Jumping Restarting Automata

Qichao Wang, Yongming Li

College of Computer Science
Shaanxi Normal University
Xi’an 710119, China

10th Workshop on Non-Classical Models of Automata and Applications

August 21-22, 2018, Košice, Slovakia
Outline

1. Restarting Automata
2. Jumping Restarting Automata
3. Fast Jumping Restarting Automata
4. Monotone Fast Jumping Restarting Automata
5. Conclusions
**Restarting Automata (RRWW-Automata)**

\[ L(M_1) = \{ a^n b^n c \mid n \geq 0 \} \]
Restarting Automata (RRWW-Automata)

$M_1 :$

\[
\begin{array}{cccccccc}
\emptyset & a & a & a & a & a & b & b & b & b & b & c & \$
\end{array}
\]

$Q. \text{ Wang, Y. Li (SNNU)}$

Jumping Restarting Automata

August 21-22, 2018, Košice

$L(M_1) = \{ a^n b^n c \mid n \geq 0 \}.$
Restarting Automata (RRWW-Automata)

\[ M_1 : \]

\[
\begin{array}{cccccc}
\phi & a & a & a & a & b & b & b & b & c & \$ \\
\end{array}
\]

Flexible tape

\[ L(M_1) = \{ a^n b^n c \mid n \geq 0 \}. \]
Restarting Automata (RRWW-Automata)

\[ M_1 : \]

\[
\begin{array}{cccccccc}
\phi & a & a & a & a & a & b & b & b & b & b & c & \$ \\
\end{array}
\]

read/write-window

\[ Q_0 \]

\[ L(M_1) = \{ a^n b^n c \mid n \geq 0 \}. \]
Restarting Automata (RRWW-Automata)

\[ M_1 : \]
\[ \phi a a a a b b b b c $ \]

\[ L(M_1) = \{ a^n b^n c \mid n \geq 0 \} . \]
Restarting Automata (RRWW-Automata)

\[ M_1 : \]

\[
\begin{array}{cccccccc}
\phi & a & a & a & a & b & b & b & c & \$ \\
\end{array}
\]

\[
q_0
\]

\[
ab \rightarrow \lambda
\]

\[ L(M_1) = \{ a^n b^n c \mid n \geq 0 \} . \]
Restarting Automata (RRWW-Automata)

$M_1 :$

$q_1$

$\text{restart}$

$L(M_1) = \{ a^n b^n c \mid n \geq 0 \}.$
Restarting Automata (RRWW-Automata)

$M_1$:

\[ L(M_1) = \{ a^n b^n c \mid n \geq 0 \}. \]
Restarting Automata (RRWW-Automata)

\[ M_1 : \]

\[
\begin{array}{c}
\phi \\
\ast \\
\$ \\
\end{array}
\]

read/write-window

flexible tape

Accept

\[ L(M_1) = \{ a^n b^n c \mid n \geq 0 \}. \]
Overview

\[ \mathcal{L}(RRWW) \]

\[ \mathcal{L}(RRW) \]

\[ \mathcal{L}(RW) \]

\[ \mathcal{L}(R) \]

\[ \mathcal{L}(det-RRWW) \]

\[ \mathcal{L}(det-RRW) \]

\[ \mathcal{L}(det-RW) \]

\[ \mathcal{L}(det-RR) \]

\[ \mathcal{L}(det-R) \]

[Otto, 2006]
$L(M_1) = \{ a^n b^n c \mid n \geq 0 \}.$
Jumping Restarting Automata (JRRWW-Automata)

\[ M_2 : \]

\[ L(M_1) = \{ a^n b^n c \mid n \geq 0 \}. \]
Jumping Restarting Automata (JRRWW-Automata)

\[ M_2 : \]

\[ L(M_1) = \{ a^n b^n c \mid n \geq 0 \} \]
Jumping Restarting Automata (JRRWW-Automata)

$M_2 :$

<table>
<thead>
<tr>
<th>q</th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>b</th>
<th>b</th>
<th>b</th>
<th>c</th>
<th>$</th>
</tr>
</thead>
</table>

Flexible tape

$q_0$

$L(M_1) = \{ a^n b^n c \mid n \geq 0 \}.$
Jumping Restarting Automata (JRRWW-Automata)

$M_2 :$

```
read/write-window

 staple tape

Accept
```

$L(M_1) = \{ a^n b^n c \mid n \geq 0 \}.$
Lemma 1

For each type $X \in \{R, RR, RW, RRW, RWW, RRWW\}$, $\mathcal{L}(JX) \subseteq \mathcal{L}(X)$. 
Results

Lemma 1

For each type $X \in \{R, RR, RW, RRW, RWW, RRWW\}$, $L(JX) \subseteq L(X)$.

