A Taxonomy of Restarting Automata

Friedrich Otto

Fachbereich Elektrotechnik/Informatik, Universität Kassel
D-34109 Kassel

ABCDays on List Automata, Forgetting and Restarting Automata,
Prague, March 27–29, 2009
Outline:

1. Introduction: Analysis by Reduction
2. Definitions and Examples
3. Restarting Automata and the Chomsky Hierarchy
4. Nondeterministic Restarting Automata
5. Monotone Deterministic Two-Way Restarting Automata
6. Freely Rewriting Restarting Automata
7. Open Problems
1. Introduction: Analysis by Reduction

**Linguistic task:** Analyze sentences of a natural language
- verify syntactic correctness
- determine dependencies
- disambiguate between morphological ambiguities

**A linguistic technique:** Analysis by reduction

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

F. Otto
Operations used in the Analysis by Reduction:

**Move-right:** MVR

**Delete/Rewrite:** replace a small part of the sentence

**Restart:** start anew on a simplified sentence


**Restarting Automaton**

<table>
<thead>
<tr>
<th>$c$</th>
<th></th>
<th></th>
<th></th>
<th>(\ldots)</th>
<th></th>
<th>$$)</th>
</tr>
</thead>
</table>

- **Flexible tape**
- **Read/write-window**
- **Finite control**
A restarting automaton is described by an 8-tuple

\[ M = (Q, \Sigma, \Gamma, \phi, $, q_0, k, \delta) : \]

- \( Q \): finite set of states
- \( \Sigma \): finite input alphabet
- \( \Gamma \): finite tape alphabet, \( \Gamma \supseteq \Sigma \)
- \( \phi, $ \): left and right delimiters (\( \phi, $ \not\in \Gamma \))
- \( q_0 \): initial state from \( Q \)
- \( k \): size of the read/write window (\( k \geq 1 \))
- \( \delta \): \( Q \times \mathcal{P}C^{(k)} \to P_{\text{fin}} \) (operations)

where

\[ \mathcal{P}C^{(k)} := (\phi \cdot \Gamma^{k-1}) \cup \Gamma^k \cup (\Gamma \leq k-1 \cdot $) \cup (\phi \cdot \Gamma \leq k-2 \cdot $) \]
Transition steps

(1.) **MVR-step**: \((q', MVR) \in \delta(q, u)\) :

\[
\begin{array}{cccccc}
  & a & b & c & d & e & f \\
  \downarrow \quad q \\
\end{array} \quad \Rightarrow \quad \begin{array}{cccccc}
  & a & b & c & d & e & f \\
  \downarrow \quad q' \\
\end{array}
\]

(2.) **MVL-step**: \((q', MVL) \in \delta(q, u)\) :

\[
\begin{array}{cccccc}
  & a & b & c & d & e & f \\
  \downarrow \quad q \\
\end{array} \quad \Rightarrow \quad \begin{array}{cccccc}
  & a & b & c & d & e & f \\
  \downarrow \quad q' \\
\end{array}
\]
(3.) **Rewrite-step:** \((q', x, y) \in \delta(q, u) \quad (u = xz) :\)

\[
\begin{array}{cccccccc}
\ldots & a & b & c & d & e & f & g & h & \ldots \\
& x & z & & & & & & & \\
\end{array}
\quad \Rightarrow \quad
\begin{array}{cccccccc}
\ldots & a & b' & c' & d & e & f & g & h & \ldots \\
& y & z & & & & & & & \\
\end{array}
\]

(4.) **Restart-step:** \((q', \text{Restart}) \in \delta(q, u) :\)

\[
\begin{array}{cccccccc}
\emptyset & a & b & c & \ldots & d & e & f & g & h & \ldots \\
& & & & q & & & & & \\
\end{array}
\quad \Rightarrow \quad
\begin{array}{cccccccc}
\emptyset & a & b & c & \ldots & d & e & f & g & h & \ldots \\
& & & & q' & & & & & \\
\end{array}
\]

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

(6.) \(\delta(q, u) = \emptyset :\) halt and reject
2. Definitions and Examples

**Configurations**

```
| $ | a_1 | \ldots | a_m | b_1 | b_2 | \ldots | b_k | b_{k+1} | \ldots | $ |
```

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

| $q$ |

Restart configuration: $q \emptyset w \$ \ (w \in \Gamma^*)$
resulting from a restart step

Initial configuration: $q_0 \emptyset w \$ \ (w \in \Sigma^*)$

Accepting configuration: Accept

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

**Cycle:**

\[ q_1 \downarrow w \quad \vdash^* \quad \downarrow xq_2 u y \quad \vdash \quad q_3 \downarrow x u y . \]

Each cycle contains at least one application of a Rewrite step.

