Jumping Restarting Automata

Qichao Wang, Yongming Li

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

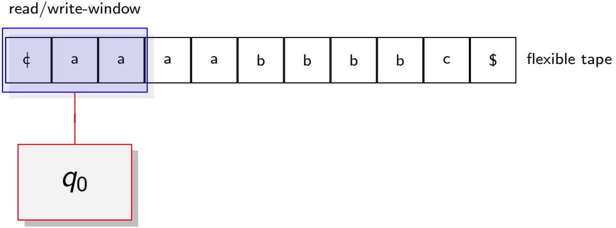
10th Workshop on Non-Classical Models of Automata and Applications

August 21-22, 2018, Košice, Slovakia

Outline

Restarting Automata (RRWW-Automata)

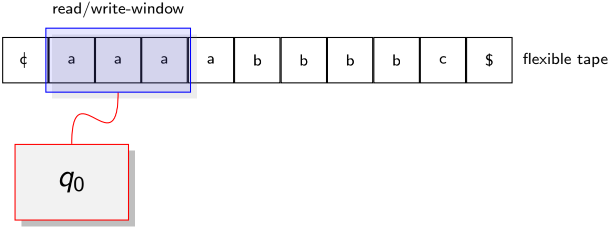
M_1 :



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

Restarting Automata (RRWW-Automata)

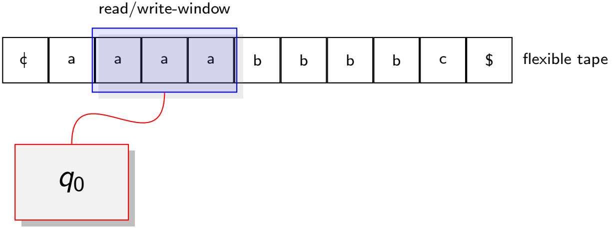
M_1 :



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

Restarting Automata (RRWW-Automata)

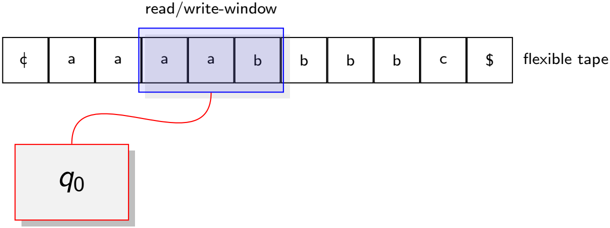
M_1 :



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

Restarting Automata (RRWW-Automata)

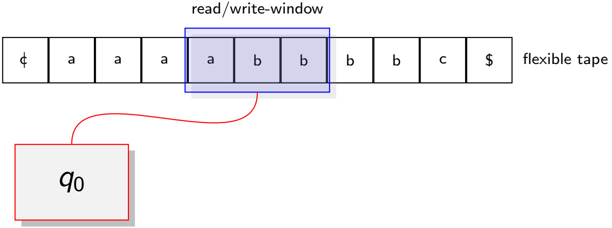
M_1 :



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

Restarting Automata (RRWW-Automata)

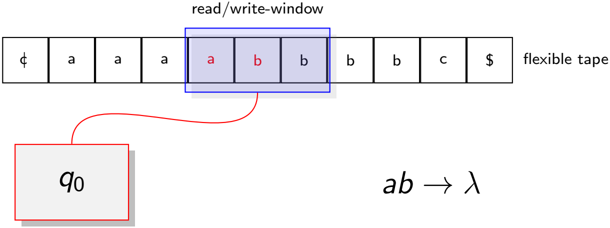
M_1 :



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

Restarting Automata (RRWW-Automata)

M_1 :

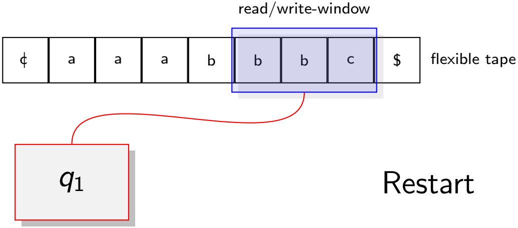


$$ab \to \lambda$$

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

Restarting Automata (RRWW-Automata)

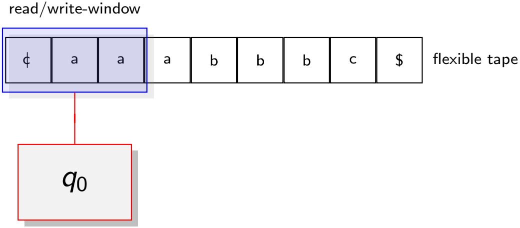
M_1 :



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

Restarting Automata (RRWW-Automata)

