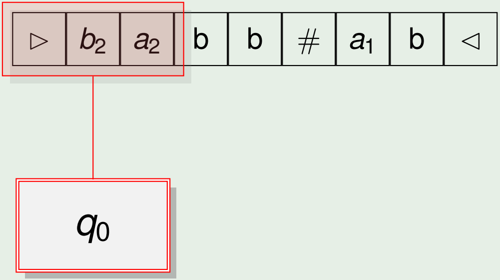
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



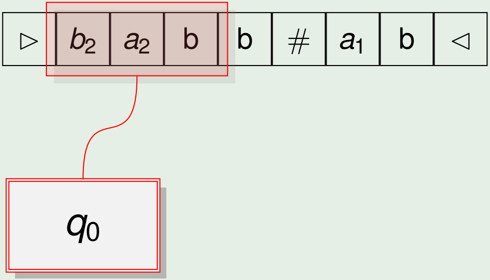
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



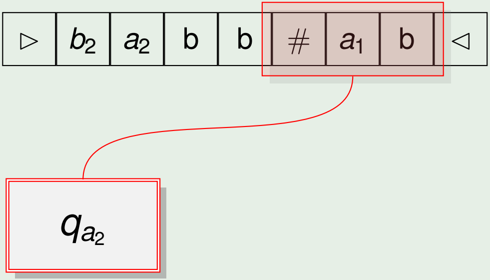
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



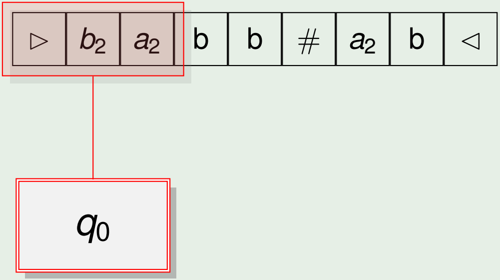
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



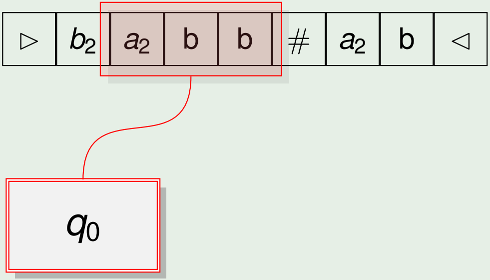
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



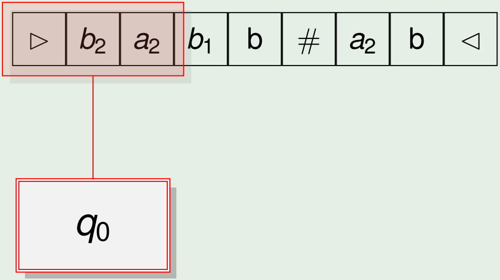
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



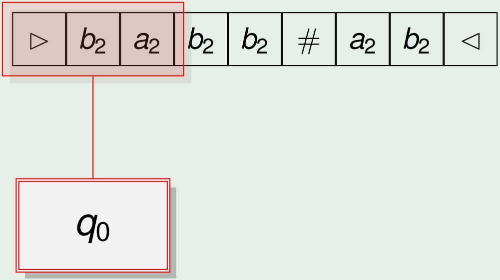
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{\, w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \,\} :$$



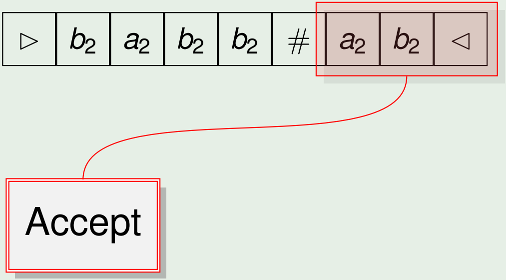
## Example 1

An ORWW-automaton $M$ on

$$\Sigma = \{a, b, \#\} \text{ and } \Gamma = \{a, a_1, a_2, b, b_1, b_2, \#\}$$

using the linear ordering $\# > a > b > a_1 > b_1 > a_2 > b_2$ for

$$L'_{\text{copy}} = \{ w\#u \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ a subsequence of } w \} :$$



**Theorem 2 (G. Buntrock, F. O. (1998))**

GCSL $\subseteq \mathcal{L}$(OW-auxPDA(log))

