# ON SHRINKING RESTARTING AUTOMATA

Friedrich Otto

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

NCMA 2017

Prague

August 17–18, 2017

# 1. Introduction

Linguistic task: Analyze sentences of a natural language

- verify syntactic correctness
- determine dependencies
- disambiguate between morphological ambiguities

A linguistic technique: Analysis by reduction

Application: natural languages with a free word order
(Czech, Polish, Russian, Turkish, ...)

Operations used in the Analysis by Reduction:

Move-right/Move-left: MVR, MVL

Delete/Rewrite: replace a small part by a shorter/simpler one

Restart: start anew on a simplified sentence

Operations used in the Analysis by Reduction:

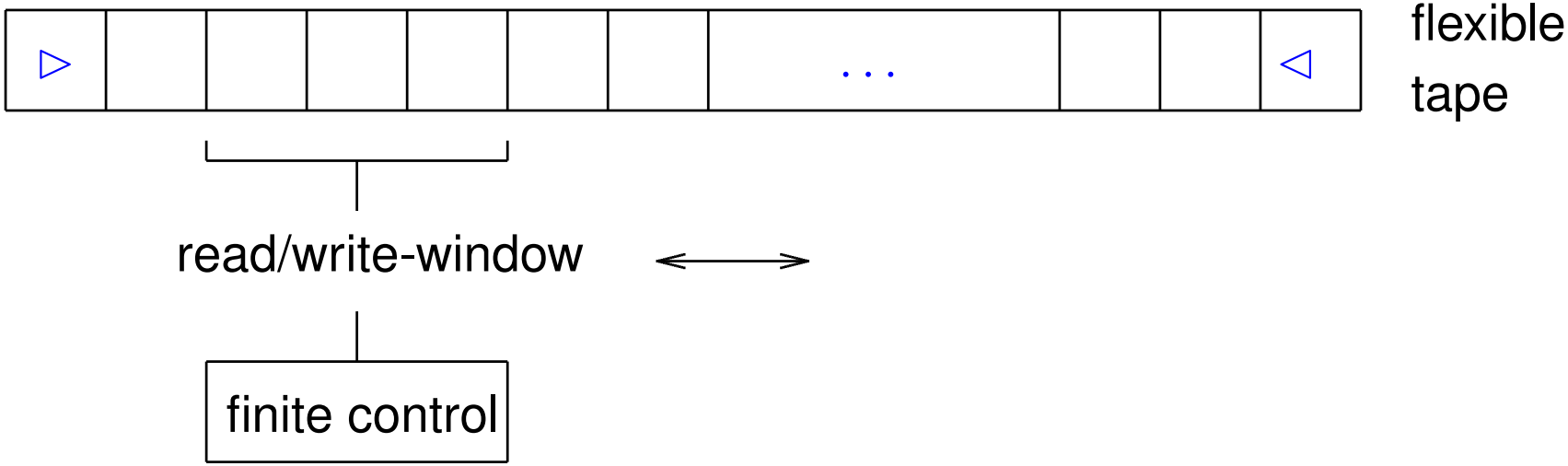
Move-right/Move-left: MVR, MVL

Delete/Rewrite: replace a small part by a shorter/simpler one

Restart: start anew on a simplified sentence

Jančar, Mráz, Plátek, Vogel (1995 – 1999):

Restarting Automaton



# 2. Definitions

## RLWW-Automata

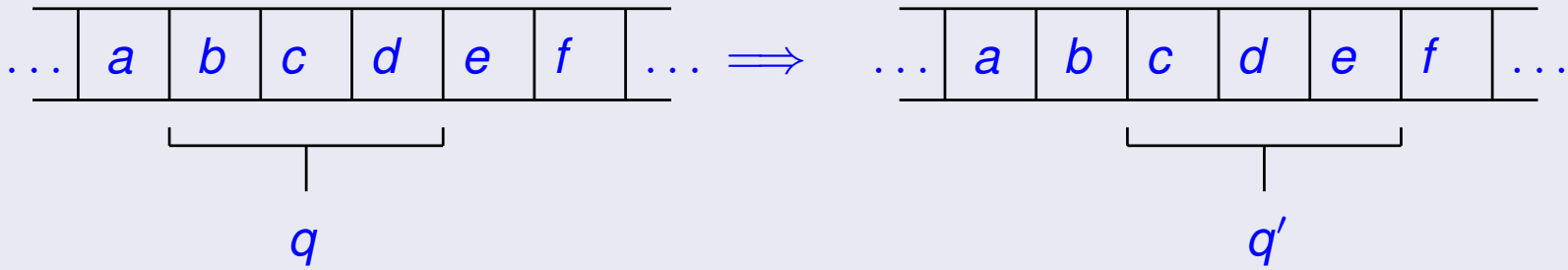
An RLWW-automaton is described as $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, k, \delta)$ :

- $Q$ : finite set of states
- $\Sigma$ : finite input alphabet
- $\Gamma$ : finite tape alphabet, $\Gamma \supseteq \Sigma$
- $\rhd, \lhd$ : left and right delimiters ($\rhd, \lhd \notin \Gamma$)
- $q_0$ : initial state from $Q$
- $k$ : size of the read/write window ($k \geq 1$)
- $\delta$ : $Q \times \mathcal{P}C^{(k)} \to P_{\text{fin}}$ (operations)

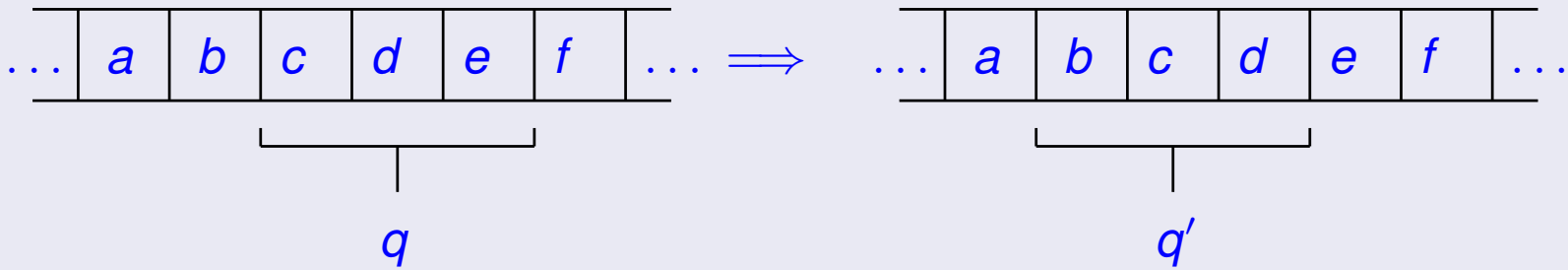
where $\mathcal{P}C^{(k)} := (\rhd \cdot \Gamma^{k-1}) \cup \Gamma^k \cup (\Gamma^{\leq k-1} \cdot \lhd) \cup (\rhd \cdot \Gamma^{\leq k-2} \cdot \lhd)$

## Transition Steps

(1.) MVR-step: $(q', \mathrm{MVR}) \in \delta(q, u)$ :