**Tail:**

\[ q_1 \downarrow w \quad \vdash^* \quad \downarrow xq_2 u y \quad \vdash \quad \text{Accept}. \]

A tail need not contain any application of a Rewrite step.
Language Accepted by $M$

$w \in \Gamma^*$ is accepted by $M$, if there exists a computation $q_0 \upharpoonright w \downharpoonright \vdash_M \text{Accept}$.

$L_C(M) := \{ w \in \Gamma^* \mid w \text{ is accepted by } M \}$ is the characteristic language of $M$.

$L(M) = L_C(M) \cap \Sigma^* = \{ w \in \Sigma^* \mid w \text{ is accepted by } M \}$ is the input language of $M$. 
Restricted Types of Restarting Automata

(a) Restrictions on the **Rewrite**-operation:

- **length-reducing** restarting automata (standard model):
  For all \((q', x, y) \in \delta(q, u)\) : \(|x| > |y|\).

- **shrinking** restarting automata:
  \[\exists \text{ weight-function } g : Q \cup \Gamma \cup \{\$, \$\} \rightarrow \mathbb{N}_+ :\]
  For all \((q', x, y) \in \delta(q, u)\) : \(g(x) > g(y)\).
  Notation: \(s\)-

- **no look-ahead** restarting automata (standard model):
  For all \((q', x, y) \in \delta(q, u)\) : \(x = u\).

- **restarting automata without aux. symbols**: \(\Gamma = \Sigma\)
  Notation: \(-W\)

- **deleting** restarting automata:
  For all \((q', x, y) \in \delta(q, u)\) : \(y\) is a scattered subword of \(x\).
  Notation: \(-\varepsilon\)
(b) **Restrictions on the** Move-operations:

- **unrestricted move-operations MVL and MVR:**
  Notation: RL-

- **one-way restarting automata:**
  no MVL-operations
  Notation: RR-

- **restricted one-way restarting automata:**
  no MVL-operations,
  rewrite- and restart-operations combined into a single operation.
  Notation: R-
(c) Restrictions on the Restart-operation:

- no restriction: non-forgetting restarting automata.
  Notation: nf-

- forgetting restarting automata (standard model):
  For all \((q', \text{Restart}) \in \delta(q, u) : q' = q_0\).
(d) Restrictions on admissible computations:

- **no restriction:**
  - freely rewriting restarting automata.
  - Notation:  \( F- \)

- **\( j \)-rewriting** restarting automata \( (j \geq 1) \):
  - Each cycle (and each tail) contains at most \( j \) applications of rewrite operations.
  - Notation:  \( j\text{-rewr}- \)

- **1-rewriting** restarting automata: standard model.

- **monotone** restarting automata:
  - In each computation the right distance of a rewrite configuration is at most as large as the right distance of the previous rewrite configuration.
  - Notation:  \( \text{mon-} \)
Example 1

Let $M := (Q, \Sigma, \Sigma, \emptyset, \$, $q_0, 3, \delta)$, where $Q := \{q_0, q_c, q_d, q_r\}$, 
$\Sigma := \{a, b, c, d\}$, and $\delta$ is given through the following table:

1. $(q_0, x) = (q_0, \text{MVR}), x \in \{aaa, aab, abb, abc, bbb, bbc, bbd\},$
2. $(q_0, \$c \$) = \text{Accept},$
3. $(q_0, \$d \$) = \text{Accept},$
4. $(q_0, \$ab\$) = (q_0, \text{MVR}),$
5. $(q_0, \$aa\$) = (q_0, \text{MVR}),$
6. $(q_0, \$c \$) = (q_c, \text{MVL}),$
7. $(q_0, \$d \$) = (q_d, \text{MVL}),$
8. $(q_r, -) = \text{Restart},$
9. $(q_c, abc) = (q_r, abc, c),$
10. $(q_c, bbc) = (q_c, \text{MVL}),$
11. $(q_c, bbb) = (q_c, \text{MVL}),$
12. $(q_c, abb) = (q_r, abb, b),$
13. $(q_d, bbd) = (q_d, \text{MVL}),$
14. $(q_d, bbb) = (q_d, \text{MVL}),$
15. $(q_d, abb) = (q_r, abb, \varepsilon).$

$L(M) = \{a^n b^n c \mid n \geq 0\} \cup \{a^n b^{2n} d \mid n \geq 0\} =: L_1.$
Proposition 2 (Error Preserving Property)

Let $M = (Q, \Sigma, \Gamma, \phi, $, $q_0, k, \delta)$ be a restarting automaton, and let $u, v$ be words over its tape alphabet $\Gamma$. If $q_0 \phi u \vdash_M^c q_0 \phi v$ holds and $u \notin L_C(M)$, then $v \notin L_C(M)$, either.