**Theorem 3 (C. Lautemann (1988))**

$L^{\#}_{copy} = \{\, w\#w \mid w \in \{a,b\}^* \,\} \notin \mathcal{L}$(OW-auxPDA(log))

**Proposition 4**

$L'_{copy} \notin$ GCSL

**Proof.**

Assume $L'_{\text{copy}} \in$ GCSL. Then there is a OW-auxPDA that accepts that language. We can simply add an extra track to implement a binary counter to accept $L^{\#}_{\text{copy}}$. □

**Corollary 5**

$\mathcal{L}$(ORWW) $\not\subseteq$ GCSL

**Theorem 2 (G. Buntrock, F. O. (1998))**

GCSL $\subseteq \mathcal{L}$(OW-auxPDA(log))

**Theorem 3 (C. Lautemann (1988))**

$L^{\#}_{copy} = \{\, w\#w \mid w \in \{a,b\}^* \,\} \notin \mathcal{L}$(OW-auxPDA(log))

**Proposition 4**

$L'_{copy} \notin$ GCSL

**Proof.**

Assume $L'_{\text{copy}} \in$ GCSL. Then there is a OW-auxPDA that accepts that language. We can simply add an extra track to implement a binary counter to accept $L^{\#}_{\text{copy}}$. □

**Corollary 5**

$\mathcal{L}$(ORWW) $\not\subseteq$ GCSL

### Theorem 2 (G. Buntrock, F. O. (1998))

GCSL $\subseteq \mathcal{L}$(OW-auxPDA(log))

### Theorem 3 (C. Lautemann (1988))

$L^{\#}_{copy} = \{\, w\#w \mid w \in \{a,b\}^* \,\} \notin \mathcal{L}$(OW-auxPDA(log))

### Proposition 4

$L'_{copy} \notin$ GCSL

### Proof.

Assume $L'_{\text{copy}} \in$ GCSL. Then there is a OW-auxPDA that accepts that language. We can simply add an extra track to implement a binary counter to accept $L^{\#}_{\text{copy}}$. □

### Corollary 5

$\mathcal{L}$(ORWW) $\not\subseteq$ GCSL

## Theorem 2 (G. Buntrock, F. O. (1998))

GCSL $\subseteq \mathcal{L}$(OW-auxPDA(log))

## Theorem 3 (C. Lautemann (1988))

$L^{\#}_{copy} = \{ w\#w \mid w \in \{a, b\}^* \} \notin \mathcal{L}$(OW-auxPDA(log))

## Proposition 4

$L'_{copy} \notin$ GCSL

## Proof.

Assume $L'_{\text{copy}} \in$ GCSL. Then there is a OW-auxPDA that accepts that language. We can simply add an extra track to implement a binary counter to accept $L^{\#}_{\text{copy}}$. □

## Corollary 5

$\mathcal{L}$(ORWW) $\not\subseteq$ GCSL

## Theorem 6 (Cut-and-Paste Lemma)

*For each ORWW-automaton M, there exists a constant $N(M) > 0$ such that each word $w \in L(M), |w| \geq N(M)$, has a factorization $w = xyz$ satisfying the following conditions*

- $|yz| \leq N(M)$
- $|y| > 0$
- $xz \in L(M)$

## Example:

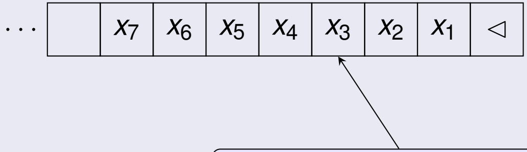
$$abcdefghijklmn \in L(M) \rightarrow abcdehijklmn \in L(M)$$
$$\rightarrow abcdeklmn \in L(M)$$

## Proof of Cut-and-Paste Lemma:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ... | | b | b | a | a | a | a | a | $\triangleleft$ |

- Assume the automaton accepts at the left sentinel.
- Consider shortest accepting computation.
- Look at sequences of operations at each position.