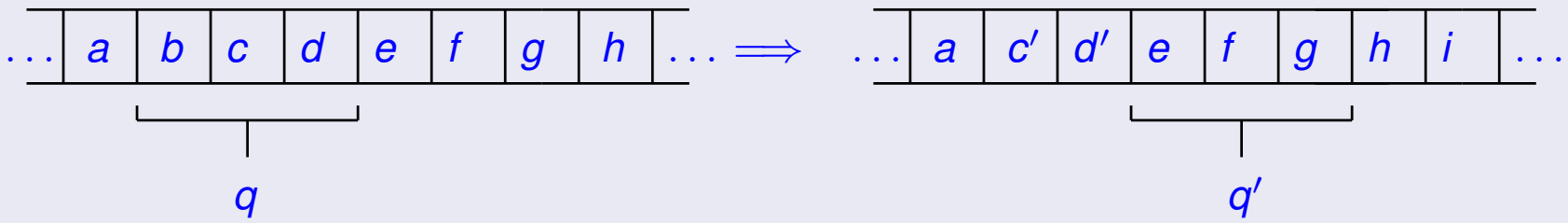



(2.) MVL-step: $(q', \mathrm{MVL}) \in \delta(q, u)$ :



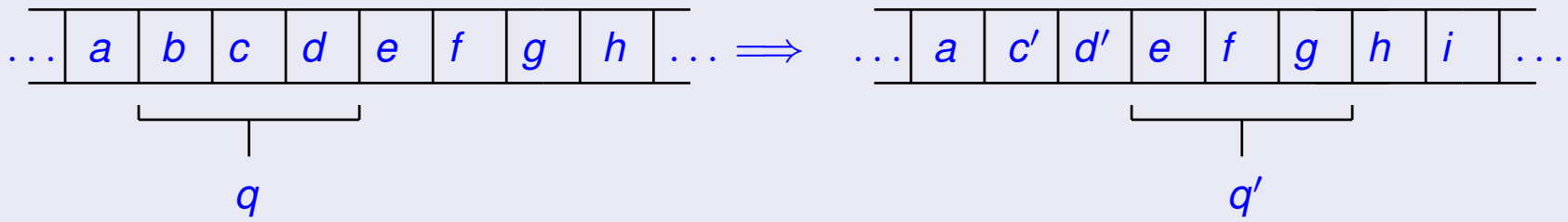
## Transition Steps (cont.)

(3.) Rewrite-step: $(q', v) \in \delta(q, u)$ :

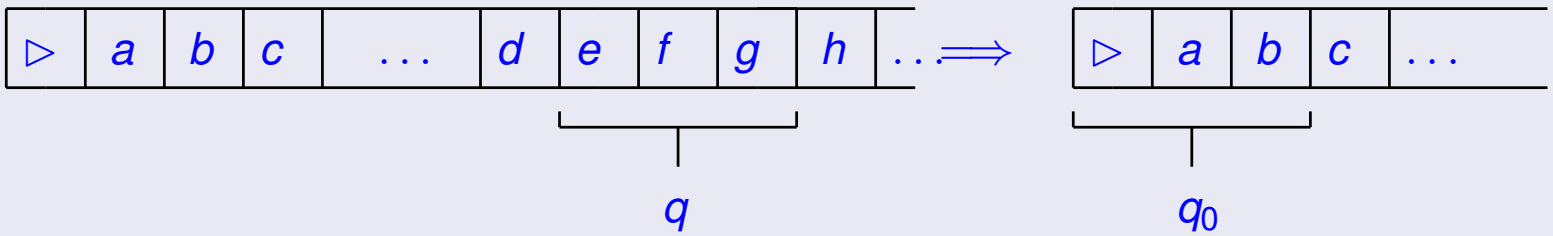


## Transition Steps (cont.)

(3.) Rewrite-step: $(q', v) \in \delta(q, u)$ :

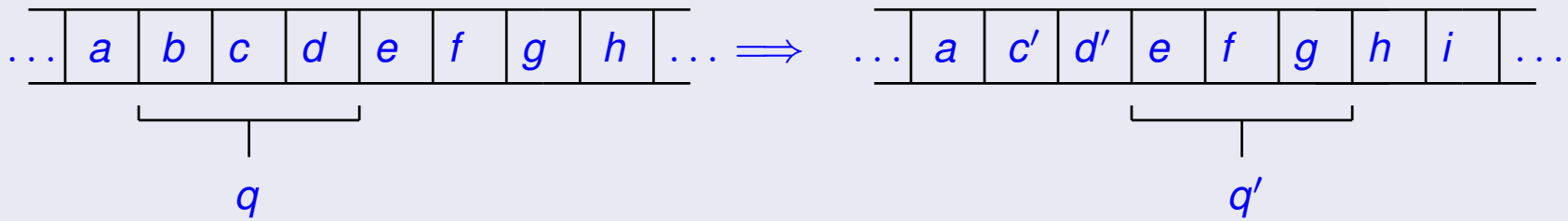



(4.) Restart-step: Restart $\in \delta(q, u)$ :

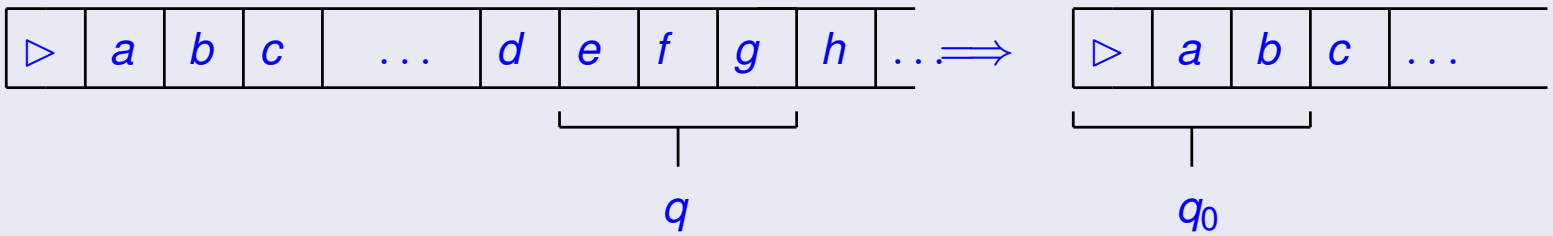


## Transition Steps (cont.)

(3.) Rewrite-step: $(q', v) \in \delta(q, u)$ :



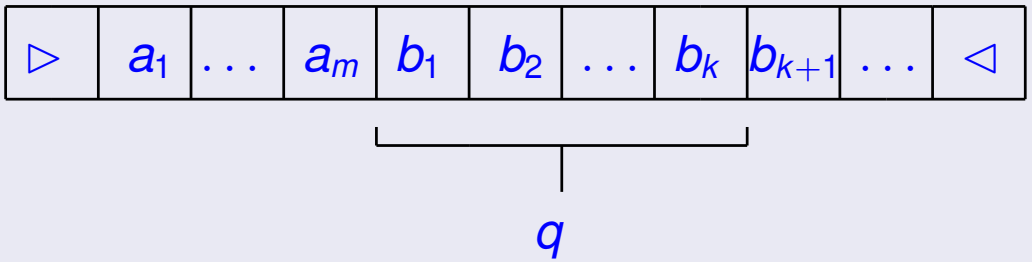

(4.) Restart-step: Restart $\in \delta(q, u)$ :




(5.) Accept-step: Accept $\in \delta(q, u)$ : halt and accept

(6.) $\delta(q, u) = \emptyset$ : halt and reject

## Configurations




$: \triangleright a_1 \ldots a_m q b_1 b_2 \ldots b_k b_{k+1} \ldots \triangleleft$

Restart configuration: $q_0 \triangleright w \triangleleft \quad (w \in \Gamma^*)$
resulting from a restart step

Initial configuration: $q_0 \triangleright w \triangleleft \quad (w \in \Sigma^*)$

Accepting configuration: Accept

Rejecting configuration: $\triangleright xquy \triangleleft$, where $\delta(q, u) = \emptyset$

## Computations

Cycle:

$$q_0 \rhd w \lhd \underset{\text{MVR,MVL,Rewrite}}{\vdash^*} \rhd x q_1 uy \lhd \underset{\text{Restart}}{\vdash} q_0 \rhd xuy \lhd .$$

Each cycle contains exactly one application of a Rewrite step.