Proposition 3 (Correctness Preserving Property)

Let $M = (Q, \Sigma, \Gamma, \phi, $, $q_0, k, \delta)$ be a restarting automaton, and let $u, v$ be words over its tape alphabet $\Gamma$. If $q_0 \phi u \vdash_M^c q_0 \phi v$ is an initial segment of an accepting computation of $M$, then $v \in L_C(M)$. 
Proposition 4 (Pumping Lemma)

For any restarting automaton $M = (Q, \Sigma, \Gamma, \psi, \$, $q_0, k, \delta)$, there exists a constant $d$ such that the following holds. Assume that $q_1 \psi uvw \vdash^c_M q_2 \psi uv'w$, where $u = u_1u_2u_3$ and $|u_2| = d$. Then there exists a factorization $u_2 = z_1z_2z_3$ such that $z_2$ is non-empty, and

$$q_1 \psi u_1z_1(z_2)^iz_3u_3vw \vdash^c_M q_2 \psi u_1z_1(z_2)^iz_3u_3v'w$$

holds for all $i \geq 0$, that is, $z_2$ is a ‘pumping factor’ in the above cycle. Similarly, such a pumping factor can be found in any factor of length $d$ of $w$. Such a pumping factor can also be found in any factor of length $d$ of a word accepted in a tail computation.
RRWW-automaton: no MVL-steps

Theorem 5 (Plátek 2001)

Let $M_L = (Q_L, \Sigma, \Gamma, \psi, \$, q_0, k, \delta_L)$ be a (nf-s-) RLWW-automaton. Then there exists a (nf-s-) RRWW-automaton $M_R = (Q_R, \Sigma, \Gamma, \psi, \$, q_0, k, \delta_R)$ such that, for all $u, v \in \Gamma^*$,

$q_1 \phi u \$ \models_{M_L}^c q_2 \phi v \$ if and only if $q_1 \phi u \$ \models_{M_R}^c q_2 \phi v \$,

and the languages $L(M_L)$ and $L(M_R)$ coincide.
Lemma 6

Each (nf-s-) RRWW-automaton $M$ is equivalent to a (nf-s-) RRWW-automaton $M'$ that satisfies the following additional restriction:

(*) $M'$ makes an accept or a restart step only when it sees the right border marker $\$$. in its read/write window.
Meta-instructions for nf-s-RRWW-automata:

\((q_1, E_1, u \rightarrow v, E_2, q_2) : E_1, E_2 \text{ regular constraints, } q_1, q_2 \in Q, u \rightarrow v \text{ Rewrite step, }\)

\(q_1 \not\vdash w \vdash_c q_2 \not\vdash w_1 v w_2 \text{ if } w = w_1 u w_2 \text{ such that } \not\vdash w_1 \in L(E_1) \text{ and } w_2 \not\vdash \in L(E_2).\)

\((q_1, E, \text{ Accept}) : E \text{ regular constraint, }\)

\(q_1 \not\vdash w \vdash^* \text{ Accept if } \not\vdash w \not\vdash \in L(E).\)

Example 7

An RRWW-automaton for \(L_1 := \{a^n b^n c, a^n b^{2n} d \mid n \geq 0\}\):

1. \((\not\vdash \cdot a^*, \ ab \rightarrow \varepsilon, \ b^* \cdot c \not\vdash),\)
2. \((\not\vdash \cdot a^*, \ ab \rightarrow \varepsilon, \ b^* \cdot d \not\vdash),\)
3. \((\not\vdash c \not\vdash, \ \text{Accept}),\)
4. \((\not\vdash d \not\vdash, \ \text{Accept}).\)
RWW-automaton: combined Rewrite/Restart steps

Meta-instructions for nf-s-RWW-automata:

\[(q_1, E, u \rightarrow v, q_2) : q_1 \vdash w \not\in^c q_2 \vdash w_1 v w_2 \]
if \( w = w_1 u w_2 \) such that \( \not\in w_1 \in L(E) \).

\[(q_1, E, \text{Accept}) : q_1 \not\in w \not\in^* \text{Accept} \text{ if } \not\in w \in L(E). \]

Example 8

An RWW-automaton for \( L_1 \):

