On Some Decision Problems for Stateless Deterministic Ordered Restarting Automata

Kent Kwee, Friedrich Otto

Fachbereich Elektrotechnik/Informatik
Universität Kassel
34109 Kassel, Germany

DCFS 2015

Waterloo, Ontario, Canada

June 25 - 27, 2015

Outline:

1. Introduction

Characterizations of REG

Many different types of automata characterize the class REG:

- the deterministic finite-state acceptor (DFA)
- the nondeterministic finite-state acceptor (NFA)
- the alternating finite-state acceptor (AFA)
- the two-way finite-state acceptor (2NFA)
- the bounded-depth pushdown automaton (bdPDA)
- the R(1)-automaton
- the constant-visit Hennie machine
- the stateless det. ordered restarting automaton (stl-det-ORWW)

1. Introduction

Characterizations of REG

Many different types of automata characterize the class REG:

- the deterministic finite-state acceptor (DFA)
- the nondeterministic finite-state acceptor (NFA)
- the alternating finite-state acceptor (AFA)
- the two-way finite-state acceptor (2NFA)
- the bounded-depth pushdown automaton (bdPDA)
- the R(1)-automaton
- the constant-visit Hennie machine
- the stateless det. ordered restarting automaton (stl-det-ORWW)

1. Introduction

Characterizations of REG

Many different types of automata characterize the class REG:

- the deterministic finite-state acceptor (DFA)
- the nondeterministic finite-state acceptor (NFA)
- the alternating finite-state acceptor (AFA)
- the two-way finite-state acceptor (2NFA)
- the bounded-depth pushdown automaton (bdPDA)
- the R(1)-automaton
- the constant-visit Hennie machine
- the stateless det. ordered restarting automaton (stl-det-ORWW)

1. Introduction

Characterizations of REG

Many different types of automata characterize the class REG:

- the deterministic finite-state acceptor (DFA)
- the nondeterministic finite-state acceptor (NFA)
- the alternating finite-state acceptor (AFA)
- the two-way finite-state acceptor (2NFA)
- the bounded-depth pushdown automaton (bdPDA)
- the R(1)-automaton
- the constant-visit Hennie machine
- the stateless det. ordered restarting automaton (stl-det-ORWW)

1. Introduction

Characterizations of REG

Many different types of automata characterize the class REG:

- the deterministic finite-state acceptor (DFA)
- the nondeterministic finite-state acceptor (NFA)
- the alternating finite-state acceptor (AFA)
- the two-way finite-state acceptor (2NFA)
- the bounded-depth pushdown automaton (bdPDA)
- the R(1)-automaton
- the constant-visit Hennie machine
- the stateless det. ordered restarting automaton (stl-det-ORWW)

Descriptional Complexity

How succinctly do these automata represent a particular language, when we take the number of states as a measure for their size?

Various trade-offs have been established:

- NFA to DFA: 2^n
- AFA to DFA: 2^{2^n} (Chandra, Kozen, Stockmeyer, 1981)
- R(1) to DFA: $2^n + 1$ (Reimann, 2007)
- stl-det-ORWW to DFA: between 2^{2^n} and $2^{2^{O(n^2 \cdot \log n)}}$ (Otto, 2014)

The succinctness of representing regular languages by stl-det-ORWW-automata is quite good, but there is still a huge gap between upper and lower bound.

Descriptional Complexity

How succinctly do these automata represent a particular language, when we take the number of states as a measure for their size?

Various trade-offs have been established:

- NFA to DFA: 2^n
- AFA to DFA: 2^{2^n} (Chandra, Kozen, Stockmeyer, 1981)
- R(1) to DFA: $2^n + 1$ (Reimann, 2007)
- stl-det-ORWW to DFA: between 2^{2^n} and $2^{2^{O(n^2 \cdot \log n)}}$ (Otto, 2014)

The succinctness of representing regular languages by stl-det-ORWW-automata is quite good, but there is still a huge gap between upper and lower bound.

Descriptional Complexity

How succinctly do these automata represent a particular language, when we take the number of states as a measure for their size?