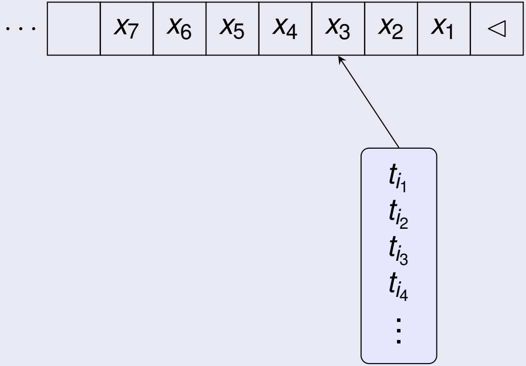
## Proof of Cut-and-Paste Lemma:




$$\begin{array}{c} (q_1, a_1 b_1 c_1) \rightarrow d_1 \\ (q_2, a_2 b_2 c_2) \rightarrow (q', \mathrm{MVR}) \\ (q_3, a_3 b_3 c_3) \rightarrow d_3 \\ (q_4, a_4 b_4 c_4) \rightarrow d_4 \\ \vdots \end{array}$$

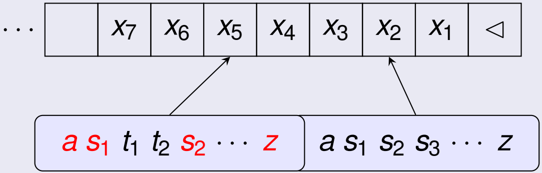
- Introduce a new alphabet $\Omega$ in 1-to-1 correspondence to these operations.

## Proof of Cut-and-Paste Lemma:



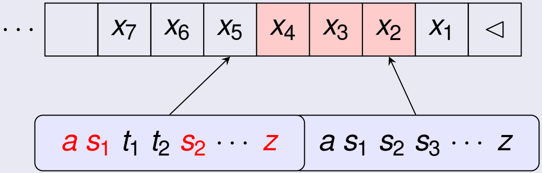

- Sequences are finite words.
- Length of words is non-decreasing and limited by $|x_j| \leq j \cdot (n-1)$.

## Proof of Cut-and-Paste Lemma:




- Higman's theorem $\rightarrow$ If a sequence of words is $\geq N(M)$, there are $i < j$ such that $x_i$ is scattered subsequence of $x_j$.
- $N(M) = H(2, n+1, \Omega \cup \Gamma) + 2$ (The length function $H$ is a total recursive function).
- $t$-letters can only be MVR operations as rewrite operations must be valid.

## Proof of Cut-and-Paste Lemma:




- Cut out the factor between these two positions.
- Consider the cycles of the original computation:
  - Keep all cycles that do not contribute to $x_j$.
  - Keep those cycles that contribute a rewrite operation to $x_j$.
  - Splice the initial part of a cycle that contributes a MVR-step $s$ to $x_j$ with the final part of the cycle that contributes the same $s$ to $x_i$.
  - Leave out cycles that contribute a MVR-step $t$ to $x_j$.
- We have a valid computation for the shorter word.

□

**Theorem 7**

*The emptiness problem for ORWW-automata is decidable.*

**Proof:**

Let $M$ be an ORWW-automaton and $N(M)$ be the corresponding constant from the Cut-and-Paste Lemma.

$$L(M) \neq \emptyset \text{ iff } L(M) \text{ contains a word of length } \leq N(M)$$

□

## Theorem 7

*The emptiness problem for ORWW-automata is decidable.*

## Proof:

Let $M$ be an ORWW-automaton and $N(M)$ be the corresponding constant from the Cut-and-Paste Lemma.

$$L(M) \neq \emptyset \text{ iff } L(M) \text{ contains a word of length } \leq N(M)$$

□

# 3. Closure Properties

**Theorem 8**

$\mathcal{L}$(ORWW) *is closed under union, intersection, product, Kleene star, inverse morphisms, and non-erasing morphisms.*

As REG $= \mathcal{L}$(det-ORWW) $\subset \mathcal{L}$(ORWW), it follows that $\mathcal{L}$(ORWW) is in particular closed under intersection with regular languages.

**Corollary 9**

$\mathcal{L}$(ORWW) *is an AFL (Abstract Family of Languages).*

# 3. Closure Properties

**Theorem 8**

$\mathcal{L}$(ORWW) *is closed under union, intersection, product, Kleene star, inverse morphisms, and non-erasing morphisms.*

As REG $= \mathcal{L}$(det-ORWW) $\subset \mathcal{L}$(ORWW), it follows that $\mathcal{L}$(ORWW) is in particular closed under intersection with regular languages.