Lemma 2

For each type $X \in \{R, RR, RW, RRW, RWW, RRWW\}$, $L(X) \subseteq L(JX)$. 
Results

Lemma 1

For each type $X \in \{R, RR, RW, RRW, RWW, RRWW\}$, $L(JX) \subseteq L(X)$.

Lemma 2

For each type $X \in \{R, RR, RW, RRW, RWW, RRWW\}$, $L(X) \subseteq L(JX)$.

Proof Idea

Use a read/write window of size $k + 1$ in order to ensure that in each jump-right step, the automaton moves exactly one position.
Results

Lemma 1

For each type $X \in \{R, RR, RW, RRW, RWW, RRWW\}$, $\mathcal{L}(JX) \subseteq \mathcal{L}(X)$.

Lemma 2

For each type $X \in \{R, RR, RW, RRW, RWW, RRWW\}$, $\mathcal{L}(X) \subseteq \mathcal{L}(JX)$.

Proof Idea

Use a read/write window of size $k + 1$ in order to ensure that in each jump-right step, the automaton moves exactly one position.

Theorem 3

For each type $X \in \{R, RR, RW, RRW, RWW, RRWW\}$, $\mathcal{L}(JX) = \mathcal{L}(X)$. 
Definition 4

A jumping restarting automaton $M$ is called a fast jumping restarting automaton, if in each cycle (or tail computation), $M$ performs at most one jump-right step before a rewrite step (or accept). For the types with RR, there exists a constant $c \in \mathbb{N}$, such that after rewriting the automaton $M$ is allowed to execute at most $c$ jump-right steps. Here we use the prefix FJ to denote the types of fast jumping restarting automata.
Definition 4

A jumping restarting automaton $M$ is called a fast jumping restarting automaton, if in each cycle (or tail computation), $M$ performs at most one jump-right step before a rewrite step (or accept). For the types with RR, there exists a constant $c \in \mathbb{N}$, such that after rewriting the automaton $M$ is allowed to execute at most $c$ jump-right steps. Here we use the prefix FJ to denote the types of fast jumping restarting automata.

Corollary 5

For each type $X \in \{R, RR, RW, RRW, RWW, RRWW\}$, $L(FJX) \subseteq L(JX)$. 
Theorem 6

\[ \text{GCSL} \subseteq \mathcal{L}(\text{FJRWW}). \]
Recognizing Growing Context-Sensitive Languages (GCSL)

**Theorem 6**

GCSL ⊆ $L(\text{FJRWW})$.

**Proof Idea**

- The language class GCSL is characterized by length-reducing *two-pushdown automata* (TPDA for short). [Niemann, Otto, 2005]
- Construct an FJRWW-automaton simulating TPDA.
The language class GCSL is characterized by length-reducing two-pushdown automata (TPDA for short). [Niemann, Otto, 2005]

Construct an FJRW-automaton simulating TPDA.
Recognizing Growing Context-Sensitive Languages (GCSL)

Theorem 6

\[ \text{GCSL} \subseteq \mathcal{L}(\text{FJRWW}). \]

Proof Idea

- The language class GCSL is characterized by length-reducing two-pushdown automata (TPDA for short). [Niemann, Otto, 2005]
- Construct an FJRWW-automaton simulating TPDA.

Lemma 7

Let \( L_{Gl} = \{ w \# w^R \# w \mid w \in \{a, b\}^* \} \) \( \notin \) GCSL. \( L_{Gl} \in \mathcal{L}(\text{FJRRWW}) \).
Theorem 6

GCSL \subseteq \mathcal{L}(FJRW).

Proof Idea

- The language class GCSL is characterized by length-reducing two-pushdown automata (TPDA for short). [Niemann, Otto, 2005]
- Construct an FJRW-automaton simulating TPDA.