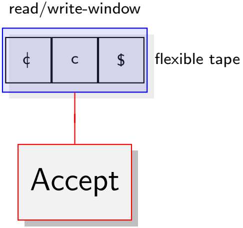
M_1 :



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

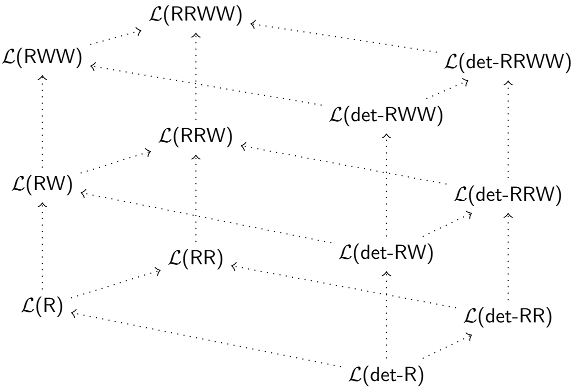
Restarting Automata (RRWW-Automata)

M_1 :



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

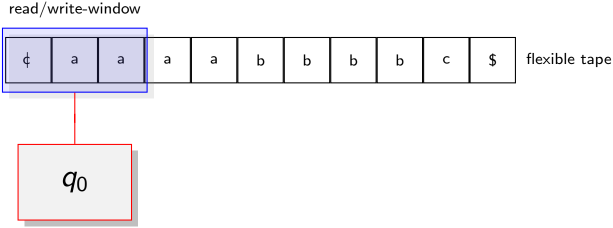
Overview



[Otto, 2006]

Jumping Restarting Automata (JRRWW-Automata)

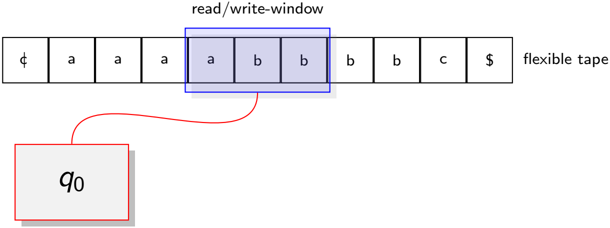
M_2 :



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

Jumping Restarting Automata (JRRWW-Automata)

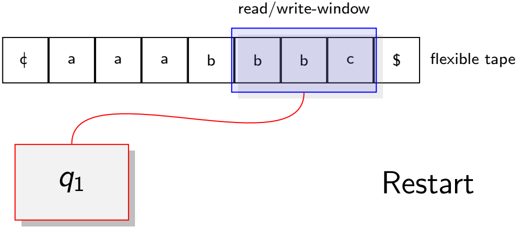
M_2 :



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

Jumping Restarting Automata (JRRWW-Automata)

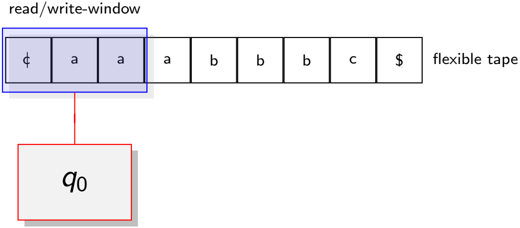
M_2 :



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

Jumping Restarting Automata (JRRWW-Automata)

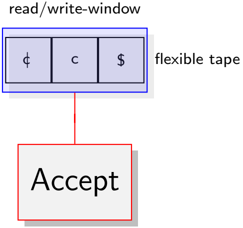
M_2 :



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

Jumping Restarting Automata (JRRWW-Automata)

M_2 :



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

Results

Lemma 1

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

Results

Lemma 1

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

Lemma 2

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

Results

Lemma 1

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

Lemma 2

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

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

Lemma 2

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

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 $\mathsf{X} \in \{\mathsf{R}, \mathsf{RR}, \mathsf{RW}, \mathsf{RRW}, \mathsf{RWW}, \mathsf{RRWW}\}$, $\mathcal{L}(\mathsf{JX}) = \mathcal{L}(\mathsf{X})$.

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

Recognizing Growing Context-Sensitive Languages (GCSL)

Theorem 6

GCSL $\subseteq \mathcal{L}$(FJRWW).

Recognizing Growing Context-Sensitive Languages (GCSL)

Theorem 6

GCSL $\subseteq \mathcal{L}$(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.

Recognizing Growing Context-Sensitive Languages (GCSL)

Theorem 6

GCSL $\subseteq \mathcal{L}$(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.

Z_0	first pushdown	second pushdown	Z_0

Recognizing Growing Context-Sensitive Languages (GCSL)

Theorem 6

GCSL $\subseteq \mathcal{L}$(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 $L_{GI} = \{\, w\#w^R\#w \mid w \in \{a,b\}^* \,\} \notin$ GCSL. $L_{GI} \in \mathcal{L}$(FJRRWW).

