On the Influence of the Various Parameters of the Restarting Automaton on its Expressive Capacity and Descriptional Complexity

Friedrich Otto

Fachbereich Elektrotechnik/Informatik
Universität Kassel
34109 Kassel, Germany

f.otto@uni-kassel.de

DCFS 2023

Potsdam, Germany

July 4–6, 2023

Outline:

Otto, F.: RESTARTING AUTOMATA - Automata Inspired by the Linguistic Technique of Analysis by Reduction. Monograph, in preparation.

1. Introduction

In 1995, Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel presented the restarting automaton at the FCT in Dresden [1].

The restarting automaton is not simply another variant of the Turing machine, but it is a new type of automaton that is inspired by the linguistic technique of analysis by reduction (see, e.g., [2]).

An Example of Analysis by Reduction

We are given a sentence in a natural language, e.g.:

They mean that the means she means are very mean.

This sentence is now simplified step by step until a simple sentence is obtained:

1. Introduction

In 1995, Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel presented the restarting automaton at the FCT in Dresden [1].

The restarting automaton is not simply another variant of the Turing machine, but it is a new type of automaton that is inspired by the linguistic technique of analysis by reduction (see, e.g., [2]).

An Example of Analysis by Reduction

We are given a sentence in a natural language, e.g.:

They mean that the means she means are very mean.

This sentence is now simplified step by step until a simple sentence is obtained:

1. Introduction

In 1995, Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel presented the restarting automaton at the FCT in Dresden [1].

The restarting automaton is not simply another variant of the Turing machine, but it is a new type of automaton that is inspired by the linguistic technique of analysis by reduction (see, e.g., [2]).

An Example of Analysis by Reduction

We are given a sentence in a natural language, e.g.:

They mean that the means she means are very mean.

This sentence is now simplified step by step until a simple sentence is obtained:

1. Introduction

In 1995, Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel presented the restarting automaton at the FCT in Dresden [1].

The restarting automaton is not simply another variant of the Turing machine, but it is a new type of automaton that is inspired by the linguistic technique of analysis by reduction (see, e.g., [2]).

An Example of Analysis by Reduction

We are given a sentence in a natural language, e.g.:

They mean that the means she means are very mean.

This sentence is now simplified step by step until a simple sentence is obtained:

They mean that the means she means are very mean.

1. Introduction

In 1995, Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel presented the restarting automaton at the FCT in Dresden [1].

The restarting automaton is not simply another variant of the Turing machine, but it is a new type of automaton that is inspired by the linguistic technique of analysis by reduction (see, e.g., [2]).

An Example of Analysis by Reduction

We are given a sentence in a natural language, e.g.:

They mean that the means she means are very mean.

This sentence is now simplified step by step until a simple sentence is obtained:

They mean that the means she means are very mean.

1. Introduction

In 1995, Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel presented the restarting automaton at the FCT in Dresden [1].

The restarting automaton is not simply another variant of the Turing machine, but it is a new type of automaton that is inspired by the linguistic technique of analysis by reduction (see, e.g., [2]).

An Example of Analysis by Reduction

We are given a sentence in a natural language, e.g.:

They mean that the means she means are very mean.

This sentence is now simplified step by step until a simple sentence is obtained:

They mean that the means she means are mean.

1. Introduction

In 1995, Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel presented the restarting automaton at the FCT in Dresden [1].

The restarting automaton is not simply another variant of the Turing machine, but it is a new type of automaton that is inspired by the linguistic technique of analysis by reduction (see, e.g., [2]).

An Example of Analysis by Reduction

We are given a sentence in a natural language, e.g.:

They mean that the means she means are very mean.

This sentence is now simplified step by step until a simple sentence is obtained:

They mean that the means she means are mean.

1. Introduction

In 1995, Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel presented the restarting automaton at the FCT in Dresden [1].

The restarting automaton is not simply another variant of the Turing machine, but it is a new type of automaton that is inspired by the linguistic technique of analysis by reduction (see, e.g., [2]).

An Example of Analysis by Reduction

We are given a sentence in a natural language, e.g.:

They mean that the means she means are very mean.

This sentence is now simplified step by step until a simple sentence is obtained:

They mean the means she means are mean.

1. Introduction

In 1995, Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel presented the restarting automaton at the FCT in Dresden [1].

The restarting automaton is not simply another variant of the Turing machine, but it is a new type of automaton that is inspired by the linguistic technique of analysis by reduction (see, e.g., [2]).

An Example of Analysis by Reduction

We are given a sentence in a natural language, e.g.:

They mean that the means she means are very mean.

This sentence is now simplified step by step until a simple sentence is obtained:

They mean the means she means are mean.

1. Introduction

In 1995, Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel presented the restarting automaton at the FCT in Dresden [1].

The restarting automaton is not simply another variant of the Turing machine, but it is a new type of automaton that is inspired by the linguistic technique of analysis by reduction (see, e.g., [2]).

An Example of Analysis by Reduction

We are given a sentence in a natural language, e.g.:

They mean that the means she means are very mean.

This sentence is now simplified step by step until a simple sentence is obtained:

They mean the means are mean.

1. Introduction

In 1995, Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel presented the restarting automaton at the FCT in Dresden [1].

The restarting automaton is not simply another variant of the Turing machine, but it is a new type of automaton that is inspired by the linguistic technique of analysis by reduction (see, e.g., [2]).

An Example of Analysis by Reduction

We are given a sentence in a natural language, e.g.:

They mean that the means she means are very mean.

This sentence is now simplified step by step until a simple sentence is obtained:

They mean the means are mean.

1. Introduction

In 1995, Petr Jančar, František Mráz, Martin Plátek, and Jörg Vogel presented the restarting automaton at the FCT in Dresden [1].

The restarting automaton is not simply another variant of the Turing machine, but it is a new type of automaton that is inspired by the linguistic technique of analysis by reduction (see, e.g., [2]).

An Example of Analysis by Reduction

We are given a sentence in a natural language, e.g.:

They mean that the means she means are very mean.

This sentence is now simplified step by step until a simple sentence is obtained:

The means are mean.

Example (cont.)

To process a sentence in this way, one needs to understand its meaning. To automate this process, additional information is needed, which can be provided in the form of tags:

Word	Tags
They	<Pronoun> <Plural>
mean	<Verb> <Present Tense> <Plural>
that	<Conjunction>
the	<Article>
means	<Noun> <Plural>
she	<Pronoun> <Singular>
means	<Verb> <Present Tense> <Singular>
are	<Verb> <Present Tense> <Plural>
very	<Adverb>
mean	<Adjective>
.	<End of Sentence>

Each word is processed together with its tags.

Example (cont.)

To process a sentence in this way, one needs to understand its meaning. To automate this process, additional information is needed, which can be provided in the form of tags:

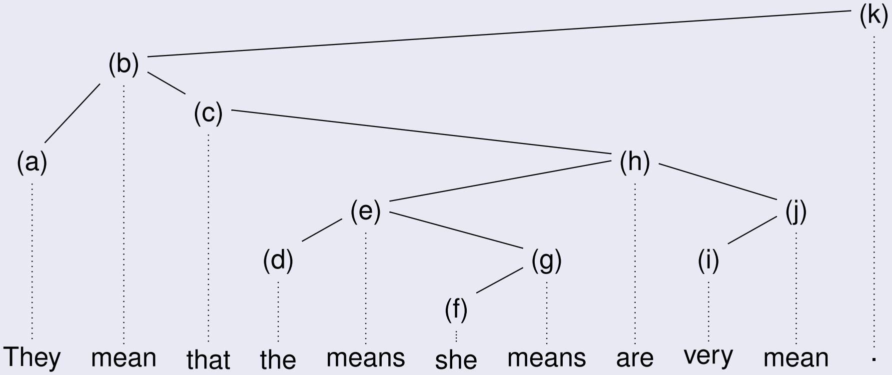
Word	Tags
They	<Pronoun> <Plural>
mean	<Verb> <Present Tense> <Plural>
that	<Conjunction>
the	<Article>
means	<Noun> <Plural>
she	<Pronoun> <Singular>
means	<Verb> <Present Tense> <Singular>
are	<Verb> <Present Tense> <Plural>
very	<Adverb>
mean	<Adjective>
.	<End of Sentence>

Each word is processed together with its tags.

Example (cont.)

In this way dependencies and independencies between words of a sentence can be determined.

This may result in the construction of a dependency tree:



Operations Required for Analysis by Reduction:

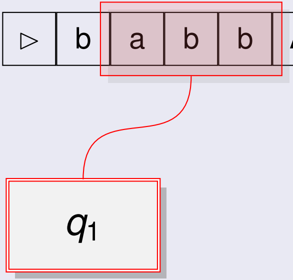
- Scan the current sentence (from left to right).
- Rewrite the current sentence by performing a local transformation.
- Restart in order to process the new sentence.
- Accept (if a correct simple sentence has been obtained).
- Reject (if an error has been detected).

2. The Restarting Automaton and Its Parameters

The Ingredients:

- The input sentence, which is a sequence of morphems, will be represented by a word, i.e., we need a (finite) input alphabet.
- To encode additional information (tags), we need auxiliary letters, i.e., we need a (finite) working alphabet.
- To mark the beginning and end of the current sentence, we need special symbols (sentinels).
- As the sentence analyzed will be transformed, the current word will be stored in a list of cells or a flexible tape.
- To scan the current word, we need a read/write window.
- To process information we need a finite-state control.
- To execute the analysis, we need a program (transition function) that determines the sequence of elementary operations.