**Corollary 9**

$\mathcal{L}$(ORWW) *is an AFL (Abstract Family of Languages).*

# 3. Closure Properties

**Theorem 8**

$\mathcal{L}$(ORWW) *is closed under union, intersection, product, Kleene star, inverse morphisms, and non-erasing morphisms.*

As REG $= \mathcal{L}$(det-ORWW) $\subset \mathcal{L}$(ORWW), it follows that $\mathcal{L}$(ORWW) is in particular closed under intersection with regular languages.

**Corollary 9**

$\mathcal{L}$(ORWW) *is an AFL (Abstract Family of Languages).*

## Theorem 10

$L_{\leq} = \{ a^m b^n \mid 1 \leq m \leq n \} \notin \mathcal{L}(\text{ORWW}).$

## Proof:

- Assume that there exists an ORWW-automaton $M$ such that $L(M) = L_{\leq}$.
- Let $N(M)$ be the constant for $M$ from the Cut-and-Paste Lemma.
- Consider the word $a^{N(M)} b^{N(M)} \in L(M)$.
- Then $a^{N(M)} b^{N(M)-i} \in L(M)$ for some $i \geq 1 \rightarrow$ contradiction!
- $L_{\leq}$ is not accepted by any ORWW-automaton.

□

## Theorem 10

$L_{\leq} = \{ a^m b^n \mid 1 \leq m \leq n \} \notin \mathcal{L}(\text{ORWW})$.

## Proof:

- Assume that there exists an ORWW-automaton $M$ such that $L(M) = L_{\leq}$.
- Let $N(M)$ be the constant for $M$ from the Cut-and-Paste Lemma.
- Consider the word $a^{N(M)} b^{N(M)} \in L(M)$.
- Then $a^{N(M)} b^{N(M)-i} \in L(M)$ for some $i \geq 1 \rightarrow$ contradiction!
- $L_{\leq}$ is not accepted by any ORWW-automaton.

□

## Corollary 11

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

## Corollary 12

*The language class* $\mathcal{L}(\mathrm{ORWW})$ *is neither closed under the operation of reversal nor under complementation.*

## Proof:

- $L_{\geq} = \{\, b^m a^n \mid m \geq n \geq 1 \,\} \in \mathcal{L}(\mathrm{ORWW})$
- $L_{\geq}^R = L_{\leq} = \{\, a^m b^n \mid n \geq m \geq 1 \,\} \notin \mathcal{L}(\mathrm{ORWW})$
- $L_{\geq}^c \cap (b^+ \cdot a^+) = \{\, b^m a^n \mid n > m \geq 1 \,\} \notin \mathcal{L}(\mathrm{ORWW})$

□

## Corollary 11

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

## Corollary 12

*The language class* $\mathcal{L}(\text{ORWW})$ *is neither closed under the operation of reversal nor under complementation.*

**Proof:**

- $L_{\geq} = \{ b^m a^n \mid m \geq n \geq 1 \} \in \mathcal{L}(\text{ORWW})$
- $L_{\geq}^R = L_{\leq} = \{ a^m b^n \mid n \geq m \geq 1 \} \notin \mathcal{L}(\text{ORWW})$
- $L_{\geq}^c \cap (b^+ \cdot a^+) = \{ b^m a^n \mid n > m \geq 1 \} \notin \mathcal{L}(\text{ORWW})$

□

## Corollary 11

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

## Corollary 12

*The language class $\mathcal{L}(\mathrm{ORWW})$ is neither closed under the operation of reversal nor under complementation.*

## Proof:

- $L_{\geq} = \{\, b^m a^n \mid m \geq n \geq 1 \,\} \in \mathcal{L}(\mathrm{ORWW})$
- $L_{\geq}^R = L_{\leq} = \{\, a^m b^n \mid n \geq m \geq 1 \,\} \notin \mathcal{L}(\mathrm{ORWW})$
- $L_{\geq}^c \cap (b^+ \cdot a^+) = \{\, b^m a^n \mid n > m \geq 1 \,\} \notin \mathcal{L}(\mathrm{ORWW})$