Various trade-offs have been established:

- NFA to DFA: 2^n
- AFA to DFA: 2^{2^n} (Chandra, Kozen, Stockmeyer, 1981)
- R(1) to DFA: $2^n + 1$ (Reimann, 2007)
- stl-det-ORWW to DFA: between 2^{2^n} and $2^{2^{O(n^2 \cdot \log n)}}$ (Otto, 2014)

The succinctness of representing regular languages by stl-det-ORWW-automata is quite good, but there is still a huge gap between upper and lower bound.

Descriptional Complexity

How succinctly do these automata represent a particular language, when we take the number of states as a measure for their size?

Various trade-offs have been established:

- NFA to DFA: 2^n
- AFA to DFA: 2^{2^n} (Chandra, Kozen, Stockmeyer, 1981)
- R(1) to DFA: $2^n + 1$ (Reimann, 2007)
- stl-det-ORWW to DFA: between 2^{2^n} and $2^{2^{O(n^2 \cdot \log n)}}$ (Otto, 2014)

The succinctness of representing regular languages by stl-det-ORWW-automata is quite good, but there is still a huge gap between upper and lower bound.

Descriptional Complexity

How succinctly do these automata represent a particular language, when we take the number of states as a measure for their size?

Various trade-offs have been established:

- NFA to DFA: 2^n
- AFA to DFA: 2^{2^n} (Chandra, Kozen, Stockmeyer, 1981)
- R(1) to DFA: $2^n + 1$ (Reimann, 2007)
- stl-det-ORWW to DFA: between 2^{2^n} and $2^{2^{O(n^2 \cdot \log n)}}$ (Otto, 2014)

The succinctness of representing regular languages by stl-det-ORWW-automata is quite good, but there is still a huge gap between upper and lower bound.

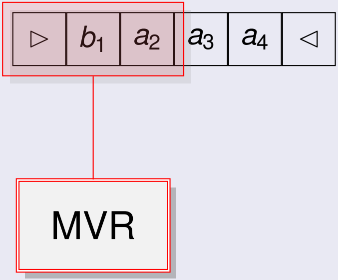
2. Stateless Det. Ordered Restarting Automata

Definition 1

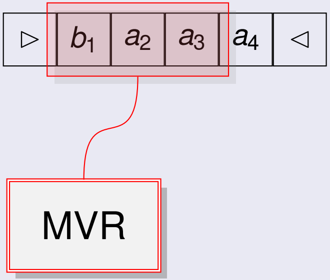
A stateless deterministic ORWW-automaton (stl-det-ORWW) is a one-tape machine that is described by a 6-tuple $M = (\Sigma, \Gamma, \rhd, \lhd, \delta, >)$:

- *Σ is a finite input alphabet,*
- *Γ is a finite tape alphabet containing Σ,*
- *the symbols $\rhd, \lhd \notin \Gamma$ serve as markers for the left and right border of the work space, respectively,*
- *$\delta : (((\Gamma \cup \{\rhd\}) \cdot \Gamma \cdot (\Gamma \cup \{\lhd\})) \cup \{\rhd\lhd\}) \rightarrow \{\mathrm{MVR}\} \cup \Gamma \cup \{\mathrm{Accept}\}$ is the transition function, and*
- *$>$ is a partial ordering on Γ.*

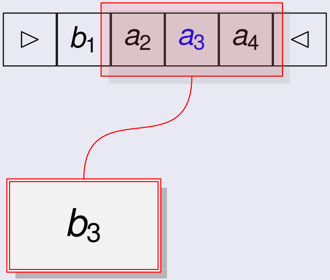
Example for transition function



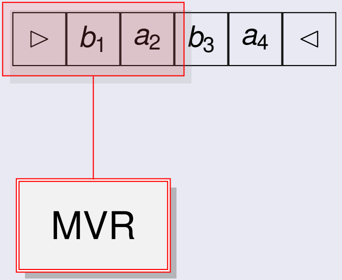
Example for transition function



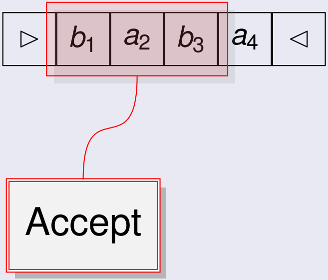
Example for transition function



Example for transition function



Example for transition function



Definition 1 (cont.)