## Computations

Cycle:

$$q_0 \rhd w \lhd \underset{\text{MVR,MVL,Rewrite}}{\vdash^*} \rhd x q_1 uy \lhd \underset{\text{Restart}}{\vdash} q_0 \rhd xuy \lhd .$$

Each cycle contains exactly one application of a Rewrite step.

Tail:

$$q_0 \rhd w \lhd \underset{\text{MVR,MVL,Rewrite}}{\vdash^*} \rhd x q_1 uy \lhd \underset{\text{Accept}}{\vdash} \text{Accept}.$$

A tail contains at most one application of a Rewrite step.

## Computations

Cycle:

$$q_0 \rhd w \lhd \underset{\text{MVR,MVL,Rewrite}}{\vdash^*} \rhd x q_1 uy \lhd \underset{\text{Restart}}{\vdash} q_0 \rhd xuy \lhd .$$

Each cycle contains exactly one application of a Rewrite step.

Tail:

$$q_0 \rhd w \lhd \underset{\text{MVR,MVL,Rewrite}}{\vdash^*} \rhd x q_1 uy \lhd \underset{\text{Accept}}{\vdash} \text{Accept}.$$

A tail contains at most one application of a Rewrite step.

## Language Accepted by a Restarting Automaton

$w \in \Sigma^*$ is accepted by $M$, if $\exists$ computation $q_0 \rhd w \lhd \vdash_M^*$ Accept.

$L(M) := \{ w \in \Sigma^* \mid w \text{ is accepted by } M \}$ is the (input) language of $M$.

## Restricted Types of Restarting Automata

(a) Restrictions on the Move-operations:

- unrestricted move-operations MVL and MVR:
  Notation: RL-
- one-way restarting automata:
  no MVL-operations
  Notation: RR-
- restricted one-way restarting automata:
  no MVL-operations,
  rewrite- and restart-operations combined into a single operation.
  Notation: R-

## Restricted Tpyes of Restarting Automata (cont.)

(b) Restrictions on the Rewrite-operation:

- length-reducing restarting automata (standard model):
  For all $(q', v) \in \delta(q, u) : |u| > |v|$.
  Notation: -WW
- shrinking restarting automata:
  $\exists$ weight-function $\omega : \Gamma \cup \{\rhd, \lhd\} \to \mathbb{N}_+$:
  For all $(q', v) \in \delta(q, u) : \omega(u) > \omega(v)$.
  Notation: s- -WW
- restarting automata without aux. symbols: $\Gamma = \Sigma$
  Notation: -W
- deleting restarting automata:
  For all $(q', v) \in \delta(q, u) : v$ is a scattered subword of $u$.
  Notation: -$\varepsilon$

# 3. sRWW-Automata Versus sRLWW-Automata

**Theorem 1 (Jurdziński, O. 2007)**

$\mathcal{L}(\text{sRLWW}) = \mathcal{L}(\text{sRRWW}) = \mathcal{L}(\text{sRWW})$.

**Proof outline:**

Each (s)RL(WW)-automaton is simulated by a (s)RR(WW)-automaton [Plátek 2001], and each sRRWW-automaton is simulated by an sRWW-automaton [Jurd., O. 2007].

□

# 3. sRWW-Automata Versus sRLWW-Automata

**Theorem 1 (Jurdziński, O. 2007)**

$\mathcal{L}(\text{sRLWW}) = \mathcal{L}(\text{sRRWW}) = \mathcal{L}(\text{sRWW})$.

**Proof outline:**

Each (s)RL(WW)-automaton is simulated by a (s)RR(WW)-automaton [Plátek 2001], and each sRRWW-automaton is simulated by an sRWW-automaton [Jurd., O. 2007].

□

Obviously, we have the chain of inclusions

$$\mathcal{L}(\text{RWW}) \subseteq \mathcal{L}(\text{RRWW}) = \mathcal{L}(\text{RLWW}) \subseteq \mathcal{L}(\text{sRLWW}) = \mathcal{L}(\text{sRWW}),$$

but it is still open whether any of the inclusions in this chain is proper.

## Theorem 2 (Jurdziński, O. 2007)

*If $M$ is an sR(R)WW-automaton, and if $\omega$ is a weight function compatible with $M$, then $r_\omega(L(M))$ is accepted by some R(R)WW-automaton.*

Here $r_\omega$ is the morphism defined by $r_\omega(a) := a\diamondsuit^{\omega(a)-1}$ $(a \in \Gamma)$, where $\diamondsuit$ and $\triangle$ are two new symbols.
Thus, for each word $u \in \Gamma^*$, $|r_\omega(u)| = \omega(u)$.

## Theorem 2 (Jurdziński, O. 2007)

*If $M$ is an sR(R)WW-automaton, and if $\omega$ is a weight function compatible with $M$, then $r_\omega(L(M))$ is accepted by some R(R)WW-automaton.*

Here $r_\omega$ is the morphism defined by $r_\omega(a) := a\diamondsuit^{\omega(a)-1}$ $(a \in \Gamma)$, where $\diamondsuit$ and $\triangle$ are two new symbols.
Thus, for each word $u \in \Gamma^*$, $|r_\omega(u)| = \omega(u)$.

## Theorem 3 (Jurdziński, O. 2007)

*For each language $L \in \mathcal{L}(\mathrm{RRWW})$, $r_{46}(L) \in \mathcal{L}(\mathrm{RWW})$.*

Here $r_{46}$ is defined by $r_{46}(a) := a\triangle^{45}$ for all $a \in \Gamma$.
Thus, for each word $u \in \Gamma^*$, $|r_{46}(u)| = 46 \cdot |u|$.

## Lemma 4 (Jančar et al. 1999)

$L_1 = \{ a^n b^n c, a^n b^{2n} d \mid n \geq 0 \} \in \mathcal{L}(\mathrm{RR}) \setminus \mathcal{L}(\mathrm{sRW})$.

## Lemma 4 (Jančar et al. 1999)

$L_1 = \{ a^n b^n c, a^n b^{2n} d \mid n \geq 0 \} \in \mathcal{L}(\text{RR}) \setminus \mathcal{L}(\text{sRW})$.

Let $L_{\text{exp}} = L^{(1)} \cup L^{(2)} \cup L^{(3)}$, where

$$\begin{aligned} L^{(1)} &= \{ a^{2^n} \mid n \geq 0 \}, \\ L^{(2)} &= \{ a^m (bc)^n \mid \exists k : m + 2n = 2^k \}, \text{ and} \\ L^{(3)} &= \{ (bc)^m a^n \mid \exists k : m + n = 2^k \}. \end{aligned}$$

For the language $L_{\text{exp}}$, we have the following new result.

## Lemma 5

$L_{\text{exp}} \in \mathcal{L}(\text{det-sRW}) \setminus \mathcal{L}(\text{RRW})$.

## Proof outline:

Given a word $w \in \Sigma^*$ as input, $M_{\exp}$ scans $w$ from left to right and proceeds as follows.

1. If $w = a$, then $M_{\exp}$ accepts, as $a \in L^{(1)}$.
2. If $w = a^n$ for some $n \geq 2$, then $M_{\exp}$ simply rewrites the suffix $aa\triangleleft$ of the tape contents into $bc\triangleleft$. In order to ensure that this rewrite step is weight-reducing, we choose the weight function $\omega(a) = 3$ and $\omega(b) = \omega(c) = 2$.
3. If $w = a^m(bc)w'$, then $M_{\exp}$ rewrites the factor $aabc$ into $bcbc$.
4. If $w = (bc)^n$, then $M_{\exp}$ rewrites the suffix $bc\triangleleft$ of the tape contents into $a\triangleleft$.
5. Finally, if $w = (bc)^n aw'$, then $M_{\exp}$ rewrites the factor $bca$ into $aa$.

