On Shrinking Restarting Automata of Window Size One and Two

František Mráz[1], Friedrich Otto[2]

[1]Charles University, Prague, Czech Republic

[2]Universität Kassel, Kassel, Germany

DLT 2019

Warsaw

August 7, 2019

Contents

1 Introduction

2 Definitions

3 Monotone Shrinking Restarting Automata

4 Non-Monotone Shrinking Restarting Automata

5 Conclusion

Restarting Automaton (1)

RRWW-automaton

- motivated from linguistics – analysis by reduction
- a flexible tape – a word delimited by sentinels ¢ and $
- a finite-state control with a read/write window of a fixed size
- a finite input alphabet Σ
- a finite working alphabet Γ ($\supseteq \Sigma$)
- operations: move right, rewrite by a shorter string, accept (reject), restart
- each computation in phases
  - a cycle: start on the left end, move right, rewrite, move right, restart
  - a tail: start on the left end, move right, halt
    - Accept operation
    - reject: no step possible

Restarting Automaton (2)

- A restricted model: RWW-automaton: an RRWW-automaton that always restarts immediately after each rewrite operation
  - **Open Problem:** $\mathcal{L}(\text{RWW}) \stackrel{?}{=} \mathcal{L}(\text{RRWW})$.
- An extended model: shrinking RRWW-automaton (sRRWW-automaton for short)
  - rewrite need not to be length-reducing, it must be weight-reducing
  - [Jurdziński, Otto, '07] $\mathcal{L}(\text{sRWW}) = \mathcal{L}(\text{sRRWW})$.

Restarting Automaton

RRWW-automaton

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

- finite state control, set of states Q,

Restarting Automaton

RRWW-automaton

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

- finite state control, set of states Q,
- q_0 is the initial state,

Restarting Automaton

RRWW-automaton

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

- finite state control, set of states Q,
- q_0 is the initial state,
- input alphabet Σ,

Restarting Automaton

RRWW-automaton

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

- finite state control, set of states Q,
- q_0 is the initial state,
- input alphabet Σ,
- working alphabet Γ, $\Sigma \subseteq \Gamma$,

Restarting Automaton

RRWW-automaton

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

- finite state control, set of states Q,
- q_0 is the initial state,
- input alphabet Σ,
- working alphabet Γ, $\Sigma \subseteq \Gamma$,
- tape = list of symbols delimited by sentinels $¢$ and $\$$,

Restarting Automaton

RRWW-automaton

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

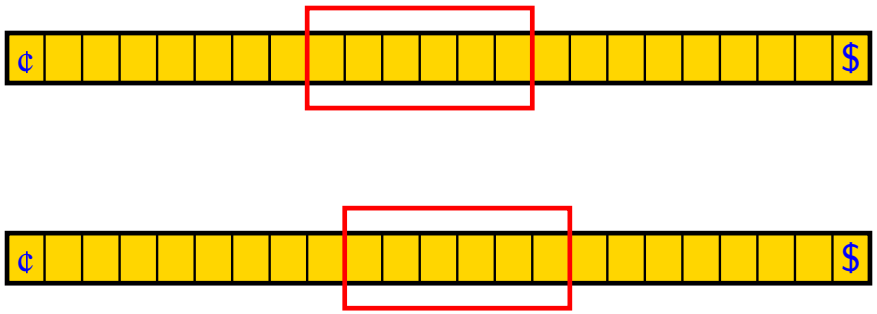
- finite state control, set of states Q,
- q_0 is the initial state,
- input alphabet Σ,
- working alphabet Γ, $\Sigma \subseteq \Gamma$,
- tape = list of symbols delimited by sentinels $¢$ and $\$$,
- read/write window of a fixed size k,

Restarting Automaton

RRWW-automaton

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

- finite state control, set of states Q,
- q_0 is the initial state,
- input alphabet Σ,
- working alphabet Γ, $\Sigma \subseteq \Gamma$,
- tape = list of symbols delimited by sentinels $¢$ and $\$$,
- read/write window of a fixed size k,
- transition function δ enables operations: MVR, rewrite, Restart, Accept.

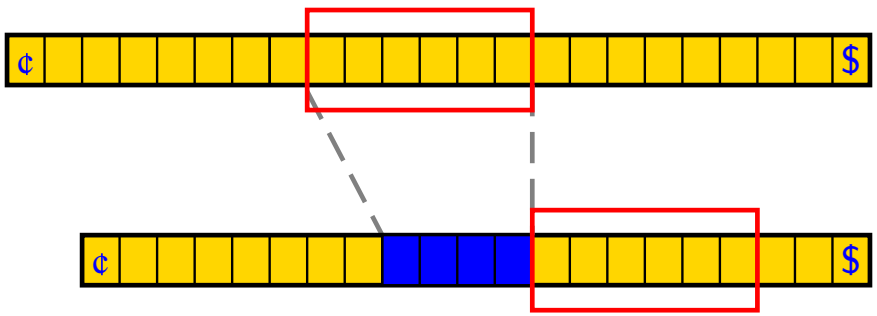


Restarting Automaton

RRWW-automaton

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

- finite state control, set of states Q,
- q_0 is the initial state,
- input alphabet Σ,
- working alphabet Γ, $\Sigma \subseteq \Gamma$,
- tape = list of symbols delimited by sentinels $¢$ and $\$$,
- read/write window of a fixed size k,
- transition function δ enables operations: MVR, rewrite, Restart, Accept.