An input $w \in \Sigma^*$ is accepted by M, if there exists a computation of M which starts with the initial configuration $\rhd w \lhd$, and which finally ends with executing an Accept instruction.

$L(M)$ is the language consisting of all words that are accepted by M.

As each cycle ends with a rewrite operation, which replaces a symbol a by a symbol b that is strictly smaller with respect to $>$, each computation of M on input w consists of $\leq |w| \cdot (|\Gamma| - 1)$ many cycles.

Theorem 2 (Mráz, Otto, SOFSEM 2014)

$\mathrm{REG} = \mathcal{L}(\text{stl-det-ORWW}) \subsetneq \mathcal{L}(\text{ORWW})$.

Definition 1 (cont.)

An input $w \in \Sigma^*$ is accepted by M, if there exists a computation of M which starts with the initial configuration $\triangleright w \triangleleft$, and which finally ends with executing an Accept instruction.

$L(M)$ is the language consisting of all words that are accepted by M.

As each cycle ends with a rewrite operation, which replaces a symbol a by a symbol b that is strictly smaller with respect to $>$, each computation of M on input w consists of $\leq |w| \cdot (|\Gamma| - 1)$ many cycles.

Theorem 2 (Mráz, Otto, SOFSEM 2014)

$\mathrm{REG} = \mathcal{L}(\text{stl-det-ORWW}) \subsetneq \mathcal{L}(\text{ORWW})$.

Definition 1 (cont.)

An input $w \in \Sigma^*$ is accepted by M, if there exists a computation of M which starts with the initial configuration $\rhd w \lhd$, and which finally ends with executing an Accept instruction.

$L(M)$ is the language consisting of all words that are accepted by M.

As each cycle ends with a rewrite operation, which replaces a symbol a by a symbol b that is strictly smaller with respect to $>$, each computation of M on input w consists of $\leq |w| \cdot (|\Gamma| - 1)$ many cycles.

Theorem 2 (Mráz, Otto, SOFSEM 2014)

$\mathrm{REG} = \mathcal{L}(\text{stl-det-ORWW}) \subsetneq \mathcal{L}(\text{ORWW})$.

3. Simulating a stl-det ORWW by an NFA

Theorem 1

Let $M = (\Sigma, \Gamma, \rhd, \lhd, \delta_M, >)$ be a stl-det-ORWW-automaton. Then an unambiguous NFA $A = (Q, \Sigma, \Delta_A, q_0, F)$ can be constructed from M such that $L(A) = L(M)$ and $|Q| \in 2^{O(|\Gamma|)}$.

Lemma 2

From a stl-det-ORWW-automaton $M = (\Sigma, \Gamma, \rhd, \lhd, \delta, >)$, one can construct a stl-det-ORWW-automaton $M' = (\Sigma, \Delta, \rhd, \lhd, \delta', >)$ such that $L(M') = L(M)$, $|\Delta| \leq |\Gamma| + 1$, and M' only accepts when its window contains the right sentinel $\lhd$.

3. Simulating a stl-det ORWW by an NFA

Theorem 1

Let $M = (\Sigma, \Gamma, \rhd, \lhd, \delta_M, >)$ be a stl-det-ORWW-automaton. Then an unambiguous NFA $A = (Q, \Sigma, \Delta_A, q_0, F)$ can be constructed from M such that $L(A) = L(M)$ and $|Q| \in 2^{O(|\Gamma|)}$.

Lemma 2

From a stl-det-ORWW-automaton $M = (\Sigma, \Gamma, \rhd, \lhd, \delta, >)$, one can construct a stl-det-ORWW-automaton $M' = (\Sigma, \Delta, \rhd, \lhd, \delta', >)$ such that $L(M') = L(M)$, $|\Delta| \leq |\Gamma| + 1$, and M' only accepts when its window contains the right sentinel $\lhd$.