Then $M_{\exp}$ is a deterministic sRW-automaton such that $L(M_{\exp}) = L_{\exp}$.

## Proof outline (cont.):

Assume that $M$ is an RRW-automaton on $\Sigma$ such that $L(M) = L_{\exp}$. Given an input of the form $w = a^{2^n}$ for some large integer $n > k$, $M$ cannot accept in a tail computation because of a pumping argument.

## Proof outline (cont.):

Assume that $M$ is an RRW-automaton on $\Sigma$ such that $L(M) = L_{\exp}$. Given an input of the form $w = a^{2^n}$ for some large integer $n > k$, $M$ cannot accept in a tail computation because of a pumping argument. Thus, an accepting computation of $M$ on $w$ begins with a cycle that rewrites $w = a^{2^n}$ into a word $w' \in L_{\exp}$ such that $2^n - k \leq |w'| \leq 2^n - 1$.

## Proof outline (cont.):

Assume that $M$ is an RRW-automaton on $\Sigma$ such that $L(M) = L_{\exp}$. Given an input of the form $w = a^{2^n}$ for some large integer $n > k$, $M$ cannot accept in a tail computation because of a pumping argument. Thus, an accepting computation of $M$ on $w$ begins with a cycle that rewrites $w = a^{2^n}$ into a word $w' \in L_{\exp}$ such that $2^n - k \leq |w'| \leq 2^n - 1$. As $|w'| \leq 2^n - 1$, it can be neither of the form $a^{2^m}$ nor of the form $a^m(bc)^l$ such that $m + 2l$ is a power of two, since in these cases the difference $|w| - |w'|$ would be at least $2^{n-1} > k$.

## Proof outline (cont.):

Assume that $M$ is an RRW-automaton on $\Sigma$ such that $L(M) = L_{\exp}$. Given an input of the form $w = a^{2^n}$ for some large integer $n > k$, $M$ cannot accept in a tail computation because of a pumping argument. Thus, an accepting computation of $M$ on $w$ begins with a cycle that rewrites $w = a^{2^n}$ into a word $w' \in L_{\exp}$ such that $2^n - k \leq |w'| \leq 2^n - 1$. As $|w'| \leq 2^n - 1$, it can be neither of the form $a^{2^m}$ nor of the form $a^m(bc)^l$ such that $m + 2l$ is a power of two, since in these cases the difference $|w| - |w'|$ would be at least $2^{n-1} > k$.
Hence, $w' \notin L^{(1)}$ and $w' \notin L^{(2)}$, which implies that $w' \in L^{(3)}$, that is, $w' = (bc)^m a^l$ such that $2^n - k \leq 2m + l \leq 2^n - 1$.

## Proof outline (cont.):

Assume that $M$ is an RRW-automaton on $\Sigma$ such that $L(M) = L_{\exp}$.
Given an input of the form $w = a^{2^n}$ for some large integer $n > k$, $M$ cannot accept in a tail computation because of a pumping argument.
Thus, an accepting computation of $M$ on $w$ begins with a cycle that rewrites $w = a^{2^n}$ into a word $w' \in L_{\exp}$ such that $2^n - k \leq |w'| \leq 2^n - 1$.
As $|w'| \leq 2^n - 1$, it can be neither of the form $a^{2^m}$ nor of the form $a^m(bc)^l$ such that $m + 2l$ is a power of two, since in these cases the difference $|w| - |w'|$ would be at least $2^{n-1} > k$.
Hence, $w' \notin L^{(1)}$ and $w' \notin L^{(2)}$, which implies that $w' \in L^{(3)}$, that is, $w' = (bc)^m a^l$ such that $2^n - k \leq 2m + l \leq 2^n - 1$.
It follows that $m + l = 2^r$ for some $r \leq n - 1$.

## Proof outline (cont.):

Assume that $M$ is an RRW-automaton on $\Sigma$ such that $L(M) = L_{\exp}$.
Given an input of the form $w = a^{2^n}$ for some large integer $n > k$, $M$ cannot accept in a tail computation because of a pumping argument.
Thus, an accepting computation of $M$ on $w$ begins with a cycle that rewrites $w = a^{2^n}$ into a word $w' \in L_{\exp}$ such that $2^n - k \leq |w'| \leq 2^n - 1$.
As $|w'| \leq 2^n - 1$, it can be neither of the form $a^{2^m}$ nor of the form $a^m(bc)^l$ such that $m + 2l$ is a power of two, since in these cases the difference $|w| - |w'|$ would be at least $2^{n-1} > k$.
Hence, $w' \notin L^{(1)}$ and $w' \notin L^{(2)}$, which implies that $w' \in L^{(3)}$, that is, $w' = (bc)^m a^l$ such that $2^n - k \leq 2m + l \leq 2^n - 1$.
It follows that $m + l = 2^r$ for some $r \leq n - 1$.
Thus, from $2^n - k \leq 2m + l = m + 2^r \leq 2^n - 1$ we obtain that $2^{n-1} - k \leq m + 2^r - 2^{n-1} \leq m$.

## Proof outline (cont.):

Assume that $M$ is an RRW-automaton on $\Sigma$ such that $L(M) = L_{\exp}$.
Given an input of the form $w = a^{2^n}$ for some large integer $n > k$, $M$ cannot accept in a tail computation because of a pumping argument.
Thus, an accepting computation of $M$ on $w$ begins with a cycle that rewrites $w = a^{2^n}$ into a word $w' \in L_{\exp}$ such that $2^n - k \leq |w'| \leq 2^n - 1$.
As $|w'| \leq 2^n - 1$, it can be neither of the form $a^{2^m}$ nor of the form $a^m(bc)^l$ such that $m + 2l$ is a power of two, since in these cases the difference $|w| - |w'|$ would be at least $2^{n-1} > k$.
Hence, $w' \notin L^{(1)}$ and $w' \notin L^{(2)}$, which implies that $w' \in L^{(3)}$, that is, $w' = (bc)^m a^l$ such that $2^n - k \leq 2m + l \leq 2^n - 1$.
It follows that $m + l = 2^r$ for some $r \leq n - 1$.
Thus, from $2^n - k \leq 2m + l = m + 2^r \leq 2^n - 1$ we obtain that $2^{n-1} - k \leq m + 2^r - 2^{n-1} \leq m$.
This means that the prefix $a^{2^n - 2k}$ of $w$ and the prefix $(bc)^{2^{n-1}-k}$ of $w'$ of length $2^n - 2k$ differ at each position, a contradiction.