The Hardware:



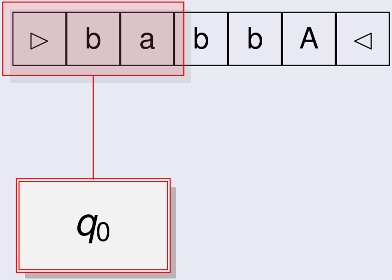
flexible tape
with sentinels

read/write window

finite-state control

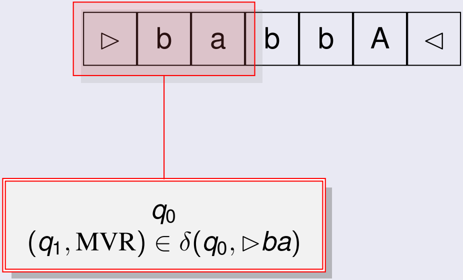
The Operations:

Move-Right Step (MVR):



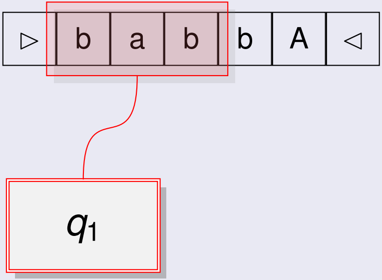
The Operations:

Move-Right Step (MVR):



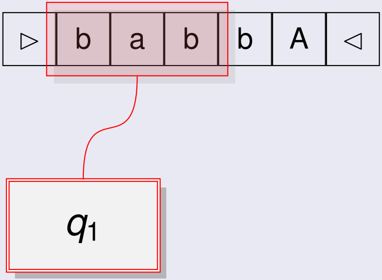
The Operations:

Move-Right Step (MVR):



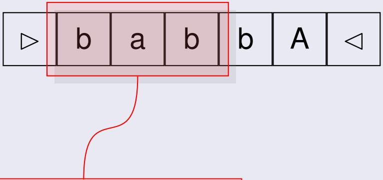
The Operations (cont.):

Move-Left Step (MVL):



The Operations (cont.):

Move-Left Step (MVL):

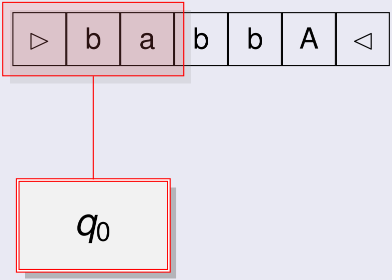


q_1

$(q_0, \mathrm{MVL}) \in \delta(q_1, bab)$

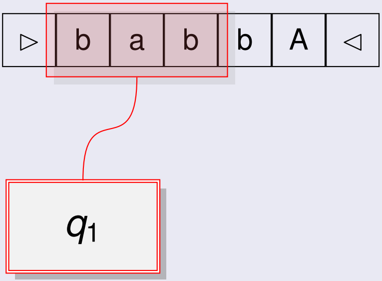
The Operations (cont.):

Move-Left Step (MVL):



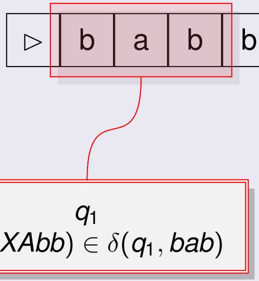
The Operations (cont.):

Rewrite Step:



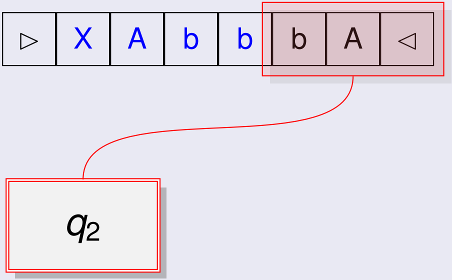
The Operations (cont.):

Rewrite Step:



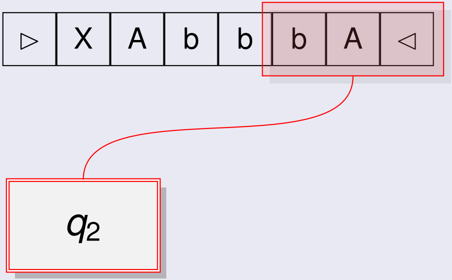
The Operations (cont.):

Rewrite Step:



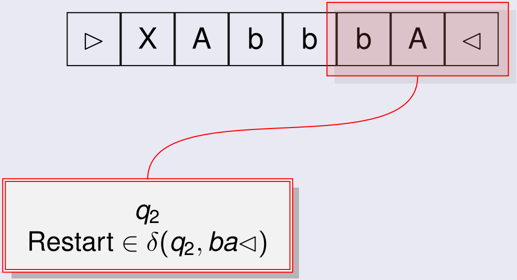
The Operations (cont.):

Restart Step:



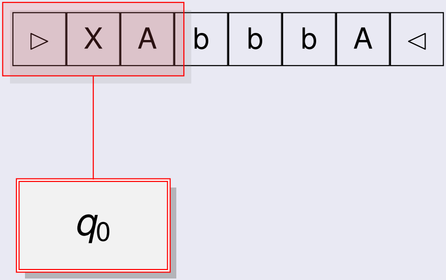
The Operations (cont.):

Restart Step:



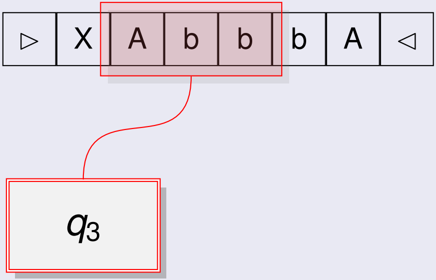
The Operations (cont.):

Restart Step:



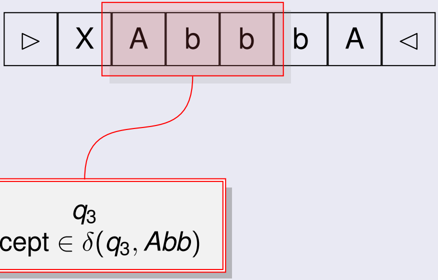
The Operations (cont.):

Accept Step:



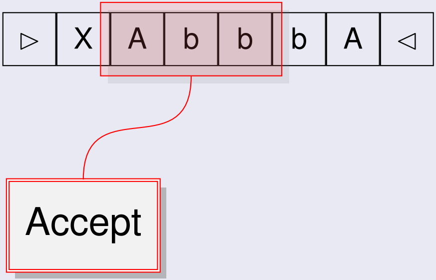
The Operations (cont.):

Accept Step:



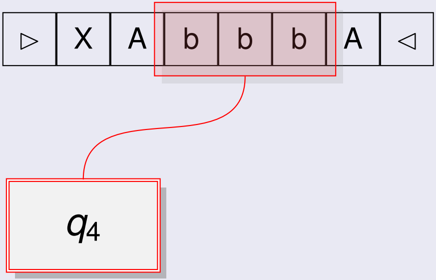
The Operations (cont.):

Accept Step:



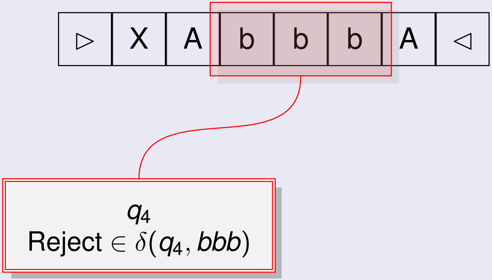
The Operations (cont.):

Reject Step:



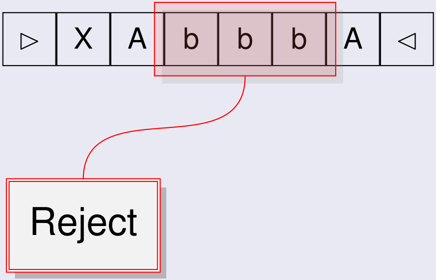
The Operations (cont.):

Reject Step:



The Operations (cont.):

Reject Step:



Definition 1

A restarting automaton is defined by an 8-tuple

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

where

- *Q is a finite set of states,*
- *Σ is a finite input alphabet,*
- *Γ is a finite working alphabet such that $\Sigma \subseteq \Gamma$,*
- *$\rhd, \lhd \notin \Gamma$ are the markers for the end of the tape (sentinels),*
- *$q_0 \in Q$ is the restart state, which also serves as initial state,*
- *$k \geq 1$ is the size of the read/write window, and*
- *$\delta : Q \times (\Gamma \cup \{\rhd, \lhd\})^k \to 2^{((Q\times\{\mathrm{MVR},\mathrm{MVL},\Gamma^*\})\cup\{\mathrm{Restart},\mathrm{Accept},\mathrm{Reject}\})}$ is the transition relation.*

Example 2

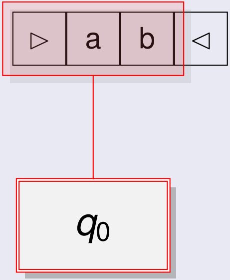
Let $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, 3, \delta)$ *be the following restarting automaton:*

- $Q = \{q_0\}$, $\Sigma = \{a, b\}$, $\Gamma = \Sigma \cup \{a_1, e, e_1, f, f_1, f_2\}$,
- *and* δ *is defined with combined rewrite/restart steps:*

(0)	$\delta(xyz)$	$\ni$	MVR	*for all* $x \in \Gamma \cup \{\rhd\}$ *and* $y, z \in \Gamma$,			
(1)	$\delta(\rhd\lhd)$	$\ni$	Accept,	(13)	$\delta(\rhd a_1 e_1)$	$\ni$	$\rhd e_1$,
(2)	$\delta(ab\lhd)$	$\ni$	$ae\lhd$,	(14)	$\delta(ae_1 e)$	$\ni$	ae,
(3)	$\delta(bb\lhd)$	$\ni$	$be\lhd, bf\lhd$,	(15)	$\delta(\rhd e_1 \lhd)$	$\ni$	$\rhd\lhd$,
(4)	$\delta(bbe)$	$\ni$	bee,	(16)	$\delta(aaf)$	$\ni$	$aa_1 f$,
(5)	$\delta(abe)$	$\ni$	aee,	(17)	$\delta(\rhd af)$	$\ni$	$\rhd a_1 f$,
(6)	$\delta(bbf)$	$\ni$	bff,	(18)	$\delta(a_1 ff)$	$\ni$	$a_1 f_1 f$,
(7)	$\delta(abf)$	$\ni$	aff,	(19)	$\delta(f_1 ff)$	$\ni$	$f_1 f_2 f$,
(8)	$\delta(aae)$	$\ni$	$aa_1 e$,	(20)	$\delta(f_1 f \lhd)$	$\ni$	$f_1 f_2 \lhd$,
(9)	$\delta(\rhd ae)$	$\ni$	$\rhd a_1 e$,	(21)	$\delta(a_1 f_1 f_2)$	$\ni$	$a_1 f_2$,
(10)	$\delta(a_1 ee)$	$\ni$	$a_1 e_1 e$,	(22)	$\delta(aa_1 f_2)$	$\ni$	af_2,
(11)	$\delta(a_1 e \lhd)$	$\ni$	$a_1 e_1 \lhd$,	(23)	$\delta(\rhd a_1 f_2)$	$\ni$	$\rhd f_2$,
(12)	$\delta(aa_1 e_1)$	$\ni$	ae_1,	(24)	$\delta(af_2 f)$	$\ni$	af,
				(25)	$\delta(\rhd f_2 \lhd)$	$\ni$	$\rhd\lhd$.