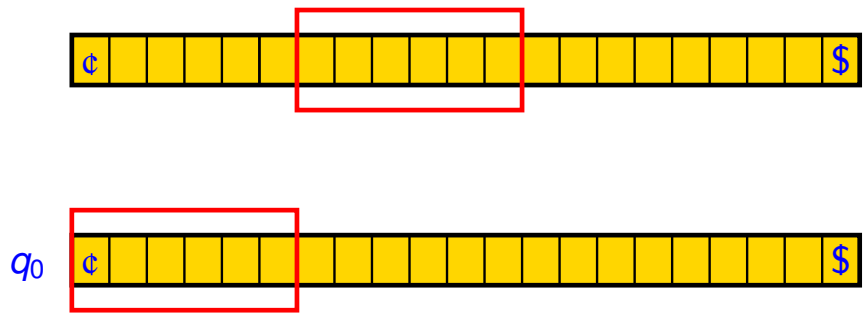


Restarting Automaton

RRWW-automaton

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

- finite state control, set of states Q,
- q_0 is the initial state,
- input alphabet Σ,
- working alphabet Γ, $\Sigma \subseteq \Gamma$,
- tape = list of symbols delimited by sentinels $¢$ and $\$$,
- read/write window of a fixed size k,
- transition function δ enables operations: MVR, rewrite, Restart, Accept.



Restarting Automaton

RRWW-automaton

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

- finite state control, set of states Q,
- q_0 is the initial state,
- input alphabet Σ,
- working alphabet Γ, $\Sigma \subseteq \Gamma$,
- tape = list of symbols delimited by sentinels $¢$ and $\$$,
- read/write window of a fixed size k,
- transition function δ enables operations: MVR, rewrite, Restart, Accept.

Accepted Language

- a *configuration* of an RRWW-automaton M: $\alpha q \beta$
  - q is the current state
  - $\alpha\beta \in \{¢\} \cdot \Gamma^* \cdot \{\$\}$ is the current contents of the tape
  - contents of the window = the first k symbols of β

Accepted Language

- a *configuration* of an RRWW-automaton M: $\alpha q \beta$
  - q is the current state
  - $\alpha\beta \in \{¢\} \cdot \Gamma^* \cdot \{\$\}$ is the current contents of the tape
  - contents of the window = the first k symbols of β
- a *restarting configuration*: $q_0 ¢ w \$$, where $w \in \Gamma^*$

Accepted Language

- a *configuration* of an RRWW-automaton M: $\alpha q \beta$
  - q is the current state
  - $\alpha\beta \in \{¢\} \cdot \Gamma^* \cdot \{\$\}$ is the current contents of the tape
  - contents of the window = the first k symbols of β
- a *restarting configuration*: $q_0 ¢ w \$$, where $w \in \Gamma^*$
- an *initial configuration*: $q_0 ¢ w \$$, where $w \in \Sigma^*$

Accepted Language

- a *configuration* of an RRWW-automaton M: $\alpha q \beta$
  - q is the current state
  - $\alpha\beta \in \{¢\} \cdot \Gamma^* \cdot \{\$\}$ is the current contents of the tape
  - contents of the window = the first k symbols of β
- a *restarting configuration*: $q_0 ¢ w \$$, where $w \in \Gamma^*$
- an *initial configuration*: $q_0 ¢ w \$$, where $w \in \Sigma^*$
- when ignoring MVR-steps, rewrite steps and restart steps alternate, rewrite step coming first

Accepted Language

- a *configuration* of an RRWW-automaton M: $\alpha q \beta$
  - q is the current state
  - $\alpha\beta \in \{¢\} \cdot \Gamma^* \cdot \{\$\}$ is the current contents of the tape
  - contents of the window = the first k symbols of β
- a *restarting configuration*: $q_0 ¢ w \$$, where $w \in \Gamma^*$
- an *initial configuration*: $q_0 ¢ w \$$, where $w \in \Sigma^*$
- when ignoring MVR-steps, rewrite steps and restart steps alternate, rewrite step coming first
- M accepts the language

$$L(M) = \{w \in \Sigma^* \mid q_0 ¢ w \$ \vdash_M^* \text{Accept}\}.$$

Cycles, Tails

- A cycle = a part of a computation between a restarting configuration and the configuration after a restart step.

Cycles, Tails

- A cycle = a part of a computation between a restarting configuration and the configuration after a restart step.
- A tail = a part of a computation after the last restarting configuration and a halting step (Acceptor reject).

Cycles, Tails

- A cycle = a part of a computation between a restarting configuration and the configuration after a restart step.
- A tail = a part of a computation after the last restarting configuration and a halting step (Acceptor reject).
- Notation:
  - $q_0 ¢ u \$ \vdash_M^c q_0 ¢ v \$$ denotes a cycle of M

Cycles, Tails

- A cycle = a part of a computation between a restarting configuration and the configuration after a restart step.
- A tail = a part of a computation after the last restarting configuration and a halting step (Acceptor reject).
- Notation:
  - $q_0 ¢ u \$ \vdash_M^c q_0 ¢ v \$$ denotes a cycle of M
  - For each $k \geq 1$, let $\mathrm{X}(k)$ denote automata of type X that have read/write window of size k.

Cycles, Tails

- A cycle = a part of a computation between a restarting configuration and the configuration after a restart step.
- A tail = a part of a computation after the last restarting configuration and a halting step (Acceptor reject).
- Notation:
  - $q_0 ¢ u \$ \vdash_M^c q_0 ¢ v \$$ denotes a cycle of M
  - For each $k \geq 1$, let $\mathrm{X}(k)$ denote automata of type X that have read/write window of size k.
  - Let $\mathcal{L}(\mathrm{X}(k))$ denote the class of languages accepted by $\mathrm{X}(k)$-automata.

Monotone RRWW-automata

- Each cycle contains a configuration $¢\alpha q\beta\$$ in which a rewrite step is applied. Then $D_r(C) = |\beta\$|$ is the *right distance* of C.

Monotone RRWW-automata

- Each cycle contains a configuration $¢\alpha q \beta \$$ in which a rewrite step is applied. Then $D_r(C) = |\beta\$|$ is the *right distance* of C.
- Monotonicity: a sequence of cycles $C_1, \ldots, C_n$ is monotone if $D_r(C_1) \geq D_r(C_2) \geq \cdots \geq D_r(C_n)$.