Lemma 7

Let \( L_{Gl} = \{ w\#w^R\#w \mid w \in \{a, b\}^* \} \notin \text{GCSL} \). \( L_{Gl} \in \mathcal{L}(FJRRWW) \).

Theorem 8

GCSL \nsubseteq \mathcal{L}(FJRRWW).
Recognizing Church-Rosser Languages (CRL)

Theorem 9

\[
\begin{align*}
\text{CRL} &= \mathcal{L}(\text{det-FJRW}) = \mathcal{L}(\text{det-FJRRWW}) \\
&= \mathcal{L}(\text{det-JRW}) = \mathcal{L}(\text{det-JRRWW}) \\
&= \mathcal{L}(\text{det-RW}) = \mathcal{L}(\text{det-RRWW}).
\end{align*}
\]

Proof Idea

- The language class CRL is characterized by deterministic length-reducing TPDAs. [Niemann, Otto, 2005]
- Use the same simulation technique as in the proof of Theorem 6.
Variants without Auxiliary Symbols

**Definition 10**

\[ L_0 = \{ a^i c a^j b^l \mid i, j, l \geq 0, i + j = l \} \]
\[ \cup \{ a^i d a^j b^l \mid i, j, l \geq 0, 2(i + j) = l \} \].

Note that \( L_0 \) is deterministic context-free.
**Variants without Auxiliary Symbols**

**Definition 10**

\[
L_0 = \{a^i c a^j b^l | i, j, l \geq 0, i + j = l \} \\
\cup \{a^i d a^j b^l | i, j, l \geq 0, 2(i + j) = l \}.
\]

Note that \(L_0\) is deterministic context-free.

**Lemma 11**

\(L_0 \notin \mathcal{L}(\text{FJRRW})\).
Variants without Auxiliary Symbols

Definition 10

\[ L_0 = \left\{ a^i c a^j b^l \mid i, j, l \geq 0, i + j = l \right\} \]
\[ \cup \left\{ a^i d a^j b^l \mid i, j, l \geq 0, 2(i + j) = l \right\}. \]

Note that \( L_0 \) is deterministic context-free.

Lemma 11

\( L_0 \notin \mathcal{L}(FJRRW). \)

Theorem 12

For each type \( X \in \{ R, RR, RW, RRW \} \),

\[ (1) \quad \mathcal{L}(FJX) \subsetneq \mathcal{L}(JX), \]
\[ (2) \quad \mathcal{L}(\text{det}-FJX) \subsetneq \mathcal{L}(\text{det}-JX). \]
Taxonomy of Nondeterministic Variants

\[ \mathcal{L}((J)RW) \quad \mathcal{L}((J)R) \]

\[ \mathcal{L}((J)RW) \quad \mathcal{L}((J)R) \]

\[ \mathcal{L}((J)RRW) \quad \mathcal{L}(FJRWW) \]

\[ \mathcal{L}((J)RR) \quad \mathcal{L}(FJR) \]

\[ \mathcal{L}((J)RR) \quad \mathcal{L}(FJR) \]

\[ \mathcal{L}(FJRW) \quad \mathcal{L}(FJRR) \]

\[ \mathcal{L}(FJRW) \quad \mathcal{L}(FJRR) \]

\[ \mathcal{L}(FJRRW) \quad \mathcal{L}(GCSL) \]

\[ \mathcal{L}(FJRRW) \quad \mathcal{L}(GCSL) \]
Taxonomy of Deterministic Variants

\[ L(\text{det-(J)RW}) \]
\[ L(\text{det-(J)RR}) \]
\[ L(\text{det-(J)RRW}) \]
\[ L(\text{det-(J)RRWW}) \]
\[ L(\text{det-FJRW}) \]
\[ L(\text{det-FJRR}) \]
\[ L(\text{det-FJR}) \]
\[ L(\text{det-FJRRW}) \]
\[ L(\text{det-FJRRWW}) \]

CRL
Monotone Restarting Automata

$q_0$ 

Rewrite

$D_r$ 

read/write-window

Flexible tape

Rewrite

Rewrite

Rewrite

Rewrite

Rewrite
Monotone Restarting Automata

\[ D_r(C_1) \geq D_r(C_2) \geq \cdots \geq D_r(C_n) \]
Results

Lemma 13

$\mathcal{L}(\text{mon-RWW}) \subseteq \mathcal{L}(\text{mon-FJRW})$.
Results

Lemma 13

\[ \mathcal{L}(\text{mon-RWW}) \subseteq \mathcal{L}(\text{mon-FJRWW}). \]