(1) \((\not\in \cdot a^*, \quad ab \rightarrow C),\) \(\text{ (5) } (\not\in C c \not\in, \text{ Accept}),\)
(2) \((\not\in \cdot a^*, \quad abb \rightarrow D),\) \(\text{ (6) } (\not\in D \cdot d \not\in, \text{ Accept}),\)
(3) \((\not\in \cdot a^*, \quad a C b \rightarrow C),\) \(\text{ (7) } (\not\in c \not\in, \text{ Accept}),\)
(4) \((\not\in \cdot a^*, \quad a D b b \rightarrow D),\) \(\text{ (8) } (\not\in \cdot d \not\in, \text{ Accept}).\)
2. Definitions and Examples

NP \cap CSL

\mathcal{L}(RLWW)

P \cap DCSL

\mathcal{L}(det-RLWW)

\mathcal{L}(RRWW)

\mathcal{L}(RWW)

\mathcal{L}(RW)

\mathcal{L}(RR)

\mathcal{L}(R)

\mathcal{L}(RLW)

\mathcal{L}(RL)

\mathcal{L}(RL)

\mathcal{L}(RW)

\mathcal{L}(RRW)

\mathcal{L}(det-RLWW)

\mathcal{L}(det-RWW)

\mathcal{L}(det-RW)

\mathcal{L}(det-RRWW)

\mathcal{L}(det-RRW)

\mathcal{L}(det-RR)

\mathcal{L}(det-R)
$M = (Q, \Sigma, \Gamma, \phi, $, $, q_0, k, \delta)$ nf-s-RLWW-automaton, $L(M) \subseteq \Sigma^*$

$M_1 := (Q, \Gamma, \Gamma, \phi, $, $, q_0, k, \delta)$ nf-s-RLW-automaton such that

$L(M_1) \subseteq \Gamma^*$ satisfies $L(M_1) \cap \Sigma^* = L(M)$.

Hence, $w \mapsto w (w \in \Sigma^*)$ is a reduction from $L(M)$ to $L(M_1)$.

**Corollary 9**

The language class $\mathcal{L}((\text{nf-s-})\text{RLWW})$ is reducible in linear time and constant space to the language class $\mathcal{L}((\text{nf-s-})\text{RLW})$.

Analogously for nf-s-RRWW- and nf-s-RWW-automata and their deterministic counter parts.
Let $\Gamma_1 = \{a_1, \ldots, a_m\}$, $k \geq 1$, $\Gamma_2 = \{0, 1, c, d\}$.
Define $\varphi_{k,m} : \Gamma_1 \to \Gamma_2^+$ by

$$a_i \mapsto c1^{m+1-i}0^i(cd1^{m+1}0^{m+1})^k \quad (1 \leq i \leq m).$$

Then, for all $1 \leq i \leq m$,

$$|\varphi_{k,m}(a_i)| = (m + 2) \cdot (2k + 1).$$

The encoding $\varphi_{k,m}$ is naturally extended to strings:

$\varphi_{k,m}(x_1 x_2 \cdots x_n) := \varphi_{k,m}(x_1) \cdots \varphi_{k,m}(x_n)$ for all $x_1, \ldots, x_n \in \Gamma_1$, $n \geq 0$.

$\varphi_{k,m} : \Gamma_1^* \to \Gamma_2^*$ is an encoding.

**Lemma 10**

*For all $u \in \Gamma_1^k$ and $v \in \Gamma_1^*$, if $|v| < k$, then $\varphi_{k,m}(v)$ is a scattered subword of $\varphi_{k,m}(u)$.***
Theorem 11

If a language $L$ is accepted by a (nf-) RLW-automaton $M$ with tape alphabet $\Gamma_1$ of size $m$ and read/write window of size $k$, then there exists a (nf-) RL-automaton $M'$ which accepts the language $\varphi_{k,m}(L) \subseteq \Gamma_2^*$. Analogously for nf-RRW- and nf-RW-automata and their deterministic variants.

Corollary 12

The language class $\mathcal{L}((\text{nf-})\text{RLW})$ is reducible in linear time and constant space to the language class $\mathcal{L}((\text{nf-})\text{RL})$.

Analogously for nf-RRW- and nf-RW-automata and their deterministic variants.
2. Definitions and Examples

Example 13

Let \( M' := (Q, \Sigma, \Gamma, \phi, \$, q_0, 4, \delta) \) be the RWW-automaton defined by

\[ \Sigma := \{a, b, \#\}, \; \Gamma := \Sigma \cup \Gamma' \cup \{\#'\}, \]
\[ \Gamma' := \{[c, d, S] \mid c, d \in \{a, b\}, S \in \{0, 1\}\} \]

and the following list of meta-instructions, where
\( c, d, e, f \in \{a, b\}, \; S, S' \in \{0, 1\} \) and \( \overline{S} := 1 - S \):