Monotone RRWW-automata

- Each cycle contains a configuration $¢\alpha q \beta \$$ in which a rewrite step is applied. Then $D_r(C) = |\beta\$|$ is the *right distance* of C.
- Monotonicity: a sequence of cycles $C_1, \ldots, C_n$ is monotone if $D_r(C_1) \geq D_r(C_2) \geq \cdots \geq D_r(C_n)$.
- M is monotone if all its computations are monotone.

Monotone RRWW-automata

- Each cycle contains a configuration $¢\alpha q \beta \$$ in which a rewrite step is applied. Then $D_r(C) = |\beta\$|$ is the *right distance* of C.
- Monotonicity: a sequence of cycles $C_1, \ldots, C_n$ is monotone if $D_r(C_1) \geq D_r(C_2) \geq \cdots \geq D_r(C_n)$.
- M is monotone if all its computations are monotone.

Theorem 1 (Mráz, Otto '19, Schluter '15)

(a) CFL $=$ $\mathcal{L}$(mon-RWW(2)) $=$ $\mathcal{L}$(mon-RRWW(2)).

(b) DCFL $=$ $\mathcal{L}$(det-mon-RWW(2)) $=$ $\mathcal{L}$(det-mon-RRWW(2)).

Monotone RRWW-automata

- Each cycle contains a configuration $¢\alpha q \beta \$$ in which a rewrite step is applied. Then $D_r(C) = |\beta\$|$ is the *right distance* of C.
- Monotonicity: a sequence of cycles $C_1, \ldots, C_n$ is monotone if $D_r(C_1) \geq D_r(C_2) \geq \cdots \geq D_r(C_n)$.
- M is monotone if all its computations are monotone.

Theorem 1 (Mráz, Otto '19, Schluter '15)

(a) CFL $= \mathcal{L}$(mon-RWW(2)) $= \mathcal{L}$(mon-RRWW(2)).
(b) DCFL $= \mathcal{L}$(det-mon-RWW(2)) $= \mathcal{L}$(det-mon-RRWW(2)).

Theorem 2 (Kutrib, Reimann '07, Mráz '01)

(a) REG $= \mathcal{L}$((mon-)RWW(1)) $= \mathcal{L}$(det-(mon-)RRWW(1)).
(b) REG $\subsetneq \mathcal{L}$(mon-RRWW(1)) $\subsetneq$ CFL.

Monotone Shrinking RRWW- and RWW-automata

Window size 2

- shrinking = rewrites are weight-reducing instead of length-reducing
- monotone = the distance of the places of rewriting from the right end does not increase
- evidently "length-reducing" $\Rightarrow$ "shrinking"

Corollary 3 (Mráz, Otto '19, Schlutter '15)

(a) CFL $= \mathcal{L}$(mon-sRWW(2)) $= \mathcal{L}$(mon-sRRWW(2)).

(b) DCFL $= \mathcal{L}$(det-mon-sRWW(2)) $= \mathcal{L}$(det-mon-sRRWW(2)).

Monotone Shrinking RWW(1)-automata

Theorem 4

$\mathcal{L}$(mon-sRWW(1)) = REG.

Proof idea:

- Convert sRWW(1)-automaton into a stateless ORWW-automaton.
- ORWW-automaton:
  - an sRWW(3)-automaton
  - cannot delete
  - can rewrite only the symbol in the middle of the window by a smaller letter
- [Kwee, Otto '16] Stateless ORWW-automata accept REG.

Monotone Shrinking RRWW(1)-automata

Theorem 5

$\mathrm{REG} \subsetneq \mathcal{L}(\text{mon-RRWW(1)}) \subsetneq \mathcal{L}(\text{mon-sRRWW(1)})$.

Proof:

- The language $L_1 = \{ a^m b^n \mid m \in \{n, n+1\}, n \geq 0 \}$ is not regular and there exists a monotone RRWW(1)-automaton M accepting L_1.

Monotone Shrinking RRWW(1)-automata

Theorem 5

$\mathrm{REG} \subsetneq \mathcal{L}(\text{mon-RRWW(1)}) \subsetneq \mathcal{L}(\text{mon-sRRWW(1)})$.

Proof:

- The language $L_1 = \{ a^m b^n \mid m \in \{n, n+1\}, n \geq 0 \}$ is not regular and there exists a monotone RRWW(1)-automaton M accepting L_1.
  On an input word $w \in \{a, b\}^*$
  - Check that $w = a^m b^n$, for some $m, n \geq 0$.

Monotone Shrinking RRWW(1)-automata

Theorem 5

$\mathrm{REG} \subsetneq \mathcal{L}(\text{mon-RRWW(1)}) \subsetneq \mathcal{L}(\text{mon-sRRWW(1)})$.

Proof:

- The language $L_1 = \{ a^m b^n \mid m \in \{n, n+1\}, n \geq 0 \}$ is not regular and there exists a monotone RRWW(1)-automaton M accepting L_1.

  On an input word $w \in \{a, b\}^*$
  - Check that $w = a^m b^n$, for some $m, n \geq 0$.
  - Guess whether the parities of m and n are the same and check it later.

Monotone Shrinking RRWW(1)-automata

Theorem 5

$\mathrm{REG} \subsetneq \mathcal{L}(\text{mon-RRWW(1)}) \subsetneq \mathcal{L}(\text{mon-sRRWW(1)})$.

Proof:

- The language $L_1 = \{ a^m b^n \mid m \in \{n, n+1\}, n \geq 0 \}$ is not regular and there exists a monotone RRWW(1)-automaton M accepting L_1.

  On an input word $w \in \{a, b\}^*$
  - Check that $w = a^m b^n$, for some $m, n \geq 0$.
  - Guess whether the parities of m and n are the same and check it later.
  - If $m \mod 2 = n \mod 2$, delete the first b, otherwise delete the last a.