## Proof outline (cont.):

Assume that $M$ is an RRW-automaton on $\Sigma$ such that $L(M) = L_{\exp}$.
Given an input of the form $w = a^{2^n}$ for some large integer $n > k$, $M$ cannot accept in a tail computation because of a pumping argument.
Thus, an accepting computation of $M$ on $w$ begins with a cycle that rewrites $w = a^{2^n}$ into a word $w' \in L_{\exp}$ such that $2^n - k \leq |w'| \leq 2^n - 1$.
As $|w'| \leq 2^n - 1$, it can be neither of the form $a^{2^m}$ nor of the form $a^m(bc)^l$ such that $m + 2l$ is a power of two, since in these cases the difference $|w| - |w'|$ would be at least $2^{n-1} > k$.
Hence, $w' \notin L^{(1)}$ and $w' \notin L^{(2)}$, which implies that $w' \in L^{(3)}$, that is, $w' = (bc)^m a^l$ such that $2^n - k \leq 2m + l \leq 2^n - 1$.
It follows that $m + l = 2^r$ for some $r \leq n - 1$.
Thus, from $2^n - k \leq 2m + l = m + 2^r \leq 2^n - 1$ we obtain that $2^{n-1} - k \leq m + 2^r - 2^{n-1} \leq m$.
This means that the prefix $a^{2^n - 2k}$ of $w$ and the prefix $(bc)^{2^{n-1}-k}$ of $w'$ of length $2^n - 2k$ differ at each position, a contradiction.
It follows that $L_{\exp}$ is not accepted by any RRW-automaton. □

## Theorem 6

(a) $\mathcal{L}(\mathsf{RLW}) = \mathcal{L}(\mathsf{RRW}) \subsetneq \mathcal{L}(\mathsf{sRRW}) = \mathcal{L}(\mathsf{sRLW})$.

(b) $\mathcal{L}(\mathsf{RW}) \subsetneq \mathcal{L}(\mathsf{sRW}) \subsetneq \mathcal{L}(\mathsf{sRRW}) = \mathcal{L}(\mathsf{sRLW})$.

## Theorem 6

(a) $\mathcal{L}(\mathsf{RLW}) = \mathcal{L}(\mathsf{RRW}) \subsetneq \mathcal{L}(\mathsf{sRRW}) = \mathcal{L}(\mathsf{sRLW})$.

(b) $\mathcal{L}(\mathsf{RW}) \subsetneq \mathcal{L}(\mathsf{sRW}) \subsetneq \mathcal{L}(\mathsf{sRRW}) = \mathcal{L}(\mathsf{sRLW})$.

## Proof outline:

From Lemma 4 we see that $\mathcal{L}(\mathsf{sRW})$ is a proper subclass of $\mathcal{L}(\mathsf{sRRW})$.

## Theorem 6

(a) $\mathcal{L}(\mathrm{RLW}) = \mathcal{L}(\mathrm{RRW}) \subsetneq \mathcal{L}(\mathrm{sRRW}) = \mathcal{L}(\mathrm{sRLW})$.

(b) $\mathcal{L}(\mathrm{RW}) \subsetneq \mathcal{L}(\mathrm{sRW}) \subsetneq \mathcal{L}(\mathrm{sRRW}) = \mathcal{L}(\mathrm{sRLW})$.

## Proof outline:

From Lemma 4 we see that $\mathcal{L}(\mathrm{sRW})$ is a proper subclass of $\mathcal{L}(\mathrm{sRRW})$. From Lemma 5 we see that $L_{\mathrm{exp}}$ is accepted by an sRW-, and therewith also by an sRRW-automaton. As it is not accepted by any RRW-automaton, we see that the inclusions $\mathcal{L}(\mathrm{RW}) \subseteq \mathcal{L}(\mathrm{sRW})$ and $\mathcal{L}(\mathrm{RRW}) \subseteq \mathcal{L}(\mathrm{sRRW})$ are also proper. □

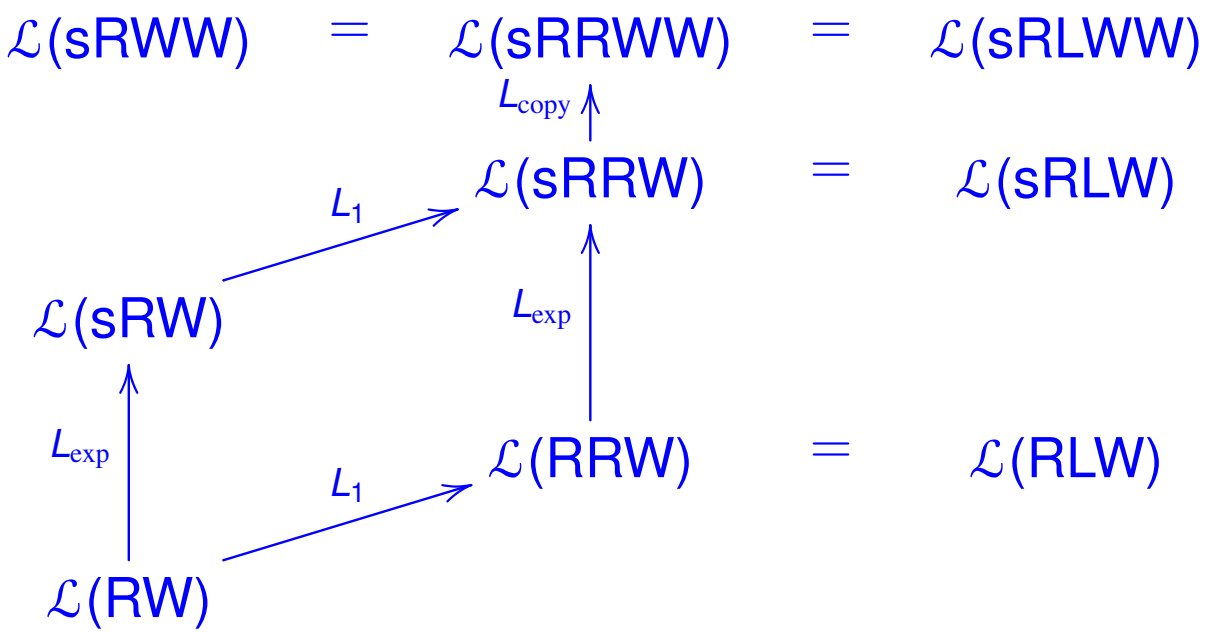



Figure: The taxonomy of nondeterministic shrinking restarting automata.

# 3. Deterministic sRWW- and sRRWW-Automata

CRL: Church-Rosser languages [McNaughton et al. 1988],

GCSL: growing context-sensitive lang. [Dahlhaus, Warmuth 1986].

CRL is the deterministic variant of GCSL [Niemann, O. 2005].

# 3. Deterministic sRWW- and sRRWW-Automata

CRL: Church-Rosser languages [McNaughton et al. 1988],

GCSL: growing context-sensitive lang. [Dahlhaus, Warmuth 1986].

CRL is the deterministic variant of GCSL [Niemann, O. 2005].

**Theorem 7 (Niemann, O. 2003, Jurdziński, O. 2007)**

$\mathrm{CRL} = \mathcal{L}(\text{det-R(R)WW}) = \mathcal{L}(\text{det-sR(R)WW}) \subsetneq \mathcal{L}(\text{det-RLWW}).$

## Theorem 8

$\mathcal{L}(\text{det-RL}) \not\subset \text{GCSL}$.

## Theorem 8

$\mathcal{L}(\text{det-RL}) \not\subset \text{GCSL}$.

## Proof outline:

Let $M$ be a det-RLWW-automaton for $L_{\text{copy}} = \{ w\#w \mid w \in \{a, b\}^* \}$ with tape alphabet $\Gamma$.

## Theorem 8

$\mathcal{L}(\text{det-RL}) \not\subset \text{GCSL}$.

## Proof outline:

Let $M$ be a det-RLWW-automaton for $L_{\text{copy}} = \{ w\#w \mid w \in \{a, b\}^* \}$ with tape alphabet $\Gamma$.
By declaring each element of $\Gamma$ to be an input symbol, we obtain a det-RLW-automaton $M_1$ such that $L(M_1) \cap \{a, b, \#\}^* = L_{\text{copy}}$.

## Theorem 8

$\mathcal{L}(\text{det-RL}) \not\subset \text{GCSL}$.