\[ L_{\text{copy}} := \{w\#w \mid w \in \{a, b\}^*\} \in \text{CSL} \setminus \text{GCSL}. \]
(1) \((\emptyset, cd \rightarrow [c, d, 0])\),
(2) \((\emptyset \cdot \Gamma'^* \cdot [e, f, S], cd \rightarrow [c, d, \overline{S}])\),
(3) \((\emptyset[c, d, 0] \cdot \{a, b\}^* \cdot \#, cd \rightarrow [c, d, 0])\),
(4) \((\emptyset \cdot \Gamma'^* \cdot [c, d, S] \cdot \{a, b\}^* \cdot \# \cdot \Gamma'^* \cdot [e, f, \overline{S}], cd \rightarrow [c, d, S])\),
(5) \((\emptyset \cdot \Gamma'^*, [c, d, S] \cdot X \cdot [e, f, S'] \rightarrow \#')\) for all \(X \in \{\#, \#'\}\),
(6) \((\emptyset \cdot \Gamma'^*, [c, d, S] \cdot g \cdot X \cdot [e, f, S'] \rightarrow g\#')\)
for all \(g \in \{a, b\}, X \in \{\#, \#'\}\),
(7) \((\emptyset \#'\$, Accept)\),
(8) \((\emptyset g\#'g\$, Accept)\) for all \(g \in \{a, b\}\),
(9) \((\emptyset \#$, Accept)\),
(10) \((\emptyset g\#$, Accept)\) for all \(g \in \{a, b\}\).
Given the string $abaab \# abaab$ as input, $M'$ can execute the following computation:

$$q_0 \notin abaab \# abaab$$

\[\begin{align*}
&\vdash_c q_0 \notin [a, b, 0]aab \# abaab$ \\
&\vdash_c (1) q_0 \notin [a, b, 0]aab \# [a, b, 0]aab$ \\
&\vdash_c (3) q_0 \notin [a, b, 0][a, a, 1]b \# [a, b, 0]aab$ \\
&\vdash_c (2) q_0 \notin [a, b, 0][a, a, 1]b \# [a, b, 0][a, a, 1]b$ \\
&\vdash_c (4) q_0 \notin [a, b, 0]b \# [a, a, 1]b$ \\
&\vdash_c (6) q_0 \notin b \# ’b$ \\
&\vdash_c (6) \text{Accept.} \\
\end{align*}\]
Input: \( w_1 \neq w_2 \)

Phase 1: Compress syllables \( w_1 \) and \( w_2 \). If success, then \( w_2 \) is a scattered substring of \( w_1 \). (\( * \))

Phase 2: Compare length. If success, then \( |w_1| = |w_2| \). (\( ** \))

(\( * \)) + (\( ** \)) imply that \( w_1 = w_2 \).

It follows that \( L(M') = L_{\text{copy}} \).
Corollary 14

\( \mathcal{L}(R) \not\subseteq \text{GCSL} \).
### Look-Ahead Hierarchies

For $k \geq 1$, $X \in \{R, RR, RW, RRW, RWW, RRWW\}$, $X(k) = X$-automata with window size $k$.

#### Theorem 15 (Mraz 2001, Reimann 2007)

(a) $\text{REG} = \mathcal{L}(R(1)) = \mathcal{L}(RW(1)) = \mathcal{L}(RWW(1))$

$b = \mathcal{L}(\text{det-RR}(1)) = \mathcal{L}(\text{det-RRW}(1)) = \mathcal{L}(\text{det-RRWW}(1))$

$c \subset \mathcal{L}(RR(1)) = \mathcal{L}(RRW(1)) = \mathcal{L}(RRWW(1)).$

(b) $\mathcal{L}(p - X(k)) \subset \mathcal{L}(p - X(k + 1))$ for all $k \geq 1$,

$p \in \{\varepsilon, \text{mon, det, det-mon}\}$ and $X \in \{R, RR, RW, RRW\}$. 
A cycle $C$ of a computation of a nf-RLWW-automaton:

$$q_1 \parr w \xrightarrow{\ast_{MVR,MVL}} \parr xquy \xrightarrow{\text{Rewrite}} \parr xvq'y \xrightarrow{\ast} q_2 \parr xvy$$

$$D_r(C) := |y| + 1 \quad \text{right distance of } C$$

A sequence of cycles $C_1, C_2, \ldots, C_{n-1}, C_n$ is monotone if $D_r(C_1) \geq \ldots \geq D_r(C_{n-1}) \geq D_r(C_n)$.

A computation of a nf-RLWW-automaton $M$ is monotone if the corresponding sequence of cycles is monotone.