Monotone Shrinking RRWW(1)-automata

Theorem 5

$\mathrm{REG} \subsetneq \mathcal{L}(\text{mon-RRWW(1)}) \subsetneq \mathcal{L}(\text{mon-sRRWW(1)})$.

Proof:

- The language $L_1 = \{ a^m b^n \mid m \in \{n, n+1\}, n \geq 0 \}$ is not regular and there exists a monotone RRWW(1)-automaton M accepting L_1.

  On an input word $w \in \{a, b\}^*$
  - Check that $w = a^m b^n$, for some $m, n \geq 0$.
  - Guess whether the parities of m and n are the same and check it later.
  - If $m \mod 2 = n \mod 2$, delete the first b, otherwise delete the last a.
  - Observe: if $w \in L_1$ and $w \Rightarrow^c w'$ then $w' \in L$. The automaton is monotone.

Monotone Shrinking RRWW(1)-automata

Theorem 5

REG $\subsetneq \mathcal{L}$(mon-RRWW(1)) $\subsetneq \mathcal{L}$(mon-sRRWW(1)).

Proof:

- The language $L_1 = \{ a^m b^n \mid m \in \{n, n+1\}, n \geq 0 \}$ is not regular and there exists a monotone RRWW(1)-automaton M accepting L_1.
- The language $L_2 = \{ a^m b^m \mid m \geq 0 \}$ cannot be accepted by any RRWW(1)-automaton, but L_2 is accepted by an sRRWW(1)-automaton.

Monotone Shrinking RRWW(1)-automata

Theorem 5

REG $\subsetneq \mathcal{L}$(mon-RRWW(1)) $\subsetneq \mathcal{L}$(mon-sRRWW(1)).

Proof:

- The language $L_1 = \{ a^m b^n \mid m \in \{n, n+1\}, n \geq 0 \}$ is not regular and there exists a monotone RRWW(1)-automaton M accepting L_1.
- The language $L_2 = \{ a^m b^m \mid m \geq 0 \}$ cannot be accepted by any RRWW(1)-automaton, but L_2 is accepted by an sRRWW(1)-automaton.
  - Modify M: during the first cycle replace the first a by an auxiliary symbol a' and check that the parities of m and n are the same.

Monotone Shrinking RRWW(1)-automata

Theorem 5

$\mathrm{REG} \subsetneq \mathcal{L}(\text{mon-RRWW(1)}) \subsetneq \mathcal{L}(\text{mon-sRRWW(1)})$.

Proof:

- The language $L_1 = \{ a^m b^n \mid m \in \{n, n+1\}, n \geq 0 \}$ is not regular and there exists a monotone RRWW(1)-automaton M accepting L_1.
- The language $L_2 = \{ a^m b^m \mid m \geq 0 \}$ cannot be accepted by any RRWW(1)-automaton, but L_2 is accepted by an sRRWW(1)-automaton.

Open problem

$\mathrm{CFL} \stackrel{?}{=} \mathcal{L}(\text{mon-sRRWW(1)})$.

sRRWW-automata and Finite Change Automata

- Finite change automaton is a nondeterministic linear-bounded automaton that does not change the contents of any tape cell more than r times during any accepting computation, where $r \geq 1$ is a constant.

sRRWW-automata and Finite Change Automata

- Finite change automaton is a nondeterministic linear-bounded automaton that does not change the contents of any tape cell more than r times during any accepting computation, where $r \geq 1$ is a constant.

Theorem 6 (Jurdziński, Otto '05)

$\mathcal{L}(\mathsf{FA}) = \mathcal{L}(\mathsf{sRWW}) = \mathcal{L}(\mathsf{sRRWW})$.

sRRWW-automata and Finite Change Automata

- Finite change automaton is a nondeterministic linear-bounded automaton that does not change the contents of any tape cell more than r times during any accepting computation, where $r \geq 1$ is a constant.

Theorem 6 (Jurdziński, Otto '05)

$\mathcal{L}(\mathrm{FA}) = \mathcal{L}(\mathrm{sRWW}) = \mathcal{L}(\mathrm{sRRWW})$.

- By inspecting the proof in [Jurdziński, Otto '05]

Corollary 7

$\mathcal{L}(\mathrm{FA}) = \mathcal{L}(\mathrm{sRRWW}(1)) = \mathcal{L}(\mathrm{ORRWW})$.

An ORRWW-automaton is an sRRWW(3)-automaton that rewrites only the symbol in the middle of the window by a letter of less weight.

sRRWW-automata and Finite Change Automata

- [Jurdziński, Otto '05] provides a simulation of an arbitrary sRRWW(k)-automaton by a sRWW-automaton of window size $\max\{2k, 9\}$

sRRWW-automata and Finite Change Automata

- [Jurdziński, Otto '05] provides a simulation of an arbitrary sRRWW(k)-automaton by a sRWW-automaton of window size $\max\{2k, 9\}$

Corollary 8

$\mathcal{L}(\mathrm{FA}) = \mathcal{L}(\mathrm{sRWW}(9))$.

sRRWW-automata and Finite Change Automata

- [Jurdziński, Otto '05] provides a simulation of an arbitrary sRRWW(k)-automaton by a sRWW-automaton of window size $\max\{2k, 9\}$

Corollary 8

$\mathcal{L}(\text{FA}) = \mathcal{L}(\text{sRWW}(9))$.

Open problem

Are sRWW(k)-automata, for $k = 2, 3, , \ldots, 8$, strictly weaker than finite-change automata?

- We will see that already sRWW(2)-automata are quite strong

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

1. Trivially, $\mathrm{REG} \subseteq \mathcal{L}(\mathrm{sRWW}(1))$.

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

1. Trivially, $\mathrm{REG} \subseteq \mathcal{L}(\mathrm{sRWW}(1))$.
2. $L_{\geq} = \{ a^{m+n}b^n \mid m, n \geq 0 \}$, is not regular, but it is accepted by an sRWW(1)-automaton $M_{\geq}$; hence $\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1))$.