## Proof outline:

Let $M$ be a det-RLWW-automaton for $L_{\text{copy}} = \{ w\#w \mid w \in \{a,b\}^* \}$ with tape alphabet $\Gamma$.
By declaring each element of $\Gamma$ to be an input symbol, we obtain a det-RLW-automaton $M_1$ such that $L(M_1) \cap \{a,b,\#\}^* = L_{\text{copy}}$.
There exists an encoding $\varphi : \Gamma \to \{c,d,0,1\}^r$ such that $\tilde{L} := \varphi(L(M_1))$ is accepted by some det-RL-automaton [Jurdziński et al. 2004].

## Theorem 8

$\mathcal{L}(\text{det-RL}) \not\subset \text{GCSL}$.

## Proof outline:

Let $M$ be a det-RLWW-automaton for $L_{\text{copy}} = \{ w\#w \mid w \in \{a,b\}^* \}$ with tape alphabet $\Gamma$.
By declaring each element of $\Gamma$ to be an input symbol, we obtain a det-RLW-automaton $M_1$ such that $L(M_1) \cap \{a,b,\#\}^* = L_{\text{copy}}$.
There exists an encoding $\varphi : \Gamma \to \{c,d,0,1\}^r$ such that $\tilde{L} := \varphi(L(M_1))$ is accepted by some det-RL-automaton [Jurdziński et al. 2004].
As GCSL is closed under inverse morphisms and intersection with REG, it follows that with $\tilde{L}$, also $L_{\text{copy}}$ is growing context-sensitive.

## Theorem 8

$\mathcal{L}(\text{det-RL}) \not\subset \text{GCSL}$.

## Proof outline:

Let $M$ be a det-RLWW-automaton for $L_{\text{copy}} = \{ w\#w \mid w \in \{a,b\}^* \}$ with tape alphabet $\Gamma$.
By declaring each element of $\Gamma$ to be an input symbol, we obtain a det-RLW-automaton $M_1$ such that $L(M_1) \cap \{a,b,\#\}^* = L_{\text{copy}}$.
There exists an encoding $\varphi : \Gamma \to \{c,d,0,1\}^r$ such that $\tilde{L} := \varphi(L(M_1))$ is accepted by some det-RL-automaton [Jurdziński et al. 2004].
As GCSL is closed under inverse morphisms and intersection with REG, it follows that with $\tilde{L}$, also $L_{\text{copy}}$ is growing context-sensitive.
As this is not the case, it follows that $\tilde{L} \notin \text{GCSL}$.
Thus, the class $\mathcal{L}(\text{det-RL})$ is not contained in GCSL. □

## Theorem 8

$\mathcal{L}(\text{det-RL}) \not\subset \text{GCSL}$.

## Proof outline:

Let $M$ be a det-RLWW-automaton for $L_{\text{copy}} = \{ w\#w \mid w \in \{a,b\}^* \}$ with tape alphabet $\Gamma$.
By declaring each element of $\Gamma$ to be an input symbol, we obtain a det-RLW-automaton $M_1$ such that $L(M_1) \cap \{a,b,\#\}^* = L_{\text{copy}}$.
There exists an encoding $\varphi : \Gamma \to \{c,d,0,1\}^r$ such that $\tilde{L} := \varphi(L(M_1))$ is accepted by some det-RL-automaton [Jurdziński et al. 2004].
As GCSL is closed under inverse morphisms and intersection with REG, it follows that with $\tilde{L}$, also $L_{\text{copy}}$ is growing context-sensitive.
As this is not the case, it follows that $\tilde{L} \notin \text{GCSL}$.
Thus, the class $\mathcal{L}(\text{det-RL})$ is not contained in GCSL. □

None of the classes $\mathcal{L}(\text{det-(s)RL(W)(W)})$ is contained in GCSL.

## Theorem 9

$\mathcal{L}(\text{det-RLW}) \subsetneq \mathcal{L}(\text{det-RLWW})$ *and* $\mathcal{L}(\text{det-sRLW}) \subsetneq \mathcal{L}(\text{det-sRLWW})$.

## Theorem 9

$\mathcal{L}(\text{det-RLW}) \subsetneq \mathcal{L}(\text{det-RLWW})$ *and* $\mathcal{L}(\text{det-sRLW}) \subsetneq \mathcal{L}(\text{det-sRLWW})$.

## Theorem 10

(a) $\mathcal{L}(\text{det-RL}) \subsetneq \mathcal{L}(\text{det-RLW})$.
(b) $\mathcal{L}(\text{det-RLW}) \subsetneq \mathcal{L}(\text{det-sRLW})$.
(c) $\text{CRL} \not\subset \mathcal{L}(\text{det-sRLW})$.

## Theorem 9

$\mathcal{L}(\text{det-RLW}) \subsetneq \mathcal{L}(\text{det-RLWW})$ *and* $\mathcal{L}(\text{det-sRLW}) \subsetneq \mathcal{L}(\text{det-sRLWW})$.

## Theorem 10

(a) $\mathcal{L}(\text{det-RL}) \subsetneq \mathcal{L}(\text{det-RLW})$.
(b) $\mathcal{L}(\text{det-RLW}) \subsetneq \mathcal{L}(\text{det-sRLW})$.
(c) $\text{CRL} \not\subset \mathcal{L}(\text{det-sRLW})$.

## Proof outline:

(a) $L_a = \{ a^n b^{n+i} c^{2n-2i} \mid n > 0,\ n \geq i \geq 0 \}$ is accepted by a det-RLW-automaton $M$, but $L_a$ is not accepted by any det-RL-automaton.

## Proof outline (cont.):

(b) $L_b = \{ a^n b^{n+i} c^{n-i} \mid n > 0, n \geq i \geq 0 \}$ is accepted by a det-sRLW-automaton $M$, but $L_b$ is not accepted by any det-RLW-automaton.

## Proof outline (cont.):

(b) $L_b = \{ a^n b^{n+i} c^{n-i} \mid n > 0, n \geq i \geq 0 \}$ is accepted by a det-sRLW-automaton $M$, but $L_b$ is not accepted by any det-RLW-automaton.

(c) $L_{\text{expo}} = \{ a^{2^n} \mid n \geq 0 \}$ is a Church-Rosser language, but it is not accepted by any det-sRLW-automaton. □

## Proof outline (cont.):

(b) $L_b = \{ a^n b^{n+i} c^{n-i} \mid n > 0, n \geq i \geq 0 \}$ is accepted by a det-sRLW-automaton $M$, but $L_b$ is not accepted by any det-RLW-automaton.

(c) $L_{\text{expo}} = \{ a^{2^n} \mid n \geq 0 \}$ is a Church-Rosser language, but it is not accepted by any det-sRLW-automaton. □

## Corollary 11

$\mathcal{L}(\text{det-RW}) \subsetneq \mathcal{L}(\text{det-sRW})$ *and* $\mathcal{L}(\text{det-RRW}) \subsetneq \mathcal{L}(\text{det-sRRW})$.

## Proof outline (cont.):

(b) $L_b = \{ a^n b^{n+i} c^{n-i} \mid n > 0, n \geq i \geq 0 \}$ is accepted by a det-sRLW-automaton $M$, but $L_b$ is not accepted by any det-RLW-automaton.

(c) $L_{\text{expo}} = \{ a^{2^n} \mid n \geq 0 \}$ is a Church-Rosser language, but it is not accepted by any det-sRLW-automaton. □

## Corollary 11

$\mathcal{L}(\text{det-RW}) \subsetneq \mathcal{L}(\text{det-sRW})$ *and* $\mathcal{L}(\text{det-RRW}) \subsetneq \mathcal{L}(\text{det-sRRW})$.