Proof Idea

- Because of the monotony the right distance does not increase from one rewrite step to the next.
- Use the same simulation technique as in the proof of Theorem 6.
Results

Lemma 13

\[ \mathcal{L}(\text{mon-RWW}) \subseteq \mathcal{L}(\text{mon-FJRWW}). \]

Proof Idea

- Because of the monotony the right distance does not increase from one rewrite step to the next.
- Use the same simulation technique as in the proof of Theorem 6.
**Results**

**Lemma 13**

\[ \mathcal{L}(\text{mon-RWW}) \subseteq \mathcal{L}(\text{mon-FJRW}) \].

**Proof Idea**

- Because of the monotony the right distance does not increase from one rewrite step to the next.
- Use the same simulation technique as in the proof of Theorem 6.

**Theorem 14**

\[
\begin{align*}
\text{CFL} &= \mathcal{L}(\text{mon-FJRW}) = \mathcal{L}(\text{mon-FJRRW}) \\
&= \mathcal{L}(\text{mon-JRW}) = \mathcal{L}(\text{mon-JRRW}) \\
&= \mathcal{L}(\text{mon-RWW}) = \mathcal{L}(\text{mon-RRW}).
\end{align*}
\]
Theorem 15

For each type $X \in \{R, RR, RW, RWW, RRWW\}$,

$$
\text{DCFL} = \mathcal{L}(\text{det-mon-FJRW}) = \mathcal{L}(\text{det-mon-FJRWW}) = \mathcal{L}(\text{det-mon-X}) = \mathcal{L}(\text{det-mon-JX}).
$$
Results (Conti.)

**Theorem 15**

*For each type $X \in \{R, RR, RW, RRW, RWW, RRWW\}$,*

\[
\text{DCFL} = \mathcal{L}({\text{det-mon-FJ}RWW}) = \mathcal{L}({\text{det-mon-FJ}RRWW}) = \mathcal{L}({\text{det-mon-X}}) = \mathcal{L}({\text{det-mon-J}X}).
\]

**Theorem 16**

*For each type $X \in \{R, RR, RW, RRW\}$,*

\[
(1) \quad \mathcal{L}({\text{mon-FJ}X}) \subset \mathcal{L}({\text{mon-J}X}),
\]

\[
(2) \quad \mathcal{L}({\text{det-mon-FJ}X}) \subset \mathcal{L}({\text{det-mon-J}R}).
\]
Taxonomy of Monotone Variants

\[
\mathcal{L}(\text{mon-(J)R}) \quad \mathcal{L}(\text{mon-(J)RW}) \quad \mathcal{L}(\text{mon-(J)RWW}) \quad \mathcal{L}(\text{mon-(J)RR}) \quad \mathcal{L}(\text{mon-FJRRW}) \\
\mathcal{L}(\text{mon-FJR}) \quad \mathcal{L}(\text{det-mon-(J)R(R)(W)(W)}) \\
\mathcal{L}(\text{det-mon-FJ(R(W))W}) \quad \mathcal{L}(\text{det-mon-FJR(R)WW}) \\
\mathcal{L}(\text{mon-(J)RRW}) \quad \mathcal{L}(\text{mon-FJRRW}) \\
\mathcal{L}(\text{mon-FJRWW}) \quad \mathcal{L}(\text{mon-FJRW}) \\
\mathcal{L}(\text{mon-FJR}) \quad \mathcal{L}(\text{mon-FJRR}) \\
\mathcal{L}(\text{det-mon-FJR(R)(W)}) \\
\text{CFL} \quad \text{DCFL}
\]
Open Problems

- It is still open whether the inclusion $\text{GCSL} \subseteq \mathcal{L}(\text{FJRWW})$ is proper.
- It remains to derive a characterization of the classes of languages that are computed by the fast jumping restarting automata without auxiliary symbols.
- The inclusion relation between the class REG and the class of languages that are computed by these automata without auxiliary symbols is still unknown.