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

1. Trivially, $\mathrm{REG} \subseteq \mathcal{L}(\mathrm{sRWW}(1))$.
2. $L_{\geq} = \{ a^{m+n}b^n \mid m, n \geq 0 \}$, is not regular, but it is accepted by an sRWW(1)-automaton $M_{\geq}$; hence $\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1))$.
3. Each sRWW(1)-automaton can be simulated by an ORWW-automaton

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

- $L_{\geq} = \{ a^{m+n}b^n \mid m, n \geq 0 \}$, is not regular, but it is accepted by an sRWW(1)-automaton $M_{\geq}$; hence $\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1))$.
  - input alphabet $\{a, b\}$

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

- $L_{\geq} = \{ a^{m+n} b^n \mid m, n \geq 0 \}$, is not regular, but it is accepted by an sRWW(1)-automaton $M_{\geq}$; hence $\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1))$.
  - input alphabet $\{a, b\}$
  - working alphabet $\{a, b, a_1, a_2, a_3, b_1, b_2\}$, $\omega(a) > \omega(a_1) > \omega(a_2) > \omega(a_3)$, $\omega(b) > \omega(b_1) > \omega(b_2)$

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

- $L_{\geq} = \{ a^{m+n} b^n \mid m, n \geq 0 \}$, is not regular, but it is accepted by an sRWW(1)-automaton $M_{\geq}$; hence $\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1))$.
  - input alphabet $\{a, b\}$
  - working alphabet $\{a, b, a_1, a_2, a_3, b_1, b_2\}$, $\omega(a) > \omega(a_1) > \omega(a_2) > \omega(a_3)$, $\omega(b) > \omega(b_1) > \omega(b_2)$
  - rewrites: either the leftmost occurrence of $a \to a_1 \to a_2 \to a_3$ or the leftmost occurrence of $b \to b_1 \to b_2$

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

- $L_{\geq} = \{ a^{m+n} b^n \mid m, n \geq 0 \}$, is not regular, but it is accepted by an sRWW(1)-automaton $M_{\geq}$; hence $\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1))$.
  - input alphabet $\{a, b\}$
  - working alphabet $\{a, b, a_1, a_2, a_3, b_1, b_2\}$, $\omega(a) > \omega(a_1) > \omega(a_2) > \omega(a_3)$, $\omega(b) > \omega(b_1) > \omega(b_2)$
  - rewrites: either the leftmost occurrence of $a \to a_1 \to a_2 \to a_3$ or the leftmost occurrence of $b \to b_1 \to b_2$
  - directly accepts any word of the form $a_3^m b_2^n$ or $a_3^m a_1 a^r b_2^n$, $m, n, r \geq 0$

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

- $L_{\geq} = \{ a^{m+n} b^n \mid m, n \geq 0 \}$, is not regular, but it is accepted by an sRWW(1)-automaton $M_{\geq}$; hence $\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1))$.
  - input alphabet $\{a, b\}$
  - working alphabet $\{a, b, a_1, a_2, a_3, b_1, b_2\}$, $\omega(a) > \omega(a_1) > \omega(a_2) > \omega(a_3)$, $\omega(b) > \omega(b_1) > \omega(b_2)$
  - rewrites: either the leftmost occurrence of $a \to a_1 \to a_2 \to a_3$ or the leftmost occurrence of $b \to b_1 \to b_2$
  - directly accepts any word of the form $a_3^m b_2^n$ or $a_3^m a_1 a^r b_2^n$, $m, n, r \geq 0$
  - b can be rewritten into b_1 only if the last rewritten a is a_1, and

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

- $L_{\geq} = \{ a^{m+n} b^n \mid m, n \geq 0 \}$, is not regular, but it is accepted by an sRWW(1)-automaton $M_{\geq}$; hence $\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1))$.
  - input alphabet $\{a, b\}$
  - working alphabet $\{a, b, a_1, a_2, a_3, b_1, b_2\}$, $\omega(a) > \omega(a_1) > \omega(a_2) > \omega(a_3)$, $\omega(b) > \omega(b_1) > \omega(b_2)$
  - rewrites: either the leftmost occurrence of $a \to a_1 \to a_2 \to a_3$ or the leftmost occurrence of $b \to b_1 \to b_2$
  - directly accepts any word of the form $a_3^m b_2^n$ or $a_3^m a_1 a^r b_2^n$, $m, n, r \geq 0$
  - b can be rewritten into b_1 only if the last rewritten a is a_1, and
  - b_1 can be rewritten into b_2 only if the last rewritten a is a_2

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

- $L_{\geq} = \{ a^{m+n} b^n \mid m, n \geq 0 \}$, is not regular, but it is accepted by an sRWW(1)-automaton $M_{\geq}$; hence $\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1))$.
  - input alphabet $\{a, b\}$
  - working alphabet $\{a, b, a_1, a_2, a_3, b_1, b_2\}$, $\omega(a) > \omega(a_1) > \omega(a_2) > \omega(a_3)$, $\omega(b) > \omega(b_1) > \omega(b_2)$
  - rewrites: either the leftmost occurrence of $a \to a_1 \to a_2 \to a_3$ or the leftmost occurrence of $b \to b_1 \to b_2$
  - directly accepts any word of the form $a_3^m b_2^n$ or $a_3^m a_1 a^r b_2^n$, $m, n, r \geq 0$
  - b can be rewritten into b_1 only if the last rewritten a is a_1, and
  - b_1 can be rewritten into b_2 only if the last rewritten a is a_2
  - the number of rewritten bs cannot be higher than the number of rewritten as.

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

- Each sRWW(1)-automaton M can be simulated by an ORWW-automaton M_o

Shrinking RWW(1)-automata

Theorem 9

REG $\subsetneq \mathcal{L}$(sRWW(1)) $\subseteq \mathcal{L}$(ORWW).