□

## Corollary 11

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

## Corollary 12

*The language class* $\mathcal{L}(\text{ORWW})$ *is neither closed under the operation of reversal nor under complementation.*

## Proof:

- $L_{\geq} = \{ b^m a^n \mid m \geq n \geq 1 \} \in \mathcal{L}(\text{ORWW})$
- $L_{\geq}^R = L_{\leq} = \{ a^m b^n \mid n \geq m \geq 1 \} \notin \mathcal{L}(\text{ORWW})$
- $L_{\geq}^c \cap (b^+ \cdot a^+) = \{ b^m a^n \mid n > m \geq 1 \} \notin \mathcal{L}(\text{ORWW})$

□

## Corollary 11

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

## Corollary 12

*The language class* $\mathcal{L}(\mathrm{ORWW})$ *is neither closed under the operation of reversal nor under complementation.*

## Proof:

- $L_{\geq} = \{\, b^m a^n \mid m \geq n \geq 1 \,\} \in \mathcal{L}(\mathrm{ORWW})$
- $L_{\geq}^{R} = L_{\leq} = \{\, a^m b^n \mid n \geq m \geq 1 \,\} \notin \mathcal{L}(\mathrm{ORWW})$
- $L_{\geq}^{c} \cap (b^+ \cdot a^+) = \{\, b^m a^n \mid n > m \geq 1 \,\} \notin \mathcal{L}(\mathrm{ORWW})$

□

# 4. Stateless Ordered Restarting Automata

**Theorem 13**

*From a stl-ORWW-automaton $M = (\Sigma, \Gamma, \triangleright, \triangleleft, \delta, >)$, an NFA $A$ with $2^{O(|\Gamma|)}$ states can be constructed such that $L(A) = L(M)$.*

**Corollary 14**

$\mathcal{L}$(stl-ORWW) = REG = $\mathcal{L}$(stl-det-ORWW).

How succinct are stl-ORWW-automata in relation to stl-det-ORWW-automata?

# 4. Stateless Ordered Restarting Automata

**Theorem 13**

*From a stl-ORWW-automaton $M = (\Sigma, \Gamma, \triangleright, \triangleleft, \delta, >)$, an NFA $A$ with $2^{O(|\Gamma|)}$ states can be constructed such that $L(A) = L(M)$.*

**Corollary 14**

$\mathcal{L}(\text{stl-ORWW}) = \text{REG} = \mathcal{L}(\text{stl-det-ORWW})$.

How succinct are stl-ORWW-automata in relation to stl-det-ORWW-automata?

# 4. Stateless Ordered Restarting Automata

**Theorem 13**

*From a stl-ORWW-automaton $M = (\Sigma, \Gamma, \triangleright, \triangleleft, \delta, >)$, an NFA $A$ with $2^{O(|\Gamma|)}$ states can be constructed such that $L(A) = L(M)$.*

**Corollary 14**

$\mathcal{L}(\text{stl-ORWW}) = \text{REG} = \mathcal{L}(\text{stl-det-ORWW})$.

How succinct are stl-ORWW-automata in relation to stl-det-ORWW-automata?

## Theorem 15

*For all $n \geq 2$, the language*

$$C_n = \{\, u_1\#u_2\#\ldots\#u_m \mid m \geq 2, u_1, \ldots, u_m \in \{a,b\}^n, \exists i < j : u_i = u_j \,\}$$

*has the following properties:*

- *It is accepted by a stl-ORWW-automaton $A_n$ with a tape alphabet of size $O(n)$.*
- *Every stl-det-ORWW-automaton for $C_n$ needs at least $2^{O(n)}$ letters.*

# 5. Conclusion

- $\mathcal{L}$(ORWW) is an AFL incomparable to DLIN, LIN, CFL, CRL, and GCSL, and it has decidable emptiness problem.
- $\mathcal{L}$(stl-ORWW) $=$ REG, but stl-ORWW-automata describe some languages exponentially more succinctly than even stl-det-ORWW-automata.
- Is $\mathcal{L}$(ORWW) closed under arbitrary morphisms?
- Is there an efficient algorithm for deciding the emptiness problem for ORWW-automata?
- Are finiteness, inclusion, or equivalence decidable for ORWW-automata?
- Find an efficient transformation from stl-ORWW-automata to stl-det-ORWW-automata!

# Thank you for your attention!