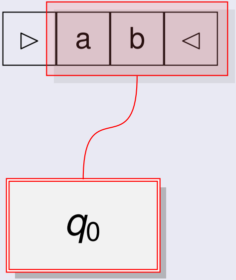
Example 2 (cont.)

This restarting automaton proceeds as follows:



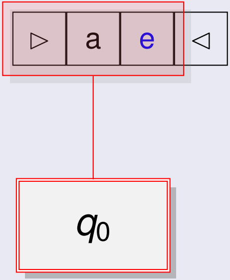
Example 2 (cont.)

This restarting automaton proceeds as follows:



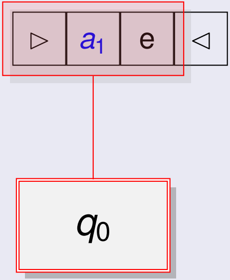
Example 2 (cont.)

This restarting automaton proceeds as follows:



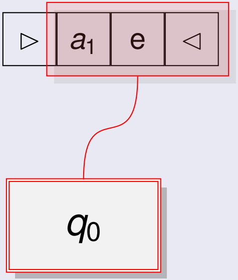
Example 2 (cont.)

This restarting automaton proceeds as follows:



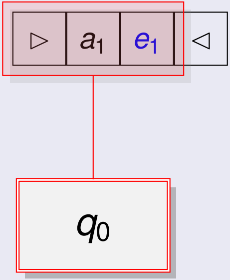
Example 2 (cont.)

This restarting automaton proceeds as follows:



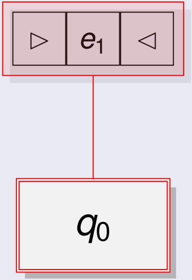
Example 2 (cont.)

This restarting automaton proceeds as follows:



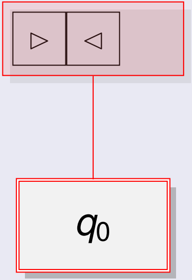
Example 2 (cont.)

This restarting automaton proceeds as follows:



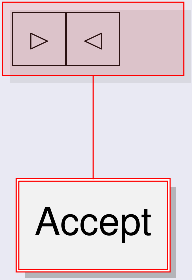
Example 2 (cont.)

This restarting automaton proceeds as follows:



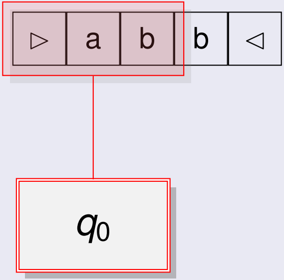
Example 2 (cont.)

This restarting automaton proceeds as follows:



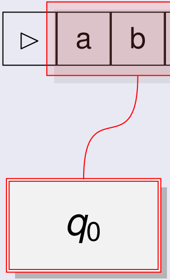
Example 2 (cont.)

This restarting automaton proceeds as follows:



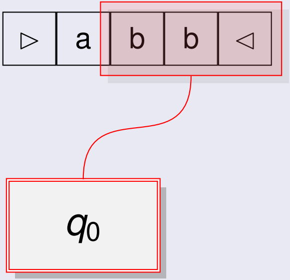
Example 2 (cont.)

This restarting automaton proceeds as follows:



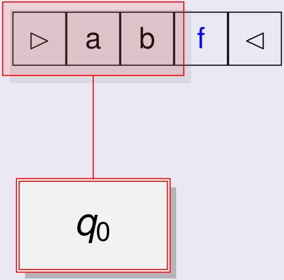
Example 2 (cont.)

This restarting automaton proceeds as follows:



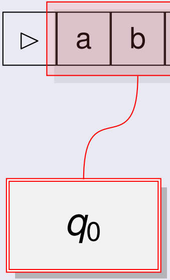
Example 2 (cont.)

This restarting automaton proceeds as follows:



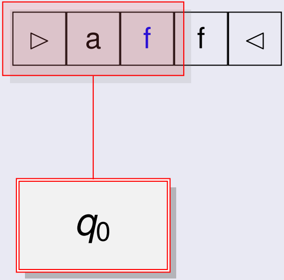
Example 2 (cont.)

This restarting automaton proceeds as follows:



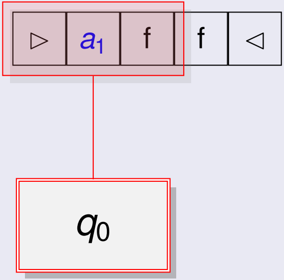
Example 2 (cont.)

This restarting automaton proceeds as follows:



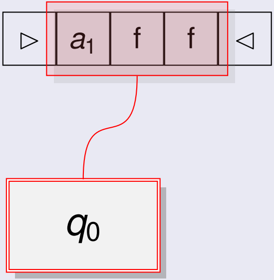
Example 2 (cont.)

This restarting automaton proceeds as follows:



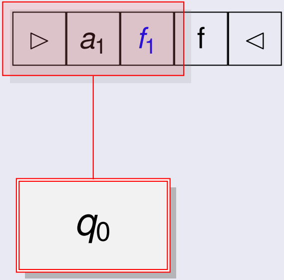
Example 2 (cont.)

This restarting automaton proceeds as follows:



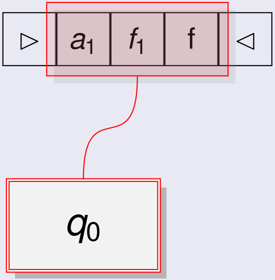
Example 2 (cont.)

This restarting automaton proceeds as follows:



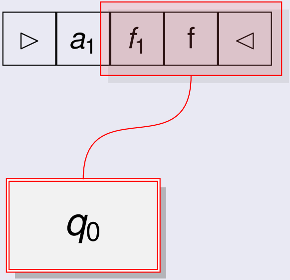
Example 2 (cont.)

This restarting automaton proceeds as follows:



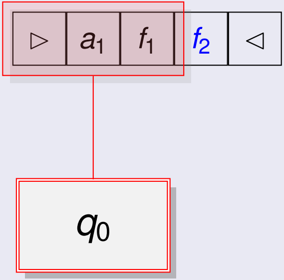
Example 2 (cont.)

This restarting automaton proceeds as follows:



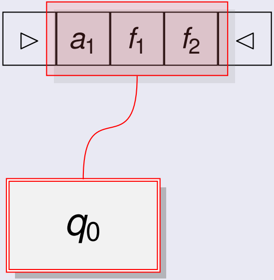
Example 2 (cont.)

This restarting automaton proceeds as follows:



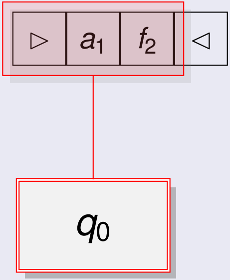
Example 2 (cont.)

This restarting automaton proceeds as follows:



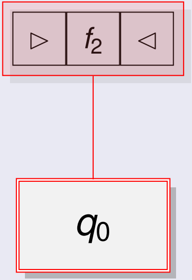
Example 2 (cont.)

This restarting automaton proceeds as follows:



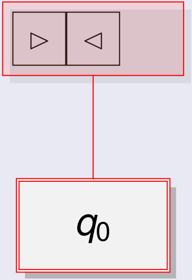
Example 2 (cont.)

This restarting automaton proceeds as follows:



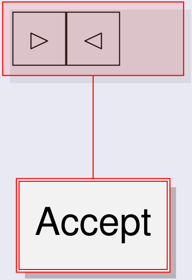
Example 2 (cont.)

This restarting automaton proceeds as follows:



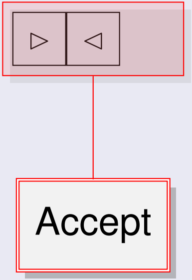
Example 2 (cont.)

This restarting automaton proceeds as follows:



Example 2 (cont.)

This restarting automaton proceeds as follows:



It accepts the language $L = \{ a^n b^n, a^n b^{2n} \mid n \geq 0 \} \in \mathrm{CFL} \smallsetminus \mathrm{DCFL}$.

Definition 1 (cont.)

Let $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, k, \delta)$ be a restarting automaton.

(a) A configuration of M is a word of the form $\rhd uqvw\lhd$, where $q \in Q$ and $u, v, w \in \Gamma^*$ with $|v| = k$.

(b) A restart configuration is of the form $q_0 \rhd vw\lhd$, where $v, w \in \Gamma^*$ with $|v| = k - 1$. If $v, w \in \Sigma^*$, then $q_0 \rhd vw\lhd$ is an initial configuration.

(c) A sequence of steps leading from a restart configuration $q_0 \rhd vw\lhd$ to the next restart configuration $q_0 \rhd xy\lhd$ is called a cycle. It is written as $q_0 \rhd vw \lhd \vdash_M^c q_0 \rhd xy\lhd$.
Condition: Each cycle of each computation of M must contain at least one rewrite step!

(d) A sequence of steps leading from a restart configuration to a final configuration (accepting or rejecting) without going through another restart configuration is called a tail.

Remark 3

A terminating computation of M consists of a finite sequence of cycles that is followed by a tail.

Definition 1 (cont.)

(e) $L_C(M) = \{\, w \in \Gamma^* \mid q_0 \rhd w \lhd \vdash_M^* \text{Accept} \,\}$
is the basic (or characteristic) language of M.

(f) $L(M) = \{\, w \in \Sigma^* \mid q_0 \rhd w \lhd \vdash_M^* \text{Accept} \,\}$
is the input language of M.

Remark 4

$L(M) = L_C(M) \cap \Sigma^*$.