$M$ is monotone, if, for each initial configuration $q_0 \parr w$, each computation starting with $q_0 \parr w$ is monotone.
Theorem 16 (Jančar et al. 1999, Messerschmidt 2008)

\[
\text{CFL} = \mathcal{L}(\text{mon-RWW}) = \mathcal{L}(\text{mon-RLWW}) = \mathcal{L}(\text{mon-nf-s-RLWW}).
\]

Proof outline

(i) Context-free grammar \( G \) in Chomsky normal form
   \[ \Rightarrow \] monotone RWW-automaton \( M \):
   \( M \) simulates rightmost derivation of \( G \) in reverse order.

(ii) Let \( M \) be a monotone nf-s-RLWW-automaton.
   \[ \Rightarrow \exists \] monotone nf-s-RRWW-automaton \( M' \): \( \mathcal{L}(M) = \mathcal{L}(M') \).

Cycle \( C \) of \( M' \):

\[
q_1 \nf \pm \mathcal{F} \Rightarrow^* \nf \pm q_2 \pm q_1
\]

PDA:

<table>
<thead>
<tr>
<th>pushdown</th>
<th>finite control</th>
<th>input tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{F} )</td>
<td>( x )</td>
<td>( u )</td>
</tr>
</tbody>
</table>
Theorem 17 (Jančar et al. 1999, Messerschmidt 2008)

\[
\text{DCFL} = \mathcal{L}(\text{det-mon-R}) = \mathcal{L}(\text{det-mon-RRWW}) \\
= \mathcal{L}(\text{det-mon-nf-s-R}) = \mathcal{L}(\text{det-mon-nf-s-RWW}).
\]

Corollary 18

(a) \(\text{DCFL} \subsetneq \mathcal{L}(\text{det-mon-RL}).\)

(b) \(\text{DCFL} \subsetneq \mathcal{L}(\text{det-mon-nf-RR}).\)
What is the expressive power of the various types of deterministic restarting automata?

Example 19

\[ M = (Q, \{a\}, \{a, A\}, \emptyset, \$, q_0, 3, \delta): \]

1. \((\emptyset \cdot a^*, aa\$ \rightarrow A\$, \varepsilon)\)
2. \((\emptyset \cdot a^*, aaA \rightarrow AA, A^* \cdot \$)\)
3. \((\emptyset \cdot A^*, AA\$ \rightarrow a\$, \varepsilon)\)
4. \((\emptyset \cdot A^*, AAh \rightarrow a, a^* \cdot \$)\)
5. \((\emptyset \cdot a \cdot \$, Accept)\)
6. \((\emptyset \cdot A \cdot \$, Accept)\).

\(M\) is a deterministic RRWW-automaton satisfying

\[ L(M) = L_5 := \{ a^{2^n} \mid n \geq 0 \} \not\in \text{CFL}. \]
Two-Pushdown Automata

\[
\begin{array}{c|c|c|c|c|c|c}
\bot & \cdots & a_1 & \cdots & a_k & b_1 & \cdots & b_k & \cdots & \bot \\
\end{array}
\]

\[M = (Q, \Sigma, \Gamma, \delta, q_0, \bot, st_1, st_2, F):\]
\[\delta: Q \times \Gamma^{\leq k} \times \Gamma^{\leq k} \rightarrow \mathcal{P}_{\text{fin}}(Q \times \Gamma^* \times \Gamma^*)\]

Initial configuration for input \( w \in \Sigma^*:\)

\[
\begin{array}{c|c|c|c|c|c|c}
\bot & st_1 & w & st_2 & \bot \\
\end{array}
\]

\(M\) is shrinking: \( \exists\) weight function \( \varphi: Q \cup \Gamma \rightarrow \mathbb{N}_+:\)
if \( \delta(q, u, v) \ni (p, u', v'), \) then \( \varphi(u'pv') < \varphi(uqv).\)

\(M\) is length-reducing: if \( \delta(q, u, v) \ni (p, u', v'), \) then \( |u'v'| < |uv|.\)
Theorem 20 (Niemann 2003)

For a language \( L \subseteq \Sigma^* \), the following statements are equivalent:

(a) \( L \) is accepted by a shrinking dTPdA.

(b) \( L \) is accepted by a length-reducing dTPdA.

(c) \( L \) is a Church-Rosser language.