Proof:

- Each sRWW(1)-automaton M can be simulated by an ORWW-automaton M_o
  - Let Σ be the input alphabet of M and Γ be the working alphabet of M and $\varphi : \Gamma \to \mathbb{N}_+$ be the corresponding weight function.

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

- Each sRWW(1)-automaton M can be simulated by an ORWW-automaton M_o
  - Let Σ be the input alphabet of M and Γ be the working alphabet of M and $\varphi : \Gamma \to \mathbb{N}_+$ be the corresponding weight function.
  - Let $c = \max_{X \in \Gamma} \varphi(X)$ be the maximal weight of a working symbol of M.

Shrinking RWW(1)-automata

Theorem 9

REG $\subsetneq \mathcal{L}$(sRWW(1)) $\subseteq \mathcal{L}$(ORWW).

Proof:

- Each sRWW(1)-automaton M can be simulated by an ORWW-automaton M_o
  - Let Σ be the input alphabet of M and Γ be the working alphabet of M and $\varphi : \Gamma \to \mathbb{N}_+$ be the corresponding weight function.
  - Let $c = \max_{X \in \Gamma} \varphi(X)$ be the maximal weight of a working symbol of M.
  - M_o uses a working alphabet containing Σ and all symbols of the form $[\alpha]$, where $\alpha \in \Gamma^*$ and $\varphi(\alpha) \leq c$.

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

- Each sRWW(1)-automaton M can be simulated by an ORWW-automaton M_o
  - Let Σ be the input alphabet of M and Γ be the working alphabet of M and $\varphi : \Gamma \to \mathbb{N}_+$ be the corresponding weight function.
  - Let $c = \max_{X \in \Gamma} \varphi(X)$ be the maximal weight of a working symbol of M.
  - M_o uses a working alphabet containing Σ and all symbols of the form $[\alpha]$, where $\alpha \in \Gamma^*$ and $\varphi(\alpha) \leq c$.
  - The partial ordering $>$ of working alphabet of M_o:

Shrinking RWW(1)-automata

Theorem 9

REG $\subsetneq \mathcal{L}$(sRWW(1)) $\subseteq \mathcal{L}$(ORWW).

Proof:

- Each sRWW(1)-automaton M can be simulated by an ORWW-automaton M_o
  - Let Σ be the input alphabet of M and Γ be the working alphabet of M and $\varphi : \Gamma \to \mathbb{N}_+$ be the corresponding weight function.
  - Let $c = \max_{X \in \Gamma} \varphi(X)$ be the maximal weight of a working symbol of M.
  - M_o uses a working alphabet containing Σ and all symbols of the form $[\alpha]$, where $\alpha \in \Gamma^*$ and $\varphi(\alpha) \leq c$.
  - The partial ordering $>$ of working alphabet of M_o:
    - $a > [a]$, for each input symbol a,

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW}(1)) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

- Each sRWW(1)-automaton M can be simulated by an ORWW-automaton M_o
  - Let Σ be the input alphabet of M and Γ be the working alphabet of M and $\varphi : \Gamma \to \mathbb{N}_+$ be the corresponding weight function.
  - Let $c = \max_{X \in \Gamma} \varphi(X)$ be the maximal weight of a working symbol of M.
  - M_o uses a working alphabet containing Σ and all symbols of the form $[\alpha]$, where $\alpha \in \Gamma^*$ and $\varphi(\alpha) \leq c$.
  - The partial ordering $>$ of working alphabet of M_o:
    - $a > [a]$, for each input symbol a,
    - $[\alpha] > [\beta]$, if $\varphi(\alpha) > \varphi(\beta)$, for $\alpha, \beta \in \Gamma^*$.

Shrinking RWW(1)-automata

Theorem 9

$\mathrm{REG} \subsetneq \mathcal{L}(\mathrm{sRWW(1)}) \subseteq \mathcal{L}(\mathrm{ORWW})$.

Proof:

- Each sRWW(1)-automaton M can be simulated by an ORWW-automaton M_o
  - Let Σ be the input alphabet of M and Γ be the working alphabet of M and $\varphi : \Gamma \to \mathbb{N}_+$ be the corresponding weight function.
  - Let $c = \max_{X \in \Gamma} \varphi(X)$ be the maximal weight of a working symbol of M.
  - M_o uses a working alphabet containing Σ and all symbols of the form $[\alpha]$, where $\alpha \in \Gamma^*$ and $\varphi(\alpha) \leq c$.
  - The partial ordering $>$ of working alphabet of M_o:
    - $a > [a]$, for each input symbol a,
    - $[\alpha] > [\beta]$, if $\varphi(\alpha) > \varphi(\beta)$, for $\alpha, \beta \in \Gamma^*$.
  - M_o either rewrites an input symbol a into $[a]$, or simulates move-right steps of M over a string α encoded as $[\alpha]$, or simulates rewriting of a symbol within α stored as $[\alpha]$; the resulting string α' is stored as $[\alpha']$ and $\alpha > \alpha'$.

Further Results

Window size 1

- The language

$$L'_{\text{copy}} = \{\, w\#u \mid \begin{array}{l} w, u \in \{a, b\}^*, |w|, |u| \geq 2, \\ u \text{ is a scattered subword of } w \,\} \end{array}$$

  is accepted by an sRWW(1)-automaton.
- L'_{copy} is not even growing context-sensitive [Kwee, Otto '16].
- The deterministic linear language $L = \{\, a^n b^n \mid n \geq 0 \,\}$ is not accepted by any ORWW-automaton [Kwee, Otto '18]. Thus, $\mathcal{L}(\text{ORWW})$ is clearly a proper subclass of $\mathcal{L}(\text{sRWW}) = \mathcal{L}(\text{FA})$.

Open problem

Is the inclusion $\mathcal{L}(\text{sRWW(1)}) \subseteq \mathcal{L}(\text{ORWW})$ proper?