Parameters of the Restarting Automaton

The type of a restarting automaton depends on many parameters:

- the number $|Q| \geq 1$ of states
- the number $|\Gamma \smallsetminus \Sigma| \geq 0$ of auxiliary letters
- the size $k \geq 1$ of the read/write window
- the type of move steps allowed
- the type of rewrite steps allowed
- the number $j \geq 1$ of allowed rewrite steps per cycle
- the positions at which rewrite steps are executed

The automaton from Example 2 satisfies $|Q| = 1$, $|\Gamma \smallsetminus \Sigma| = 6$, $k = 3$, only MVR-steps, rewrite steps replace or delete only the symbol in the middle of the window, and only one rewrite step per cycle that is immediately followed by a restart.

Parameters of the Restarting Automaton

The type of a restarting automaton depends on many parameters:

- the number $|Q| \geq 1$ of states
- the number $|\Gamma \smallsetminus \Sigma| \geq 0$ of auxiliary letters
- the size $k \geq 1$ of the read/write window
- the type of move steps allowed
- the type of rewrite steps allowed
- the number $j \geq 1$ of allowed rewrite steps per cycle
- the positions at which rewrite steps are executed

The automaton from Example 2 satisfies $|Q| = 1$, $|\Gamma \smallsetminus \Sigma| = 6$, $k = 3$, only MVR-steps, rewrite steps replace or delete only the symbol in the middle of the window, and only one rewrite step per cycle that is immediately followed by a restart.

Remark 5

(a) *Each recursively enumerable language is accepted by a deterministic restarting automaton with window size 1 and MVR- and MVL-steps that executes a single rewrite step per cycle which replaces a single letter or inserts a single letter.*

(b) *Each context-sensitive language is accepted by a (nondeterministic) restarting automaton with window size 1 and MVR- and MVL-steps that executes a single rewrite step per cycle which replaces a single letter.*

Proof.

The given type of restarting automaton can simulate a single-tape Turing machine (linear-bounded automaton), where each step of the Turing machine (LBA) is simulated by several cycles. □

Definition 6

Let $M = (Q, \Sigma, \Gamma, \triangleright, \triangleleft, q_0, k, \delta)$ be a restarting automaton.

(a) *M is called* length-reducing *if $|v| < |u|$ holds for each rewrite step $(q', v) \in \delta(q, u)$ of M.*

(b) *M is called* weight-reducing *(or* shrinking*) if there exists a weight function $\omega : \Gamma \to \mathbb{N}_+$ such that $\omega(v) < \omega(u)$ holds for each rewrite step $(q', v) \in \delta(q, u)$ of M.*

Fact 7

If M is a weight-reducing restarting automaton, then each computation that begins with a tape contents of length n consists of at most $c \cdot n$ cycles and a tail, where $c \geq 1$ is a constant that depends on the underlying weight function.

Definition 6

Let $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, k, \delta)$ be a restarting automaton.

(a) *M is called* length-reducing *if $|v| < |u|$ holds for each rewrite step $(q', v) \in \delta(q, u)$ of M.*

(b) *M is called* weight-reducing *(or* shrinking*) if there exists a weight function $\omega : \Gamma \to \mathbb{N}_+$ such that $\omega(v) < \omega(u)$ holds for each rewrite step $(q', v) \in \delta(q, u)$ of M.*

Fact 7

If M is a weight-reducing restarting automaton, then each computation that begins with a tape contents of length n consists of at most $c \cdot n$ cycles and a tail, where $c \geq 1$ is a constant that depends on the underlying weight function.

Corollary 8

If M is a weight-reducing restarting automaton, then $L(M) \in \mathrm{NTIMESPACE}(n^2, n)$.

Proof.

A computation of M can be simulated by a nondeterministic 2-tape Turing machine that runs in quadratic time and that needs linear space. □

From now on we only consider restarting automata that are weight-reducing (or even length-reducing)!

The restarting automaton from Example 2 is weight-reducing: Choose the weight function ω in such a way that

$$\omega(a), \omega(b) > \omega(a_1), \omega(e), \omega(f) > \omega(e_1), \omega(f_1), \omega(f_2)$$

Corollary 8

If M is a weight-reducing restarting automaton, then $L(M) \in \mathrm{NTIMESPACE}(n^2, n)$.

Proof.

A computation of M can be simulated by a nondeterministic 2-tape Turing machine that runs in quadratic time and that needs linear space. □

From now on we only consider restarting automata that are weight-reducing (or even length-reducing)!

The restarting automaton from Example 2 is weight-reducing: Choose the weight function ω in such a way that

$$\omega(a), \omega(b) > \omega(a_1), \omega(e), \omega(f) > \omega(e_1), \omega(f_1), \omega(f_2)$$

Corollary 8

If M is a weight-reducing restarting automaton, then $L(M) \in \mathrm{NTIMESPACE}(n^2, n)$.

Proof.

A computation of M can be simulated by a nondeterministic 2-tape Turing machine that runs in quadratic time and that needs linear space. □

From now on we only consider restarting automata that are weight-reducing (or even length-reducing)!

The restarting automaton from Example 2 is weight-reducing: Choose the weight function ω in such a way that

$$\omega(a), \omega(b) > \omega(a_1), \omega(e), \omega(f) > \omega(e_1), \omega(f_1), \omega(f_2)$$

3. Some Important Results on Formal Languages

3.1 Parameter 1: The Type of Operations

In [1] and other papers by Petr Jančar et al., only length-reducing restarting automata were studied that execute exactly one rewrite step in each cycle. The resulting types of restarting automata are the following:

	with auxiliary symbols (-WW)	no auxiliary symbols (-W)	deletions only (-ε)
MVL-steps (RL-)	RLWW	RLW	RL
no MVL-steps (RR-)	RRWW	RRW	RR
no MVL-steps, rewrite followed by restart (R-)	RWW	RW	R

Definition 9

Let M be an RLWW*-automaton.*

(a) *If a cycle C of M contains the rewrite transition*

$$\triangleright uqvw\triangleleft \vdash_M \triangleright uxq'w\triangleleft,$$

then $D_r(C) = |vw| + 1$ is the right distance of C.

(b) *M is monotone if, in each computation of M, the right distance does not increase from one cycle to the next.*

(c) *M is weakly monotone if there exists a constant $c \geq 0$ such that, in each computation of M, the right distance increases by at most c from one cycle to the next.*

Prefixes are used to denote subtypes of restarting automata:

deterministic automata	det-
monotone automata	mon-
weakly monotone automata	wmon-
weight-reducing automata	s

Definition 9

Let M be an RLWW*-automaton.*

(a) *If a cycle C of M contains the rewrite transition*

$$\triangleright uqvw\triangleleft \vdash_M \triangleright uxq'w\triangleleft,$$

then $D_r(C) = |vw| + 1$ *is the right distance of C.*

(b) *M is monotone if, in each computation of M, the right distance does not increase from one cycle to the next.*

(c) *M is weakly monotone if there exists a constant* $c \geq 0$ *such that, in each computation of M, the right distance increases by at most c from one cycle to the next.*

Prefixes are used to denote subtypes of restarting automata:

deterministic automata	det-
monotone automata	mon-
weakly monotone automata	wmon-
weight-reducing automata	s

Theorem 10 (Plátek 2001 [3])

$$\mathcal{L}((\text{s})\text{RL}(\text{W})(\text{W})) = \mathcal{L}((\text{s})\text{RR}(\text{W})(\text{W})).$$
$$\mathcal{L}((\text{w})\text{mon-}(\text{s})\text{RL}(\text{W})(\text{W})) = \mathcal{L}((\text{w})\text{mon-}(\text{s})\text{RR}(\text{W})(\text{W})).$$

Theorem 11 (Jurdziński, Otto 2007 [4])

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

Theorem 12 (Jančar et al. 1999 [5])

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

Theorem 13 (Jurdziński et al. 2004 [6])

$\mathcal{L}(\text{wmon-RWW}) = \mathcal{L}(\text{wmon-RRWW}) = \text{GCSL}$.

Theorem 14 (Niemann, Otto 2000 [7])

$\mathcal{L}(\text{det-RWW}) = \mathcal{L}(\text{det-RRWW}) = \text{CRL}$.

Theorem 10 (Plátek 2001 [3])

$$\mathcal{L}((s)\text{RL}(W)(W)) = \mathcal{L}((s)\text{RR}(W)(W)).$$
$$\mathcal{L}((w)\text{mon-}(s)\text{RL}(W)(W)) = \mathcal{L}((w)\text{mon-}(s)\text{RR}(W)(W)).$$

Theorem 11 (Jurdziński, Otto 2007 [4])

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

Theorem 12 (Jančar et al. 1999 [5])

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

Theorem 13 (Jurdziński et al. 2004 [6])

$\mathcal{L}(\text{wmon-RWW}) = \mathcal{L}(\text{wmon-RRWW}) = \text{GCSL}$.

Theorem 14 (Niemann, Otto 2000 [7])

$\mathcal{L}(\text{det-RWW}) = \mathcal{L}(\text{det-RRWW}) = \text{CRL}$.

Theorem 10 (Plátek 2001 [3])

$$\mathcal{L}((\mathsf{s})\mathsf{RL}(\mathsf{W})(\mathsf{W})) = \mathcal{L}((\mathsf{s})\mathsf{RR}(\mathsf{W})(\mathsf{W})).$$
$$\mathcal{L}((\mathsf{w})\mathsf{mon}\text{-}(\mathsf{s})\mathsf{RL}(\mathsf{W})(\mathsf{W})) = \mathcal{L}((\mathsf{w})\mathsf{mon}\text{-}(\mathsf{s})\mathsf{RR}(\mathsf{W})(\mathsf{W})).$$

Theorem 11 (Jurdziński, Otto 2007 [4])

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

Theorem 12 (Jančar et al. 1999 [5])

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

Theorem 13 (Jurdziński et al. 2004 [6])

$\mathcal{L}(\mathsf{wmon\text{-}RWW}) = \mathcal{L}(\mathsf{wmon\text{-}RRWW}) = \mathsf{GCSL}$.

Theorem 14 (Niemann, Otto 2000 [7])

$\mathcal{L}(\mathsf{det\text{-}RWW}) = \mathcal{L}(\mathsf{det\text{-}RRWW}) = \mathsf{CRL}$.