Recognizing Growing Context-Sensitive Languages (GCSL)

Theorem 6

GCSL $\subseteq \mathcal{L}$(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. $L_{Gl} \in \mathcal{L}$(FJRRWW).

Theorem 8

GCSL $\subsetneq \mathcal{L}$(FJRRWW).

Recognizing Church-Rosser Languages (CRL)

Theorem 9

$$\begin{aligned} \mathrm{CRL} &= \mathcal{L}(\text{det-FJRWW}) &= \mathcal{L}(\text{det-FJRRWW}) \\ &= \mathcal{L}(\text{det-JRWW}) &= \mathcal{L}(\text{det-JRRWW}) \\ &= \mathcal{L}(\text{det-RWW}) &= \mathcal{L}(\text{det-RRWW}). \end{aligned}$$

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

$$\begin{array}{rcl} L_0 & = & \{\, a^i c a^j b^l \mid i, j, l \geq 0, i + j = l \,\} \\ & \cup & \{\, a^i d a^j b^l \mid i, j, l \geq 0, 2(i + j) = l \,\}. \end{array}$$

Note that L_0 is deterministic context-free.

Variants without Auxiliary Symbols

Definition 10

$$\begin{aligned} L_0 &= \{\, a^i c a^j b^l \mid i,j,l \geq 0, i+j=l \,\} \\ &\cup \{\, a^i d a^j b^l \mid i,j,l \geq 0, 2(i+j)=l \,\}. \end{aligned}$$

Note that L_0 is deterministic context-free.

Lemma 11

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

Variants without Auxiliary Symbols

Definition 10

$$\begin{array}{rcl} L_0 & = & \{\, a^i c a^j b^l \mid i,j,l \geq 0, i+j = l \,\} \\ & \cup & \{\, a^i d a^j b^l \mid i,j,l \geq 0, 2(i+j) = l \,\}. \end{array}$$

Note that L_0 is deterministic context-free.

Lemma 11

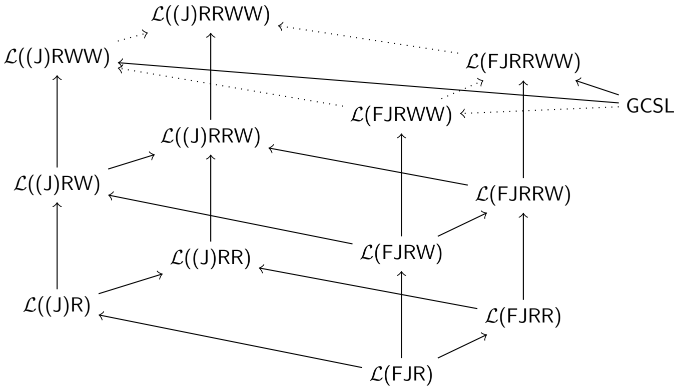
$L_0 \notin \mathcal{L}(\mathrm{FJRRW})$.

Theorem 12

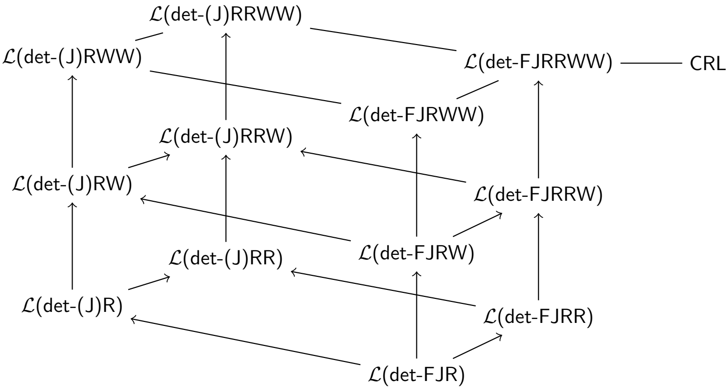
For each type $\mathrm{X} \in \{\mathrm{R}, \mathrm{RR}, \mathrm{RW}, \mathrm{RRW}\}$,

$$\begin{array}{ccc} (1) & \mathcal{L}(\mathrm{FJX}) & \subsetneq & \mathcal{L}(\mathrm{JX}), \\ (2) & \mathcal{L}(\text{det-FJX}) & \subsetneq & \mathcal{L}(\text{det-JX}). \end{array}$$