\( L \subseteq \Sigma^* \) is a Church-Rosser language (CRL) if there exist an alphabet \( \Gamma \supseteq \Sigma \), a finite, length-reducing, confluent srs \( S \) on \( \Gamma \), two strings \( t_1, t_2 \in (\Gamma \setminus \Sigma)^* \cap \text{IRR}(S) \), and a letter \( Y \in (\Gamma \setminus \Sigma) \cap \text{IRR}(S) \) such that, for all \( w \in \Sigma^* \),

\[
    w \in L \quad \text{iff} \quad t_1 w t_2 \to^*_S Y.
\]

The shrinking/length-reducing (nondeterministic) TPdA characterize the growing context-sensitive languages.
Theorem 21 (Niemann, Otto 1999, 2000)

\[ \text{CRL} \equiv \mathcal{L}(\text{det-RWW}) = \mathcal{L}(\text{det-s-RRWW}). \]

Corollary 22

The language classes \( \mathcal{L}(\text{det-R}) \) and CFL are incomparable with respect to inclusion.

Theorem 23

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

(b) \( \text{CRL} \subsetneq \mathcal{L}(\text{det-nf-RWW}). \)
Let $M$ be an RLWW-automaton, and let $c \in \mathbb{N}$.

A sequence of cycles $C_1, C_2, \ldots, C_n$ of $M$ is \textit{weakly $c$-monotone} if $D_r(C_{i+1}) \leq D_r(C_i) + c$ ($i = 1, \ldots, n - 1$).

A computation of $M$ is weakly $c$-monotone if the corresponding sequence of cycles is weakly $c$-monotone.

$M$ is \textit{weakly $c$-monotone} if all its computations are weakly $c$-monotone.

$M$ is \textit{weakly monotone} $\iff \exists c \in \mathbb{N} : M$ is weakly $c$-monotone.

Let $M$ be a deterministic RRWW-automaton:

$C_i : \not\exists x \not\exists y : D_r(C_i) = |y| + 1$

$C_{i+1} : D_r(C_{i+1}) \leq |y| - 1 + |y| + 1 = D_r(C_i) + |y| - 1$

$\Rightarrow M$ is weakly monotone.

\textbf{Theorem 24}

$\text{GCSL} = \mathcal{L}(\text{w-mon-RWW}) = \mathcal{L}(\text{w-mon-s-RRWW})$. 
Corollary 25

\[ \text{GC SL} \subseteq \mathcal{L}(RWW). \]

Theorem 26

\[ \mathcal{L}(R(R)WW) \text{ contains NP-complete languages.} \]

Corollary 27

\[ \text{LOG}(\mathcal{L}(R)) = \text{NP}. \]
Theorem 28

\[ \mathcal{L}(sRWW) = \mathcal{L}(sRRWW) = \mathcal{L}(sRLWW) = \mathcal{L}(\text{nf-s-RLWW}). \]

Theorem 29

If \( L \in \mathcal{L}(RRWW) \), then \( r_{54}(L) \in \mathcal{L}(RWW) \), where \( r_{54} : \Sigma \rightarrow \Sigma \cup \{\Delta\} \) is defined through \( a \mapsto a\Delta \) (\( a \in \Sigma \)).
A finite-change automaton is a nondeterministic Turing machine $A$ with a single tape that is parameterized by a function $f : \mathbb{N} \rightarrow \mathbb{N}$ satisfying $f(n) \geq n$ for all $n \in \mathbb{N}$ and a constant $k \in \mathbb{N}_+$. Given an input of length $n$, $A$ must not visit more than $f(n)$ cells, and it must not change the content of any cell more than $k$ times during any accepting computation on the given input.

$$FC(f) := \bigcup_{k>0} kC(f)$$

If $f(n) = n$ for all $n$, then we use the notation $FC$ and $kC$.

**Lemma 30 (v.Braunmühl, Verbeek 1979)**

For all $c \geq 1$, $\mathcal{L}(FC(f(n))) = \mathcal{L}(FC(c \cdot f(n)))$.

**Theorem 31**

$\mathcal{L}(sRRWW) = \mathcal{L}(FC)$. 
4. Nondeterministic Restarting Automata

\[ \mathcal{L}(FC) \uparrow \mathcal{L}(sRRWW) \]
\[ \mathcal{L}(RRWW) \uparrow \mathcal{L}(RWW) \]
\[ \mathcal{L}(w\text{-mon-RRWW}) \]

**Fig.** The taxonomy of language classes between GCSL and CSL
Theorem 32

(a) \( \mathcal{L}(\text{det-left-mon-(s-)R}(X)) = (\mathcal{L}(\text{det-mon-(s-)R}(X)))^R \).

(b) \( \mathcal{L}(\text{det-mon-RL}) = \mathcal{L}(\text{det-mon-s-RLWW}) \).

(c) \( \mathcal{L}(\text{det-left-mon-RL}) = \mathcal{L}(\text{det-left-mon-s-RLWW}) = \mathcal{L}(\text{det-left-mon-RWW}) = \mathcal{L}(\text{det-left-mon-s-RRWW}) \).