Theorem 10 (Plátek 2001 [3])

$$\mathcal{L}((s)\mathsf{RL}(W)(W)) = \mathcal{L}((s)\mathsf{RR}(W)(W)).$$
$$\mathcal{L}((w)\text{mon-}(s)\mathsf{RL}(W)(W)) = \mathcal{L}((w)\text{mon-}(s)\mathsf{RR}(W)(W)).$$

Theorem 11 (Jurdziński, Otto 2007 [4])

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

Theorem 12 (Jančar et al. 1999 [5])

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

Theorem 13 (Jurdziński et al. 2004 [6])

$\mathcal{L}(\text{wmon-RWW}) = \mathcal{L}(\text{wmon-RRWW}) = \text{GCSL}$.

Theorem 14 (Niemann, Otto 2000 [7])

$\mathcal{L}(\text{det-RWW}) = \mathcal{L}(\text{det-RRWW}) = \text{CRL}$.

Theorem 10 (Plátek 2001 [3])

$$\begin{aligned}\mathcal{L}((\text{s})\text{RL}(\text{W})(\text{W})) &= \mathcal{L}((\text{s})\text{RR}(\text{W})(\text{W})).\\ \mathcal{L}((\text{w})\text{mon-}(\text{s})\text{RL}(\text{W})(\text{W})) &= \mathcal{L}((\text{w})\text{mon-}(\text{s})\text{RR}(\text{W})(\text{W})).\end{aligned}$$

Theorem 11 (Jurdziński, Otto 2007 [4])

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

Theorem 12 (Jančar et al. 1999 [5])

$$\begin{aligned}&\text{(a)}\quad \mathcal{L}(\text{mon-RWW}) = \mathcal{L}(\text{mon-RRWW}) = \text{CFL}.\\ &\text{(b)}\quad \mathcal{L}(\text{det-mon-R}) = \mathcal{L}(\text{det-mon-RRWW}) = \text{DCFL}.\end{aligned}$$

Theorem 13 (Jurdziński et al. 2004 [6])

$\mathcal{L}(\text{wmon-RWW}) = \mathcal{L}(\text{wmon-RRWW}) = \text{GCSL}$.

Theorem 14 (Niemann, Otto 2000 [7])

$\mathcal{L}(\text{det-RWW}) = \mathcal{L}(\text{det-RRWW}) = \text{CRL}$.

Theorem 15 (Otto 2009 [8])

$\mathcal{L}(\text{det-mon-RL}) = \mathcal{L}(\text{det-mon-RLWW}) = \text{LRR}.$

Theorems 12, 13, 14, and 15 extend to the corresponding weight-reducing types of restarting automata.

Hierarchy of Language Classes Accepted by Various Subtypes of sRLWW-Automata:

Theorem 15 (Otto 2009 [8])

$\mathcal{L}(\text{det-mon-RL}) = \mathcal{L}(\text{det-mon-RLWW}) = \text{LRR}.$

Theorems 12, 13, 14, and 15 extend to the corresponding weight-reducing types of restarting automata.

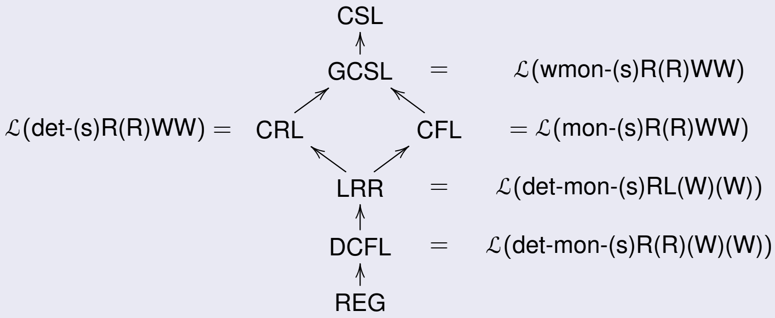
Hierarchy of Language Classes Accepted by Various Subtypes of sRLWW-Automata:

Theorem 15 (Otto 2009 [8])

$\mathcal{L}(\text{det-mon-RL}) = \mathcal{L}(\text{det-mon-RLWW}) = \text{LRR}.$

Theorems 12, 13, 14, and 15 extend to the corresponding weight-reducing types of restarting automata.

Hierarchy of Language Classes Accepted by Various Subtypes of sRLWW-Automata:



Problem 1

Is the inclusion $\mathcal{L}(\text{sRRWW}) \subseteq \text{CSL}$ *proper?*
Find a context-sensitive language that is not accepted by an sRRWW-automaton!

Problem 2

Is the inclusion $\mathcal{L}(\text{RRWW}) \subseteq \mathcal{L}(\text{sRRWW})$ *proper?*
Find a language that is accepted by a weight-reducing, but not by any length-reducing RRWW-automaton!

Problem 3

Is the inclusion $\mathcal{L}(\text{RWW}) \subseteq \mathcal{L}(\text{RRWW})$ *proper?*
Find a language that is accepted by an RRWW-automaton, but not by any RWW-automaton!

Problem 1

Is the inclusion $\mathcal{L}(\text{sRRWW}) \subseteq \text{CSL}$ *proper?*
Find a context-sensitive language that is not accepted by an sRRWW-automaton!

Problem 2

Is the inclusion $\mathcal{L}(\text{RRWW}) \subseteq \mathcal{L}(\text{sRRWW})$ *proper?*
Find a language that is accepted by a weight-reducing, but not by any length-reducing RRWW-automaton!

Problem 3

Is the inclusion $\mathcal{L}(\text{RWW}) \subseteq \mathcal{L}(\text{RRWW})$ *proper?*
Find a language that is accepted by an RRWW-automaton, but not by any RWW-automaton!

Problem 1

Is the inclusion $\mathcal{L}$(sRRWW) $\subseteq$ CSL *proper?*
Find a context-sensitive language that is not accepted by an sRRWW-automaton!

Problem 2

Is the inclusion $\mathcal{L}$(RRWW) $\subseteq$ $\mathcal{L}$(sRRWW) *proper?*
Find a language that is accepted by a weight-reducing, but not by any length-reducing RRWW-automaton!

Problem 3

Is the inclusion $\mathcal{L}$(RWW) $\subseteq$ $\mathcal{L}$(RRWW) *proper?*
Find a language that is accepted by an RRWW-automaton, but not by any RWW-automaton!

Theorem 16 (Holzer, Kutrib, Reimann 2007 [9])

Let $\mathsf{X}, \mathsf{Y} \in \{\mathsf{R}, \mathsf{RR}, \mathsf{RW}, \mathsf{RRW}, \mathsf{RWW}, \mathsf{RRWW}\}$. *Then the trade-off for the conversion of a restarting automaton of type* X *or* det-X *into an equivalent monotone restarting automaton of type* Y *is non-recursive.*

Theorem 17 (Holzer, Kutrib, Reimann 2007 [9])

The trade-off for the conversion of a monotone R(R)WW-automaton into an equivalent deterministic R(R)W- or R(R)-automaton is non-recursive.

Theorem 16 (Holzer, Kutrib, Reimann 2007 [9])

Let $\mathsf{X}, \mathsf{Y} \in \{\mathsf{R}, \mathsf{RR}, \mathsf{RW}, \mathsf{RRW}, \mathsf{RWW}, \mathsf{RRWW}\}$. *Then the trade-off for the conversion of a restarting automaton of type* X *or* det-X *into an equivalent monotone restarting automaton of type* Y *is non-recursive.*

Theorem 17 (Holzer, Kutrib, Reimann 2007 [9])

The trade-off for the conversion of a monotone R(R)WW-automaton into an equivalent deterministic R(R)W- or R(R)-automaton is non-recursive.

3.2 Parameter 2: The Size of the Read/Write Window

For each type X of restarting automaton, let X(k) denote the restarting automata of type X that have a read/write window of size k.

Theorem 18 (Mráz 2001 [11])

For all $k \geq 1$ *and all* $\mathrm{X} \in \{\mathrm{R}, \mathrm{RR}, \mathrm{RW}, \mathrm{RRW}\}$,

$$\mathcal{L}((\text{det-})(\text{mon-})\mathrm{X}(k)) \subsetneq \mathcal{L}((\text{det-})(\text{mon-})\mathrm{X}(k+1)).$$

Theorem 19 (Kutrib, Otto 2012 [12])

For all $\mathrm{X} \in \{\text{det-R(R)W}, \mathrm{R(R)W}\}$ *and all* $k \geq 3$, *the trade-off from* $\mathrm{X}(k)$*-automata to* $\mathrm{X}(k-1)$*-automata is non-recursive.*

Theorem 20 (Kutrib, Otto 2013 [13])

For all $\mathrm{X} \in \{\text{det-R}, \text{det-RR}, \mathrm{R}, \mathrm{RR}\}$ *and all* $k \geq 5$, *the trade-off from* $\mathrm{X}(k)$*-automata to* $\mathrm{X}(k-1)$*-automata is non-recursive.*

3.2 Parameter 2: The Size of the Read/Write Window

For each type X of restarting automaton, let X(k) denote the restarting automata of type X that have a read/write window of size k.

Theorem 18 (Mráz 2001 [11])

For all $k \geq 1$ *and all* $\mathrm{X} \in \{\mathrm{R}, \mathrm{RR}, \mathrm{RW}, \mathrm{RRW}\}$,

$$\mathcal{L}((\text{det-})(\text{mon-})\mathrm{X}(k)) \subsetneq \mathcal{L}((\text{det-})(\text{mon-})\mathrm{X}(k+1)).$$

Theorem 19 (Kutrib, Otto 2012 [12])

For all $\mathrm{X} \in \{\text{det-R(R)W}, \text{R(R)W}\}$ *and all* $k \geq 3$*, the trade-off from* $\mathrm{X}(k)$*-automata to* $\mathrm{X}(k-1)$*-automata is non-recursive.*

Theorem 20 (Kutrib, Otto 2013 [13])

For all $\mathrm{X} \in \{\text{det-R}, \text{det-RR}, \mathrm{R}, \mathrm{RR}\}$ *and all* $k \geq 5$*, the trade-off from* $\mathrm{X}(k)$*-automata to* $\mathrm{X}(k-1)$*-automata is non-recursive.*

3.2 Parameter 2: The Size of the Read/Write Window

For each type X of restarting automaton, let X(k) denote the restarting automata of type X that have a read/write window of size k.