Further Results

sRWW-automata of window size 2

- GCSL = the class of languages generated bu the growing context-sensitive grammars.
- Based on an characterization of GCSL by shrinking two-pushdown automata (sTPDA) in [Buntrock, Otto '98] and the fact that GCSL $\subsetneq \mathcal{L}(\text{sRWW}) = \mathcal{L}(\text{FA})$:

Theorem 10

GCSL $\subsetneq \mathcal{L}(\text{sRWW}(2))$.

- Each ORWW-automaton can be simulated by an sRWW(2)-automaton:

Corollary 11

$\mathcal{L}(\text{ORWW}) \subsetneq \mathcal{L}(\text{sRWW}(2))$.

Further Results

Deterministic sRWW- amd sRRWW-automata

Theorem 12

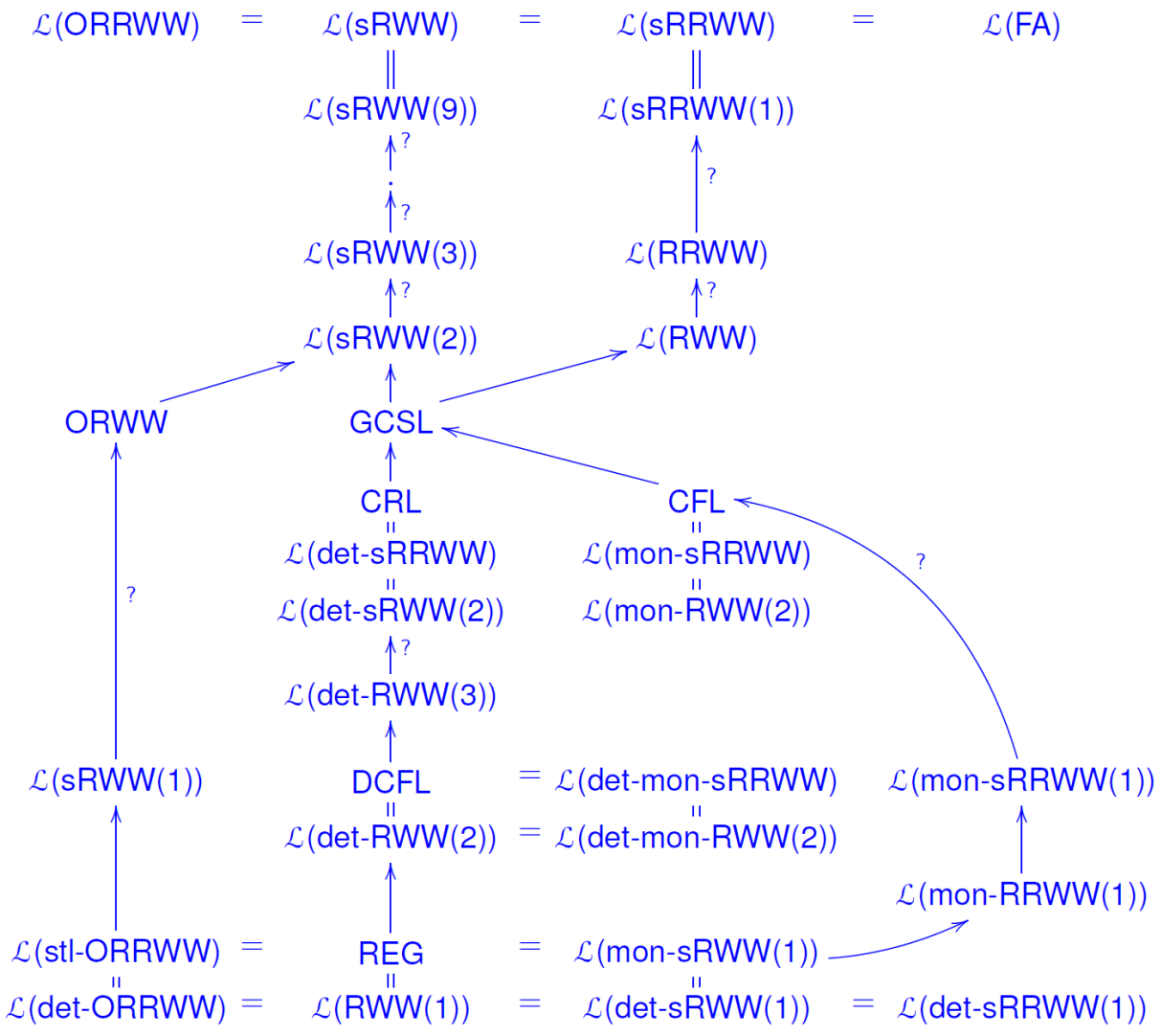
(a) $\mathcal{L}(\text{det-sRWW}(1)) = \mathcal{L}(\text{det-ORWW}) = \text{REG}.$
(b) $\mathcal{L}(\text{det-sRRWW}(1)) = \mathcal{L}(\text{det-ORRWW}) = \text{REG}.$

- Deterministic sTPDAs characterize the class CRL of Church-Rosser languages [Niemann, Otto '05].