## Corollary 12

$\mathcal{L}(\text{det-sRW}) \subsetneq \mathcal{L}(\text{det-sRRW})$.

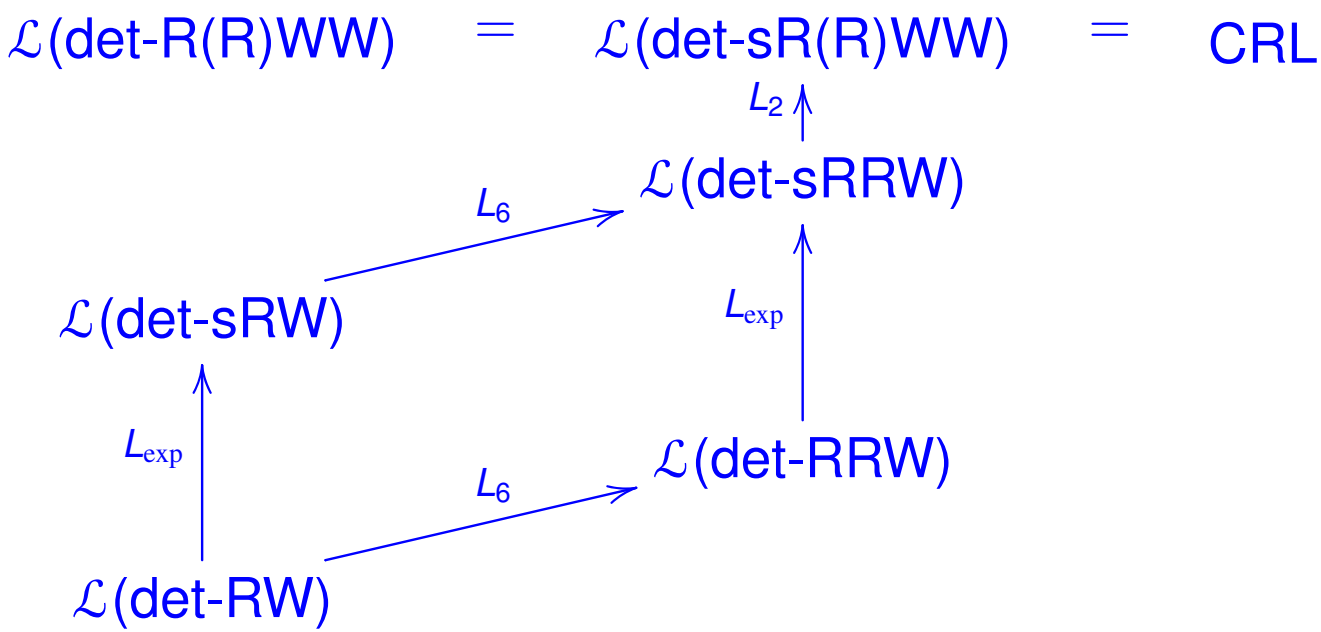



Figure: The taxonomy of deterministic shrinking restarting automata.

Each cycle $C$ of a computation of a restarting automaton contains a unique configuration $\alpha q \beta$ in which a rewrite instruction is applied. Then $D_l(C) = |\alpha|$ is the left distance of $C$ and $D_r(C) = |\beta|$ is the right distance of $C$.

Each cycle $C$ of a computation of a restarting automaton contains a unique configuration $\alpha q \beta$ in which a rewrite instruction is applied.
Then $D_l(C) = |\alpha|$ is the left distance of $C$
and $D_r(C) = |\beta|$ is the right distance of $C$.
A sequence of cycles $S = (C_1, C_2, \ldots, C_n)$ is
(right-) monotone if $D_r(C_1) \geq D_r(C_2) \geq \cdots \geq D_r(C_n)$, and
it is left-monotone if $D_l(C_1) \geq D_l(C_2) \geq \cdots \geq D_l(C_n)$.

Each cycle $C$ of a computation of a restarting automaton contains a unique configuration $\alpha q \beta$ in which a rewrite instruction is applied.
Then $D_l(C) = |\alpha|$ is the left distance of $C$
and $D_r(C) = |\beta|$ is the right distance of $C$.
A sequence of cycles $S = (C_1, C_2, \ldots, C_n)$ is
(right-) monotone if $D_r(C_1) \geq D_r(C_2) \geq \cdots \geq D_r(C_n)$, and
it is left-monotone if $D_l(C_1) \geq D_l(C_2) \geq \cdots \geq D_l(C_n)$.
A computation is (right-) monotone (left-monotone), if the corresponding sequence of cycles is (right-) monotone (left-monotone).
Finally, a restarting automaton is called (right-) monotone (left-monotone), if all its computations that start with an initial configuration are (right-) monotone (left-monotone).

## Theorem 13 (O. 2006)

$\mathcal{L}(\text{det-mon-RL}) \subsetneq \text{CRL}$.

## Theorem 13 (O. 2006)

$\mathcal{L}(\text{det-mon-RL}) \subsetneq \text{CRL}$.

Theorem 8 implies that the inclusion $\mathcal{L}(\text{det-mon-RL}) \subset \mathcal{L}(\text{det-RL})$ is proper. Thus, concerning the various classes $\mathcal{L}(\text{det-(s)RL(W)(W)})$ only the following questions remain open:

(1) Is CFL contained in $\mathcal{L}(\text{det-RLWW})$ or at least in $\mathcal{L}(\text{det-sRLWW})$?

(2) Is $\mathcal{L}(\text{det-sRLW})$ contained in $\mathcal{L}(\text{det-RLWW})$?

(3) Is the inclusion $\mathcal{L}(\text{det-RLWW}) \subseteq \mathcal{L}(\text{det-sRLWW})$ proper?

## Theorem 13 (O. 2006)

$\mathcal{L}(\text{det-mon-RL}) \subsetneq \text{CRL}$.

Theorem 8 implies that the inclusion $\mathcal{L}(\text{det-mon-RL}) \subset \mathcal{L}(\text{det-RL})$ is proper. Thus, concerning the various classes $\mathcal{L}(\text{det-(s)RL(W)(W)})$ only the following questions remain open:

(1) Is CFL contained in $\mathcal{L}(\text{det-RLWW})$ or at least in $\mathcal{L}(\text{det-sRLWW})$?

(2) Is $\mathcal{L}(\text{det-sRLW})$ contained in $\mathcal{L}(\text{det-RLWW})$?

(3) Is the inclusion $\mathcal{L}(\text{det-RLWW}) \subseteq \mathcal{L}(\text{det-sRLWW})$ proper?

## Remark 14

$\mathcal{L}(\text{det-sRLW}) \subseteq \mathcal{L}(\text{det-RLWW})$ *if and only if* $\mathcal{L}(\text{det-RLWW}) = \mathcal{L}(\text{det-sRLWW})$.