Theorem 18 (Mráz 2001 [11])

For all $k \geq 1$ and all $\mathrm{X} \in \{\mathrm{R}, \mathrm{RR}, \mathrm{RW}, \mathrm{RRW}\}$,

$$\mathcal{L}((\text{det-})(\text{mon-})\mathrm{X}(k)) \subsetneq \mathcal{L}((\text{det-})(\text{mon-})\mathrm{X}(k+1)).$$

Theorem 19 (Kutrib, Otto 2012 [12])

For all $\mathrm{X} \in \{\text{det-R(R)W}, \text{R(R)W}\}$ and all $k \geq 3$, the trade-off from $\mathrm{X}(k)$-automata to $\mathrm{X}(k-1)$-automata is non-recursive.

Theorem 20 (Kutrib, Otto 2013 [13])

For all $\mathrm{X} \in \{\text{det-R}, \text{det-RR}, \mathrm{R}, \mathrm{RR}\}$ and all $k \geq 5$, the trade-off from $\mathrm{X}(k)$-automata to $\mathrm{X}(k-1)$-automata is non-recursive.

Theorem 21 (Mráz 2001 [11], Kutrib, Reimann 2008 [14])

(a) $\mathcal{L}((\text{det-})(\text{mon-})\text{R}(1)) = \text{REG}.$

(b) $\mathcal{L}(\text{det-}(\text{mon-})\text{RR}(1)) = \text{REG} \subsetneq \mathcal{L}(\text{mon-RR}(1)).$

Theorem 22 (Mráz, Otto 2019 [15])

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

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

(c) $\mathcal{L}(\text{mon-RR}(1)) \subsetneq \mathcal{L}(\text{mon-sRRWW}(1)).$

Problem 4

Is the inclusion $\mathcal{L}(\text{mon-sRRWW}(1)) \subseteq \text{CFL}$ *proper?*
Find a context-free language that is not accepted by any monotone sRRWW-automaton with window size one!
Is $L_{\text{pal}} = \{ ww^R \mid w \in \{a,b\}^* \}$ *such a language?*

Theorem 21 (Mráz 2001 [11], Kutrib, Reimann 2008 [14])

(a) $\mathcal{L}(\text{(det-)(mon-)R(1)}) = \text{REG}.$

(b) $\mathcal{L}(\text{det-(mon-)RR(1)}) = \text{REG} \subsetneq \mathcal{L}(\text{mon-RR(1)}).$

Theorem 22 (Mráz, Otto 2019 [15])

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

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

(c) $\mathcal{L}(\text{mon-RR(1)}) \subsetneq \mathcal{L}(\text{mon-sRRWW(1)}).$

Problem 4

Is the inclusion $\mathcal{L}(\text{mon-sRRWW(1)}) \subseteq \text{CFL}$ proper?
Find a context-free language that is not accepted by any monotone sRRWW-automaton with window size one!
Is $L_{\text{pal}} = \{ ww^R \mid w \in \{a,b\}^* \}$ such a language?

Theorem 21 (Mráz 2001 [11], Kutrib, Reimann 2008 [14])

(a) $\mathcal{L}$((det-)(mon-)R(1)) $=$ REG.
(b) $\mathcal{L}$(det-(mon-)RR(1)) $=$ REG $\subsetneq$ $\mathcal{L}$(mon-RR(1)).

Theorem 22 (Mráz, Otto 2019 [15])

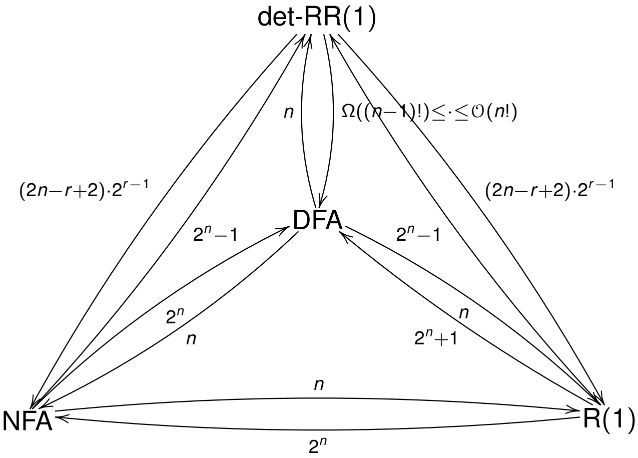
(a) $\mathcal{L}$(det-sRWW(1)) $=$ $\mathcal{L}$(mon-sRWW(1)) $=$ REG.
(b) $\mathcal{L}$(det-sRRWW(1)) $=$ REG.
(c) $\mathcal{L}$(mon-RR(1)) $\subsetneq$ $\mathcal{L}$(mon-sRRWW(1)).

Problem 4

Is the inclusion $\mathcal{L}$(mon-sRRWW(1)) $\subseteq$ CFL proper?
Find a context-free language that is not accepted by any monotone sRRWW-automaton with window size one!
Is $L_{\mathrm{pal}} = \{ ww^R \mid w \in \{a, b\}^* \}$ such a language?

Theorem 23 (Kutrib, Reimann 2008 [16])

The trade-offs between DFAs, NFAs, R(1)- , and det-RR(1)-automata are as depicted below.



Theorem 24 (Schluter 2015 [17])

For all $k \geq 2$,

(a) $\mathcal{L}(\text{mon-RWW}(k)) = \mathcal{L}(\text{mon-RRWW}(k))$.

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

(c) $\mathcal{L}(\text{mon-RRWW}(k+1)) = \mathcal{L}(\text{mon-RRWW}(k))$.

(d) $\mathcal{L}(\text{RRWW}(k+1)) = \mathcal{L}(\text{RRWW}(k))$.

Corollary 25 (Schluter 2015 [17])

$\mathcal{L}(\text{mon-RWW}(2)) = \text{CFL} = \mathcal{L}(\text{mon-sRRWW})$.

Theorem 24 (Schluter 2015 [17])

For all $k \geq 2$,

$$\begin{array}{lll} \text{(a)} & \mathcal{L}(\text{mon-RWW}(k)) & = & \mathcal{L}(\text{mon-RRWW}(k)). \\ \text{(b)} & \mathcal{L}(\text{det-mon-RWW}(k)) & = & \mathcal{L}(\text{det-mon-RRWW}(k)). \\ \text{(c)} & \mathcal{L}(\text{mon-RRWW}(k+1)) & = & \mathcal{L}(\text{mon-RRWW}(k)). \\ \text{(d)} & \mathcal{L}(\text{RRWW}(k+1)) & = & \mathcal{L}(\text{RRWW}(k)). \end{array}$$

Corollary 25 (Schluter 2015 [17])

$\mathcal{L}(\text{mon-RWW}(2)) = \text{CFL} = \mathcal{L}(\text{mon-sRRWW})$.

Theorem 26 (Mráz, Otto 2019 [18])

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

As $L_{\text{expo}} = \{ a^{2^n} \mid n \geq 0 \} \in \mathcal{L}$(det-RWW(3)), we see that $\mathcal{L}$(det-RWW(2)) $\subsetneq \mathcal{L}$(det-RWW(3)).

Problem 5

Is the inclusion $\mathcal{L}$(det-RWW(k)) $\subseteq \mathcal{L}$(det-RWW($k+1$)) proper for all $k \geq 3$, or is there an integer $\hat{k} \geq 3$ such that $\mathcal{L}$(det-RWW($\hat{k}$)) $=$ CRL?

Theorem 27 (Mráz, Otto 2019 [18])

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

Theorem 26 (Mráz, Otto 2019 [18])

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

As $L_{\text{expo}} = \{ a^{2^n} \mid n \geq 0 \} \in \mathcal{L}$(det-RWW(3)), we see that $\mathcal{L}$(det-RWW(2)) $\subsetneq \mathcal{L}$(det-RWW(3)).

Problem 5

Is the inclusion $\mathcal{L}$(det-RWW(k)) $\subseteq \mathcal{L}$(det-RWW($k+1$)) proper for all $k \geq 3$, or is there an integer $\hat{k} \geq 3$ such that $\mathcal{L}$(det-RWW($\hat{k}$)) = CRL?

Theorem 27 (Mráz, Otto 2019 [18])

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

Theorem 26 (Mráz, Otto 2019 [18])

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

As $L_{\text{expo}} = \{ a^{2^n} \mid n \geq 0 \} \in \mathcal{L}(\text{det-RWW}(3))$, we see that $\mathcal{L}(\text{det-RWW}(2)) \subsetneq \mathcal{L}(\text{det-RWW}(3))$.

Problem 5

Is the inclusion $\mathcal{L}(\text{det-RWW}(k)) \subseteq \mathcal{L}(\text{det-RWW}(k+1))$ proper for all $k \geq 3$, or is there an integer $\hat{k} \geq 3$ such that $\mathcal{L}(\text{det-RWW}(\hat{k})) = \text{CRL}$?

Theorem 27 (Mráz, Otto 2019 [18])

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

3.3 Parameter 3: The Number of States

Definition 28

A restarting automaton $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, k, \delta)$ *is called stateless if* $Q = \{q_0\}$. *We use the prefix stl- to denote stateless types of restarting automata.*

Theorem 29 (Kutrib, Messerschmidt, Otto 2010 [19])

(a)	$\mathcal{L}$(stl-det-mon-R)	$\supsetneq$	REG
(b)	$\mathcal{L}$(stl-det-mon-RWW)	$=$	DCFL
(c)	$\mathcal{L}$(stl-mon-RWW)	$=$	CFL
(d)	$\mathcal{L}$(stl-det-RWW)	$=$	CRL
(e)	$\mathcal{L}$(stl-wmon-RWW)	$=$	GCSL

3.3 Parameter 3: The Number of States

Definition 28

A restarting automaton $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, k, \delta)$ is called stateless if $Q = \{q_0\}$. We use the prefix stl- to denote stateless types of restarting automata.