Corollary 13

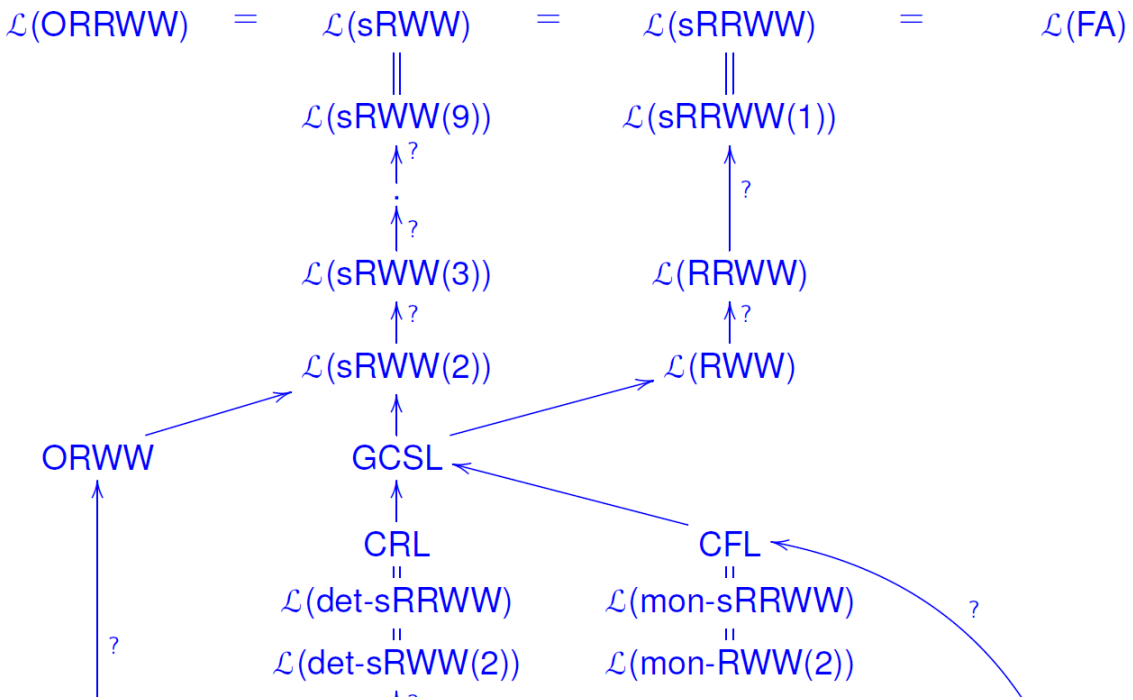
$\mathcal{L}(\text{det-sRWW}(2)) = \mathcal{L}(\text{det-sRRWW}(2)) = \mathcal{L}(\text{det-sRWW}) = \mathcal{L}(\text{det-sRRWW}) = \text{CRL}.$

Complete Diagram



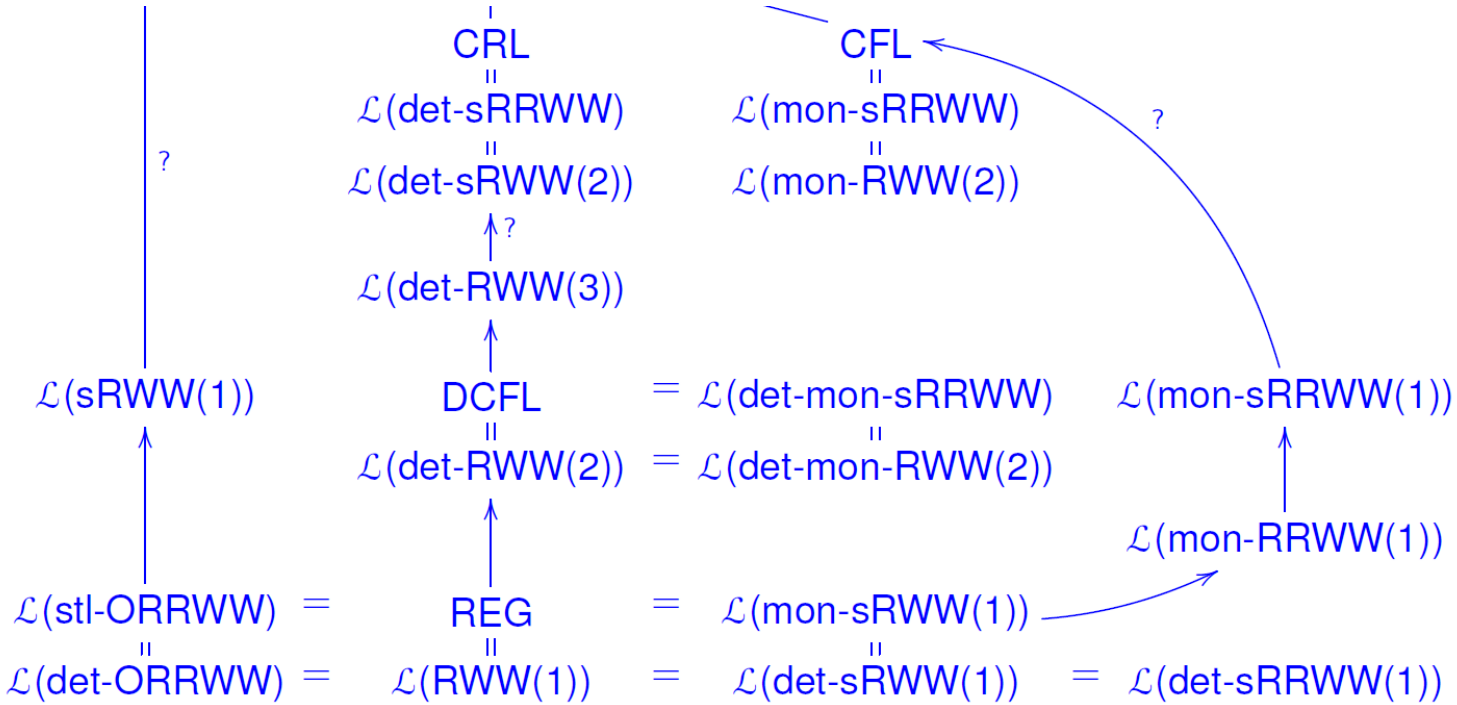
Conclusions

- for sRRWW-automata, already window size one suffices
- for monotone and/or deterministic sRWW- and sRRWW-automata, window size two is required to obtain the full expressive power
- for deterministic shrinking RWW- and RRWW-automata, the hierarchy based on window size consist of only two levels
- for deterministic RWW- and RRWW-automata we do not know



Conclusions

- it remains open whether sRWW(1)-automata are as powerful as ORWW-automata
- it remain open whether window size 9 is really needed to obtain the full power of sRWW-automata, and whether monotone sRRWW(1)-automata accept all context-free languages.



Thank you for your attention!