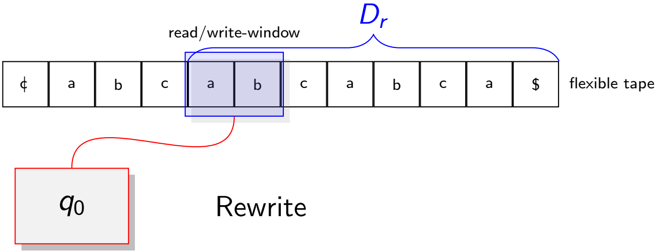
Taxonomy of Nondeterministic Variants



Taxonomy of Deterministic Variants

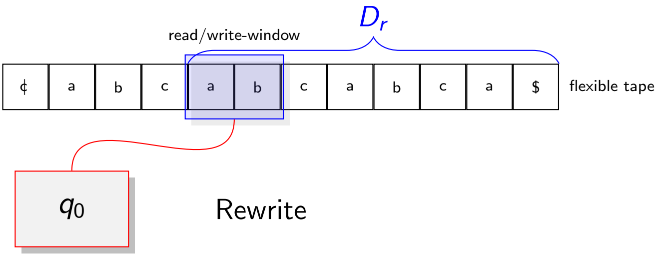


Monotone Restarting Automata



Rewrite

Monotone Restarting Automata



$$D_r(C_1) \geq D_r(C_2) \geq \cdots \geq D_r(C_n)$$

Results

Lemma 13

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

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.

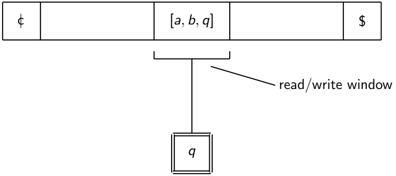
Results

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

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

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.

Theorem 14

$$\begin{aligned} \text{CFL} &= \mathcal{L}(\text{mon-FJRWW}) = \mathcal{L}(\text{mon-FJRRWW}) \\ &= \mathcal{L}(\text{mon-JRWW}) = \mathcal{L}(\text{mon-JRRWW}) \\ &= \mathcal{L}(\text{mon-RWW}) = \mathcal{L}(\text{mon-RRWW}). \end{aligned}$$

Results (Conti.)

Theorem 15

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

$$\begin{aligned} \mathsf{DCFL} &= \mathcal{L}(\text{det-mon-FJRWW}) = \mathcal{L}(\text{det-mon-FJRRWW}) \\ &= \mathcal{L}(\text{det-mon-X}) = \mathcal{L}(\text{det-mon-JX}). \end{aligned}$$

Results (Conti.)

Theorem 15

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

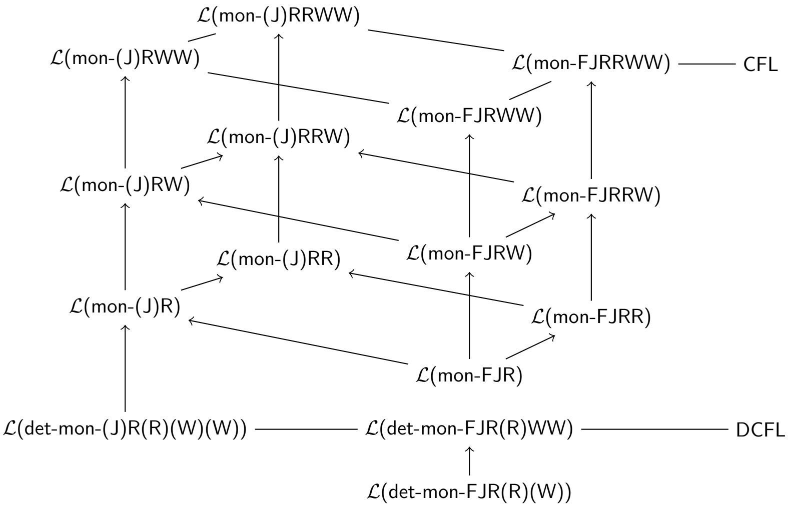
$$\begin{array}{rcccc} \mathsf{DCFL} & = & \mathcal{L}(\text{det-mon-FJRWW}) & = & \mathcal{L}(\text{det-mon-FJRRWW}) \\ & = & \mathcal{L}(\text{det-mon-X}) & = & \mathcal{L}(\text{det-mon-JX}). \end{array}$$

Theorem 16

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

$$\begin{array}{rccc} (1) & \mathcal{L}(\text{mon-FJX}) & \subsetneq & \mathcal{L}(\text{mon-JX}), \\ (2) & \mathcal{L}(\text{det-mon-FJX}) & \subsetneq & \mathcal{L}(\text{det-mon-JR}). \end{array}$$

Taxonomy of Monotone Variants



Open Problems

- It is still open whether the inclusion GCSL $\subseteq \mathcal{L}$(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.