Theorem 29 (Kutrib, Messerschmidt, Otto 2010 [19])

(a)	$\mathcal{L}$(stl-det-mon-R)	$\supsetneq$	REG
(b)	$\mathcal{L}$(stl-det-mon-RWW)	$=$	DCFL
(c)	$\mathcal{L}$(stl-mon-RWW)	$=$	CFL
(d)	$\mathcal{L}$(stl-det-RWW)	$=$	CRL
(e)	$\mathcal{L}$(stl-wmon-RWW)	$=$	GCSL

Definition 30

An ordered RWW-automaton (ORWW) is an sRWW(3)-automaton in which each rewrite step replaces the letter in the middle of the window by a letter with less weight.

Theorem 31 (Kwee, Otto 2018 [20, 21])

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

(b) $\mathcal{L}(\text{stl-ORWW}) = \text{REG} \subsetneq \mathcal{L}(\text{ORWW})$

(c) $\mathcal{L}(\text{ORWW})$ *is incomparable to* DLIN *and* GCSL.

Proof of (c):

$\{ a^n b^n \mid n \geq 1 \} \in \text{DLIN} \smallsetminus \mathcal{L}(\text{ORWW})$, while

$$L'_{\text{copy}} = \{ wcu \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ is a scattered factor of } w \}$$

belongs to $\mathcal{L}(\text{ORWW}) \smallsetminus \text{GCSL}$. □

Definition 30

An ordered RWW-automaton (ORWW) is an sRWW(3)-automaton in which each rewrite step replaces the letter in the middle of the window by a letter with less weight.

Theorem 31 (Kwee, Otto 2018 [20, 21])

(a)	$\mathcal{L}$(stl-det-ORWW)	$=$	$\mathcal{L}$(det-ORWW)	$=$	REG
(b)	$\mathcal{L}$(stl-ORWW)	$=$	REG	$\subsetneq$	$\mathcal{L}$(ORWW)
(c)	$\mathcal{L}$(ORWW) *is incomparable to* DLIN *and* GCSL.				

Proof of (c):

$\{ a^n b^n \mid n \geq 1 \} \in \text{DLIN} \smallsetminus \mathcal{L}(\text{ORWW})$, while

$$L'_{\text{copy}} = \{ wcu \mid w, u \in \{a,b\}^*, |w|, |u| \geq 2, u \text{ is a scattered factor of } w \}$$

belongs to $\mathcal{L}(\text{ORWW}) \smallsetminus \text{GCSL}$. □

Definition 30

An ordered RWW-automaton (ORWW) is an sRWW(3)-automaton in which each rewrite step replaces the letter in the middle of the window by a letter with less weight.

Theorem 31 (Kwee, Otto 2018 [20, 21])

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

(b) $\mathcal{L}(\text{stl-ORWW}) = \text{REG} \subsetneq \mathcal{L}(\text{ORWW})$

(c) $\mathcal{L}(\text{ORWW})$ *is incomparable to* DLIN *and* GCSL.

Proof of (c):

$\{ a^n b^n \mid n \geq 1 \} \in \text{DLIN} \smallsetminus \mathcal{L}(\text{ORWW})$, while

$$L'_{\text{copy}} = \{ wcu \mid w, u \in \{a, b\}^*, |w|, |u| \geq 2, u \text{ is a scattered factor of } w \}$$

belongs to $\mathcal{L}(\text{ORWW}) \smallsetminus \text{GCSL}$. □

Remark

For a stl-det-ORWW-automaton $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, k, \delta)$, we use $|\Gamma|$ to measure the size of M.

Theorem 32 (Kwee, Otto 2018 [21])

From a stl-ORWW-automaton M with a tape alphabet of cardinality n, an NFA $A = (Q, \Sigma, S, \delta_A, F)$ can be constructed such that $|Q| \leq 2^{13 \cdot n}$ and $L(A) = L(M)$.

Corollary 33 (Kwee, Otto 2018 [21])

For converting a stl-ORWW-automaton or a stl-det-ORWW-automaton with a tape alphabet of cardinality n into an equivalent DFA (NFA), $2^{2^{\mathcal{O}(n)}}$ ($2^{\mathcal{O}(n)}$) states are sufficient, and there are cases in which these many states are also necessary.

Remark

For a stl-det-ORWW-automaton $M = (Q, \Sigma, \Gamma, \triangleright, \triangleleft, q_0, k, \delta)$, we use $|\Gamma|$ to measure the size of M.

Theorem 32 (Kwee, Otto 2018 [21])

From a stl-ORWW-automaton M with a tape alphabet of cardinality n, an NFA $A = (Q, \Sigma, S, \delta_A, F)$ can be constructed such that $|Q| \leq 2^{13 \cdot n}$ and $L(A) = L(M)$.

Corollary 33 (Kwee, Otto 2018 [21])

For converting a stl-ORWW-automaton or a stl-det-ORWW-automaton with a tape alphabet of cardinality n into an equivalent DFA (NFA), $2^{2^{\mathcal{O}(n)}}$ ($2^{\mathcal{O}(n)}$) states are sufficient, and there are cases in which these many states are also necessary.

Remark

For a stl-det-ORWW-automaton $M = (Q, \Sigma, \Gamma, \triangleright, \triangleleft, q_0, k, \delta)$, we use $|\Gamma|$ to measure the size of M.

Theorem 32 (Kwee, Otto 2018 [21])

From a stl-ORWW-automaton M with a tape alphabet of cardinality n, an NFA $A = (Q, \Sigma, S, \delta_A, F)$ can be constructed such that $|Q| \leq 2^{13 \cdot n}$ and $L(A) = L(M)$.

Corollary 33 (Kwee, Otto 2018 [21])

For converting a stl-ORWW-automaton or a stl-det-ORWW-automaton with a tape alphabet of cardinality n into an equivalent DFA (NFA), $2^{2^{\mathcal{O}(n)}}$ ($2^{\mathcal{O}(n)}$) states are sufficient, and there are cases in which these many states are also necessary.

Definition 34

An ordered restart-delete-automaton (ORD) is an sRWW(3)-automaton in which each rewrite step replaces the letter in the middle of the window by a letter with less weight or by the empty word.

Theorem 35 (Otto 2018 [22, 23])

(a) $\mathrm{DCFL} \subsetneq \mathcal{L}(\text{stl-det-ORD}) = \mathcal{L}(\text{det-ORD}) \subsetneq \mathrm{CRL} \cap \mathrm{CFL}$

(b) $\mathrm{CFL} \subseteq \mathcal{L}(\text{stl-ORD})$

Problem 6

Is the inclusion $\mathrm{CFL} \subseteq \mathcal{L}(\text{stl-ORD})$ *proper?*
Find a non-context-free language that accepted by a stl-ORD-automaton!

Definition 34

An ordered restart-delete-automaton (ORD) is an sRWW(3)-automaton in which each rewrite step replaces the letter in the middle of the window by a letter with less weight or by the empty word.

Theorem 35 (Otto 2018 [22, 23])

(a) $\mathsf{DCFL} \subsetneq \mathcal{L}(\text{stl-det-ORD}) = \mathcal{L}(\text{det-ORD}) \subsetneq \mathsf{CRL} \cap \mathsf{CFL}$

(b) $\mathsf{CFL} \subseteq \mathcal{L}(\text{stl-ORD})$

Problem 6

Is the inclusion $\mathsf{CFL} \subseteq \mathcal{L}(\text{stl-ORD})$ *proper?*
Find a non-context-free language that accepted by a stl-ORD-automaton!

Definition 34

An ordered restart-delete-automaton (ORD) is an sRWW(3)-automaton in which each rewrite step replaces the letter in the middle of the window by a letter with less weight or by the empty word.

Theorem 35 (Otto 2018 [22, 23])

(a) $\mathsf{DCFL} \subsetneq \mathcal{L}(\text{stl-det-ORD}) = \mathcal{L}(\text{det-ORD}) \subsetneq \mathsf{CRL} \cap \mathsf{CFL}$

(b) $\mathsf{CFL} \subseteq \mathcal{L}(\text{stl-ORD})$

Problem 6

Is the inclusion $\mathsf{CFL} \subseteq \mathcal{L}(\text{stl-ORD})$ *proper?*
Find a non-context-free language that accepted by a stl-ORD-automaton!

Definition 36

A stl-ORD-automaton $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, k, \delta)$ is called swift, if it can always perform move-right steps unless its window is already at the right end of the tape.

Theorem 37 (Otto 2020 [23, 24])

$\mathcal{L}$(swift-ORD) = CFL

Theorem 38 (Otto 2019 [23, 24])

The trade-off for the conversion of a swift-ORD-automaton into an equivalent stl-ORWW-automaton is non-recursive.

Definition 36

A stl-ORD-automaton $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, k, \delta)$ is called swift, if it can always perform move-right steps unless its window is already at the right end of the tape.

Theorem 37 (Otto 2020 [23, 24])

$\mathcal{L}$(swift-ORD) = CFL

Theorem 38 (Otto 2019 [23, 24])

The trade-off for the conversion of a swift-ORD-automaton into an equivalent stl-ORWW-automaton is non-recursive.

Definition 36

A stl-ORD-automaton $M = (Q, \Sigma, \Gamma, \rhd, \lhd, q_0, k, \delta)$ is called swift, if it can always perform move-right steps unless its window is already at the right end of the tape.

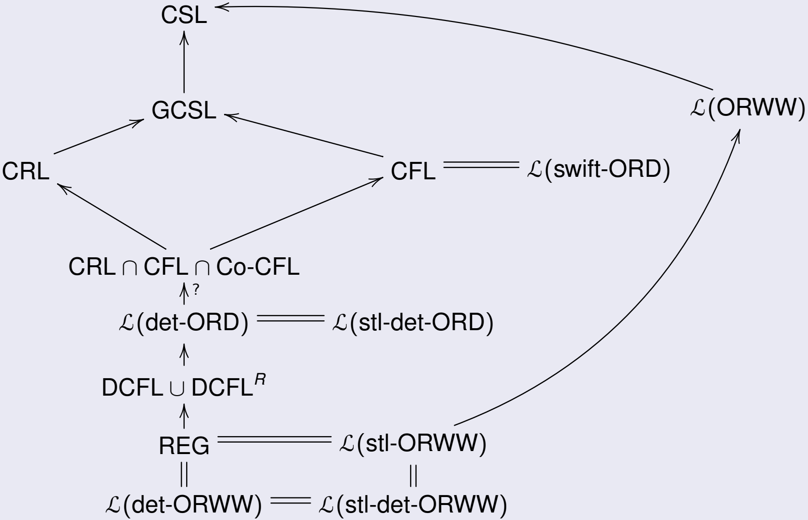
Theorem 37 (Otto 2020 [23, 24])

$\mathcal{L}(\text{swift-ORD}) = \text{CFL}$

Theorem 38 (Otto 2019 [23, 24])

The trade-off for the conversion of a swift-ORD-automaton into an equivalent stl-ORWW-automaton is non-recursive.