Theorem 33 (Messerschmidt 2008)

(a) \( \mathcal{L}(\text{det-mon-nf-s-RLWW}) = \mathcal{L}(\text{det-mon-RL}) \).

(b) \( \mathcal{L}(\text{det-mon-nf-RRWW}) = \mathcal{L}(\text{det-mon-RL}) \).
Let $G = (N, T, P, S)$ be a context-free grammar. By $\Rightarrow^*_R$ we denote the rightmost derivation relation induced by $G$. If $S \Rightarrow^*_R \alpha$, then $\alpha$ is a right sentential form of $G$.

**Definition 34 (Culik, Cohen 1973)**

Let $\pi = \{E_1, \ldots, E_n\}$ be a partition of $T^*$ into $n$ disjoint regular sets. Then $x \equiv y \mod (\pi)$ iff $x$ and $y$ are in the same set $E_i$.

$G$ is called **LR($\pi$)**, if, for all rightmost derivations

$S \Rightarrow^*_R \alpha_1 A_1 y_1 \Rightarrow_R \alpha_1 \gamma y_1$ and $S \Rightarrow^*_R \alpha_2 A_2 y_3 \Rightarrow_R \alpha_1 \gamma y_2$,

$(A_1, A_2 \in N, \alpha_1, \alpha_2, \gamma \in (N \cup T)^*, y_1, y_2, y_3 \in T^*)$

$y_1 \equiv y_2 \mod (\pi)$ implies that $A_1 = A_2, \alpha_1 = \alpha_2$, and $y_2 = y_3$.

$G$ is called **left-to-right regular** (LR-regular) if it is LR($\pi$) for some regular partition $\pi$ of $T^*$.

$L$ is **LR-regular** if it is generated by some LR-regular grammar.

LRR denotes the class of LR-regular languages.
Theorem 35 (Culik,Cohen 1973)

1. DCFL $\subsetneq$ LRR $\subsetneq$ UCFL.
2. LRR is incomparable to $\text{DCFL}^R$ under inclusion.
3. LRR is an abstract family of languages.
4. It is decidable whether a given LR-regular language is regular.

Theorem 36

LRR $= \mathcal{L}(\text{det-mon-RL}) = \mathcal{L}(\text{det-mon-nf-s-RLWW})$
$= \mathcal{L}(\text{det-mon-nf-RRWW})$
$\subsetneq$ CRL $\cap$ CFL.
Definition 37

Let $M = (Q, \Sigma, \Gamma, \phi, \$, $q_0, k, \delta)$ be a det-FRR-automaton.

(a) A word $w \in \Gamma^*$ is not immediately rejected by $M$ if, starting from the restarting configuration $q_0 \phi w \$, $M$ either performs a cycle of the form $w \vdash^C_M z$ for some word $z \in \Gamma^*$, or $M$ accepts $w$ in a tail computation. By NIR($M$) we denote the set of all words that are not immediately rejected by $M$.

(b) The det-FRR-automaton $M$ is called lexicalized if there exists a constant $j \in \mathbb{N}_+$ such that, whenever $v \in (\Gamma \setminus \Sigma)^*$ is a factor of a word $w \in \text{NIR}(M)$, then $|v| \leq j$.

(c) $M$ is called strongly lexicalized if it is lexicalized, and if each of its rewrite operations just deletes some symbols.
6. Freely Rewriting Restarting Automata

LRR: class of lexicalized FRR-automata

SLRR: class of strongly lexicalized FRR-automata

\( t \text{-}(S)LRR \): class of (S)LRR-automata executing at most
\( t \) rewrite steps in any cycle
\[ L_\infty := \{ (a^n b^n)^m \mid n, m \geq 0 \} \in \mathcal{L}_C(\text{SLRR}). \]

On the other hand,

\[ \{ (a^n b^n)^m \mid n \geq 0, m \leq j \} \in \mathcal{L}_C(j\text{-SLRR}) \setminus \mathcal{L}_C((j - 1)\text{-LRR}). \]

**Corollary 38**

For all \( j \in \mathbb{N}_+ \),

(a) \( \mathcal{L}_C(j\text{-}(S)LRR) \subset \mathcal{L}_C((j + 1)\text{-}(S)LRR). \)

(b) \( \bigcup_{j \geq 1} \mathcal{L}_C(j\text{-}(S)LRR) \subset \mathcal{L}_C((S)LRR). \)
7. Open Problems

1. $\mathcal{L}(\text{RWW}) \subseteq \mathcal{L}(\text{RRWW})$?
2. Are shrinking restarting automata in general more expressive than their length-reducing counterparts?
3. Give an example language that is context-sensitive, but that is not accepted by any RRWW-automaton!