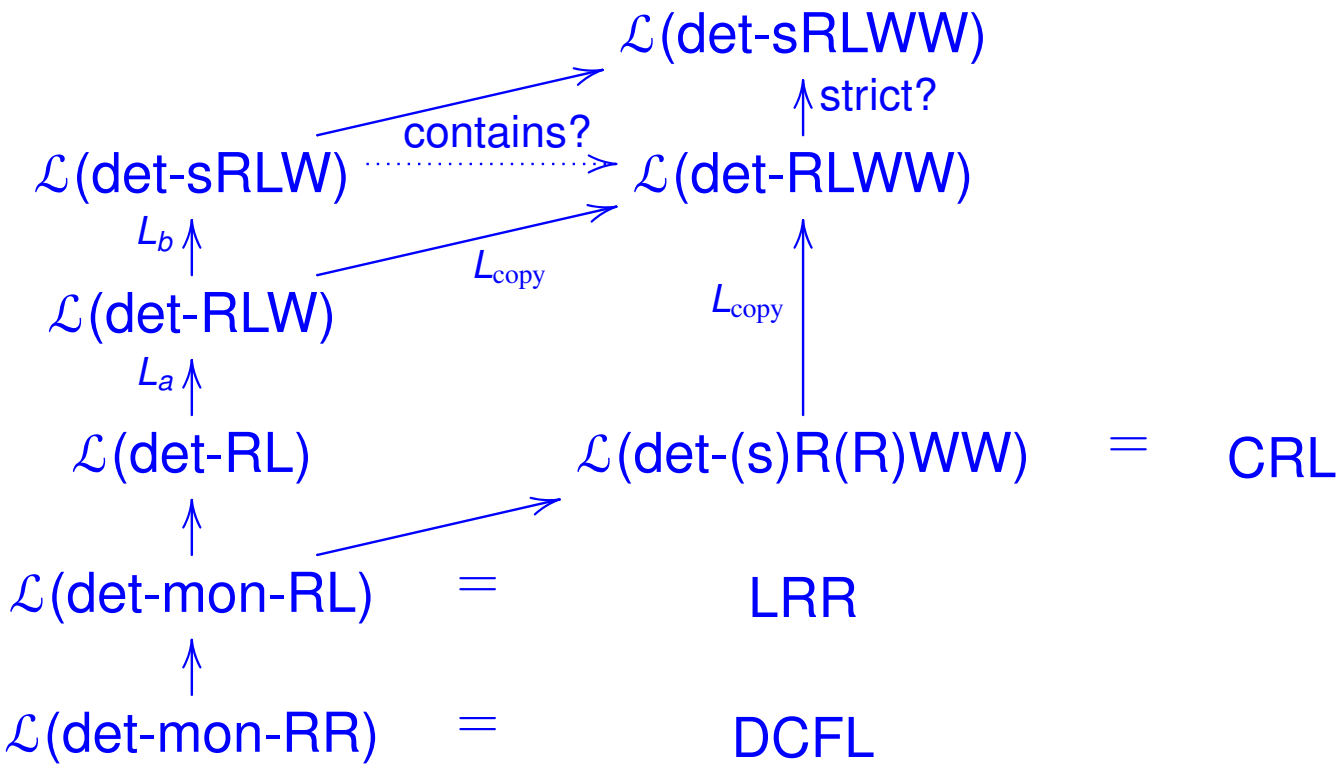



Figure: The taxonomy of deterministic RL-automata.

LRR: left-to-right regular languages of [Čulik, Cohen 1973].

# 4. Monotone Shrinking Restarting Automata

**Theorem 15 (Jurdziński et al. 2004)**

$$\begin{aligned} \text{GCSL} &= \mathcal{L}(\text{wmon-RWW}) = \mathcal{L}(\text{wmon-sRWW}) \\ &= \mathcal{L}(\text{wmon-RRWW}) = \mathcal{L}(\text{wmon-sRRWW}) \\ &= \mathcal{L}(\text{wmon-RLWW}) = \mathcal{L}(\text{wmon-sRLWW}). \end{aligned}$$

# 4. Monotone Shrinking Restarting Automata

## Theorem 15 (Jurdziński et al. 2004)

$$\begin{aligned} \text{GCSL} &= \mathcal{L}(\text{wmon-RWW}) = \mathcal{L}(\text{wmon-sRWW}) \\ &= \mathcal{L}(\text{wmon-RRWW}) = \mathcal{L}(\text{wmon-sRRWW}) \\ &= \mathcal{L}(\text{wmon-RLWW}) = \mathcal{L}(\text{wmon-sRLWW}). \end{aligned}$$

## Theorem 16 (Jančar et al. 1999, Jurdziński et al. 2006)

$$\begin{aligned} \text{CFL} &= \mathcal{L}(\text{mon-sRLWW}) = \mathcal{L}(\text{left-mon-sRLWW}) \\ &= \mathcal{L}(\text{mon-sRRWW}) = \mathcal{L}(\text{left-mon-sRRWW}) \\ &= \mathcal{L}(\text{mon-sRWW}) = \mathcal{L}(\text{left-mon-sRWW}) \\ &= \mathcal{L}(\text{mon-RLWW}) = \mathcal{L}(\text{left-mon-RLWW}) \\ &= \mathcal{L}(\text{mon-RRWW}) = \mathcal{L}(\text{left-mon-RRWW}) \\ &= \mathcal{L}(\text{mon-RWW}) = \mathcal{L}(\text{left-mon-RWW}). \end{aligned}$$

**Theorem 17 (Jančar et al. 1999, Jurdziński et al. 2004)**

$$\begin{array}{lcccc}
\text{DCFL} & = & \mathcal{L}(\text{det-mon-sRRWW}) & = & \mathcal{L}(\text{det-mon-RRWW}) \\
& = & \mathcal{L}(\text{det-mon-sRWW}) & = & \mathcal{L}(\text{det-mon-RWW}) \\
& = & \mathcal{L}(\text{det-mon-sRRW}) & = & \mathcal{L}(\text{det-mon-RRW}) \\
& = & \mathcal{L}(\text{det-mon-sRW}) & = & \mathcal{L}(\text{det-mon-RW}) \\
& = & \mathcal{L}(\text{det-mon-RR}) & = & \mathcal{L}(\text{det-mon-R}).
\end{array}$$

## Theorem 17 (Jančar et al. 1999, Jurdziński et al. 2004)

$$\begin{array}{rcccc} \mathrm{DCFL} & = & \mathcal{L}(\text{det-mon-sRRWW}) & = & \mathcal{L}(\text{det-mon-RRWW}) \\ & = & \mathcal{L}(\text{det-mon-sRWW}) & = & \mathcal{L}(\text{det-mon-RWW}) \\ & = & \mathcal{L}(\text{det-mon-sRRW}) & = & \mathcal{L}(\text{det-mon-RRW}) \\ & = & \mathcal{L}(\text{det-mon-sRW}) & = & \mathcal{L}(\text{det-mon-RW}) \\ & = & \mathcal{L}(\text{det-mon-RR}) & = & \mathcal{L}(\text{det-mon-R}). \end{array}$$

## Theorem 18 (Jurdziński et al. 2004, O. 2009, Plátek 2001)

$$\begin{array}{rcccc} \mathrm{LRR} & = & \mathcal{L}(\text{det-mon-RL}) & = & \mathcal{L}(\text{det-mon-RLW}) \\ & = & \mathcal{L}(\text{det-mon-sRLW}) & = & \mathcal{L}(\text{det-mon-RLWW}) \\ & = & \mathcal{L}(\text{det-mon-sRLWW}) & \supsetneq & \mathrm{DCFL}. \end{array}$$

## Theorem 19 (Jurdziński et al. 2006)

$$
\begin{aligned}
\mathcal{L}(\text{det-left-mon-RL}) &= \mathcal{L}(\text{det-left-mon-RLW}) \\
&= \mathcal{L}(\text{det-left-mon-sRLW}) \\
&= \mathcal{L}(\text{det-left-mon-RLWW}) \\
&= \mathcal{L}(\text{det-left-mon-sRLWW}) \\
&= \mathcal{L}(\text{det-left-mon-RRWW}) \\
&= \mathcal{L}(\text{det-left-mon-sRRWW}) \\
&= \mathcal{L}(\text{det-left-mon-RWW}) \\
&= \mathcal{L}(\text{det-left-mon-sRWW}) \\
&\supsetneq \text{DCFL}^R.
\end{aligned}
$$

# Thank you for your attention!