The Taxonomy of Ordered Restarting Automata



4. Conclusion

Further Parameters of Restarting Automata:

1. number of auxiliary letters (Jurdziński, Otto 2006 [25])
2. utilization of auxiliary letters:
   - lexicalized restarting automata (Mráz, Otto, Plátek 2009 [26])
   - h-lexicalized restarting automata (Plátek, Otto 2017 [27])
3. number of rewrites per cycle (Mráz, Otto, Plátek 2014 [28])

Further Variants of Restarting Automata

1. nonforgetting restarting automata [29]
2. cooperating distributed systems of restarting automata [30]
3. parallel communicating systems of restarting automata [31]
4. restarting transducer [32]
5. weighted restarting automata [33]
6. restarting automata with structured output [34]
7. restarting automata for picture languages [35]
8. restarting tree automata [36]

Another Topic:

- Inductive inference of restricted types of restarting automata (See, e.g., Mráz, Plátek, Otto 2006 [37], and Hoffmann 2013 [38])

Further Variants of Restarting Automata

1. nonforgetting restarting automata [29]
2. cooperating distributed systems of restarting automata [30]
3. parallel communicating systems of restarting automata [31]
4. restarting transducer [32]
5. weighted restarting automata [33]
6. restarting automata with structured output [34]
7. restarting automata for picture languages [35]
8. restarting tree automata [36]

Another Topic:

- Inductive inference of restricted types of restarting automata (See, e.g., Mráz, Plátek, Otto 2006 [37], and Hoffmann 2013 [38])

Thank you for your attention!

6. Bibliography

[1] Jančar, P., Mráz, F., Plátek, M., Vogel, J.: Restarting automata. In: Reichel, H. (ed.), FCT'95, Proc., *Lecture Notes in Computer Science 965*, pp. 283–292, Springer, Heidelberg (1995)

[2] Plátek, M., Lopatková, M., Oliva, K.: Restarting automata: motivations and applications. In: Holzer, M. (ed.), Workshop "Petrinets" und 13. Theorietag "Automaten und Formale Sprachen, Proc., pp. 90–96, Institut für Informatik, Technische Universität München (2003)

[3] Plátek, M.: Two-way restarting automata and j-monotonicity. In: L. Pacholski, L., Ružička, P. (eds.), *SOFSEM 2001, Proc.*, *Lecture Notes in Computer Science 2234*, Springer, Berlin, 2001, 316–325.

[4] Jurdziński, T., Otto,F.: Shrinking restarting automata. *International Journal of Foundations of Computer Science* 18 (2007) 361–385.

[5] Jančar, P., Mráz, F., Plátek, M., Vogel, J.: On monotonic automata with a restart operation. *Journal of Automata, Languages and Combinatorics* 4 (1999) 287–311.

[6] Jurdziński, T., Loryś, K., Niemann, G., Otto, F.: Some results on RWW- and RRWW-automata and their relation to the class of growing context-sensitive languages, *Journal of Automata, Languages and Combinatorics* 9 (2004) 407–437.

[7] Niemann, G., Otto, F.: Restarting automata, Church-Rosser languages, and representations of r.e. languages. In: Rozenberg, G., Thomas, W. (eds.), *DLT 1999, Proc.*, World Scientific, Singapore, 2000, 103–114.

[8] Otto, F.: Left-to-right regular languages and two-way restarting automata. *RAIRO - Theoretical Informatics and Applications* 43 (2009) 653–665.

[9] Holzer, M., Kutrib, M., Reimann, J.: Non-Recursive Trade-Offs for Deterministic Restarting Automata. *Journal of Automata, Languages and Combinatorics* 12 (2007) 195–213.

[10] Reimann, J.:
Beschreibungskomplexität von Restart-Automaten.
Doctoral dissertation, University of Giessen (2007),
http://geb.uni-giessen.de/geb/volltexte/2007/4618/index.html

[11] Mráz, F.: Lookahead hierarchies of restarting automata. *Journal of Automata, Languages and Combinatorics* 6 (2001) 493–506.

[12] Kutrib, M., Otto, F.: On the descriptional complexity of the window size for deterministic restarting automata. In: Moreira, N., Reis, R. (eds.), *CIAA 2012, Proc.*, *Lecture Notes in Computer Science 7381*, Springer, Heidelberg, 2012, 253–264.

[13] Kutrib, M., Otto, F.: On the descriptional complexity of the window size for deleting restarting automata. *International Journal of Foundations of Computer Science* 24 (2013) 831–846.

[14] Kutrib, M., Reimann, J.: Succinct description of regular languages by weak restarting automata. *Information and Computation* 206 (2008) 1152–1160.

[15] Mráz, F., Otto, F.: On shrinking restarting automata of window size one and two. In: Hofman, P., Skrzypczak, M. (eds.), *DLT 2019, Proc.*, *Lecture Notes in Computer Science 11647*, Springer, Heidelberg, 2019, 140–153.

[16] Kutrib, M., Reimann, J.: Optimal Simulations of Weak Restarting Automata. *International Journal of Foundations of Computer Science* 19 (2008) 795–811.

[17] Schluter, N.: Restarting automata with auxiliary symbols restricted by lookahead size. *International Journal of Computer Mathematics* 92 (2015) 908–938.

[18] Mráz, F., Otto, F.: Window size two suffices for deterministic monotone RWW-automata. In: Freund, R., Holzer, M., Sempere, J.M. (eds.), *NCMA 2019, Proc.*, books@ocg.at, vol. 336, Österreichische Computer Gesellschaft, Wien, 2019, 139–154.

[19] Kutrib, M., Messerschmidt, H., Otto, F.: On stateless two-pushdown automata and restarting automata. *International Journal of Foundations of Computer Science* 21 (2010) 781–798.

[20] Otto, F., Kwee, K.: On the descriptional complexity of stateless deterministic ordered restarting automata. *Information and Computation* 259 (2018) 277–302.

[21] Kwee, K., Otto, F.: Nondeterministic ordered restarting automata. *International Journal of Foundations of Computer Science* 29 (2018) 663–685.

[22] Otto, F.: On deterministic ordered restart-delete automata. *Theoretical Computer Science* 795 (2019) 257–274.

[23] Otto, F.: A characterization of the context-free languages by stateless ordered restart-delete automata. In: Chatzigeorgiou, A., et al. (eds.), *SOFSEM 2020, Proc.*, *Lecture Notes in Computer Science 12011*, Springer, Heidelberg, 2020, 39–50.

[24] Otto, F.: On the expressive power of stateless ordered restart-delete automata. *Theory of Computing Systems* 65 (2021) 1033–1068.

[25] Jurdziński, T., Otto, F.: Restarting automata with restricted utilization of auxiliary symbols. *Theoretical Computer Science* 363 (2006) 162–181.

[26] Mráz, F., Otto, F., Plátek, M.: The degree of word-expansion of lexicalized RRWW-automata - A new measure for the degree of nondeterminism of (context-free) languages. *Theoretical Computer Science* 410 (2009) 3530–3538.

[27] Plátek, M., Otto, F.: On h-lexicalized restarting automata. In: Csuhaj-Varjú, E., et al. (eds.), *AFL 2017, Proc., EPTCS* 252 (2017) 219–233.

[28] Mráz, F., Otto, F., Plátek, M.: Free word-order and restarting automata. *Fundamenta Informaticae* 133 (2014) 399–419.

[29] Messerschmidt, H., Otto, F.: On nonforgetting restarting automata that are deterministic and/or monotone. In: Grigoriev, D., Harrison, J., Hirsch, E.A. (eds.), *CSR 2006, Proc.*, *Lecture Notes in Computer Science 3967*, Springer, Heidelberg, 2006, 247–258.

[30] Messerschmidt, H., Otto, F.: Cooperating distributed systems of restarting automata. *International Journal of Foundations of Computer Science* 18 (2007) 1333–1342.

[31] Vollweiler, M., Otto, F.: Systems of parallel communicating restarting automata. In: Freund, R., et al. (eds.), *NCMA 2012, Proc.*, books@ocg.at, vol. 290, OCG, Vienna, 2012, 197–212.

[32] Hundeshagen, N., Otto, F.: Characterizing the rational functions by restarting transducers. In: Dediu, A.-H., Martín-Vide, C. (eds.), *LATA 2012, Proc.*, *Lecture Notes in Computer Science 7183*, Springer, Heidelberg, 2012, 325–336.

[33] Wang, Q., Otto, F.: Weighted restarting automata. *Soft Computing* 22 (2018) 1067–1083.

[34] Plátek, M., Mráz, F., Lopatkivá, M.: Restarting automata with structured output and functional generative description. In: Dediu, A.-H., Fernau, H., Martín-Vide, C. (eds.), *LATA 2010, Proc.*, *Lecture Notes in Computer Science 6031*, Springer, Heidelberg, 2010, 500–511.

[35] Otto, F.: Restarting automata for picture languages: A survey on recent developments. In: Holzer, M., Kutrib, M. (eds.), *CIAA 2014, Proc.*, *Lecture Notes in Computer Science 8587*, Springer, Heidelberg, 2014, 16–41.

[36] Stamer, H., Otto, F.: Restarting tree automata. In: van Leeuwen, J., et al. (eds.), *SOFSEM 2007, Proc.*, *Lecture Notes in Computer Science 4362*, Springer, Heidelberg, 2007, 510–521.

[37] Mráz, F., Otto, F., Plátek, M.: Learning analysis by reduction from positive data. In: Sakakibara, Y., et al. (eds.), *ICGI 2006, Proc.*, *Lecture Notes in Computer Science 4201*, Springer, Heidelberg, 2006, 125–136.

[38] Hoffmann, P.: Machine Learning of Analysis by Reduction. *Doctoral Thesis*, Department of Software and Computer Science Education, Faculty of Mathematics and Physics, Charles University in Prague, 2013.