Proof outline of Theorem 1

$\triangleright$	a_1	a_2	a_3	a_4	$\triangleleft$

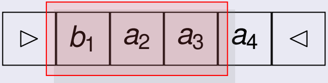
W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	Λ	a_1	Λ	Λ	a_2	Λ	Λ	a_3	Λ	Λ	a_4	Λ	$\triangleleft$

Proof outline of Theorem 1

$\triangleright$	b_1	a_2	a_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	Λ	a_2	Λ	Λ	a_3	Λ	Λ	a_4	Λ	$\triangleleft$
		b_1											

Proof outline of Theorem 1



W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	Λ	a_2	Λ	Λ	a_3	Λ	Λ	a_4	Λ	$\triangleleft$
		b_1											

Proof outline of Theorem 1

$\triangleright$	b_1	a_2	a_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	Λ	a_2	Λ	Λ	a_3	Λ	Λ	a_4	Λ	$\triangleleft$
		b_1											

Proof outline of Theorem 1

$\triangleright$	b_1	a_2	b_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	Λ	a_2	Λ	1	a_3	1	Λ	a_4	Λ	$\triangleleft$
		b_1						b_3					

Proof outline of Theorem 1

$\triangleright$	b_1	a_2	b_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	Λ	a_2	Λ	1	a_3	1	Λ	a_4	Λ	$\triangleleft$
		b_1						b_3					

Proof outline of Theorem 1

$\triangleright$	b_1	b_2	b_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	Λ	a_4	Λ	$\triangleleft$
		b_1			b_2			b_3					

Proof outline of Theorem 1

$\triangleright$	b_1	b_2	b_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	Λ	a_4	Λ	$\triangleleft$
		b_1			b_2			b_3					

Proof outline of Theorem 1

$\triangleright$	c_1	b_2	b_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	Λ	a_4	Λ	$\triangleleft$
	1	b_1	2		b_2			b_3					
		c_1											

Proof outline of Theorem 1

$\triangleright$	c_1	b_2	b_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	Λ	a_4	Λ	$\triangleleft$
	1	b_1	2		b_2			b_3					
		c_1											

Proof outline of Theorem 1

$\triangleright$	c_1	b_2	b_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	Λ	a_4	Λ	$\triangleleft$
	1	b_1	2		b_2			b_3					
		c_1											

Proof outline of Theorem 1

$\triangleright$	c_1	b_2	c_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	Λ	a_4	Λ	$\triangleleft$
	1	b_1	2		b_2		2	b_3	1				
		c_1						c_3					

Proof outline of Theorem 1

$\triangleright$	c_1	b_2	c_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	Λ	a_4	Λ	$\triangleleft$
	1	b_1	2		b_2		2	b_3	1				
		c_1						c_3					

Proof outline of Theorem 1

$\triangleright$	c_1	c_2	c_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	Λ	a_4	Λ	$\triangleleft$
	1	b_1	2	3	b_2	3	2	b_3	1				
		c_1			c_2			c_3					

Proof outline of Theorem 1

$\triangleright$	c_1	c_2	c_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	Λ	a_4	Λ	$\triangleleft$
	1	b_1	2	3	b_2	3	2	b_3	1				
		c_1			c_2			c_3					

Proof outline of Theorem 1

$\triangleright$	c_1	c_2	c_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	Λ	a_4	Λ	$\triangleleft$
	1	b_1	2	3	b_2	3	2	b_3	1				
		c_1			c_2			c_3					

Proof outline of Theorem 1

$\triangleright$	c_1	c_2	c_3	a_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	Λ	a_4	Λ	$\triangleleft$
	1	b_1	2	3	b_2	3	2	b_3	1				
		c_1			c_2			c_3					

Proof outline of Theorem 1

$\triangleright$	c_1	c_2	c_3	b_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	3	a_4	1	$\triangleleft$
	1	b_1	2	3	b_2	3	2	b_3	1		b_4		
		c_1			c_2			c_3					

Proof outline of Theorem 1

$\triangleright$	c_1	c_2	c_3	b_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	3	a_4	1	$\triangleleft$
	1	b_1	2	3	b_2	3	2	b_3	1		b_4		
		c_1			c_2			c_3					

Proof outline of Theorem 1

$\triangleright$	c_1	c_2	c_3	b_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	3	a_4	1	$\triangleleft$
	1	b_1	2	3	b_2	3	2	b_3	1		b_4		
		c_1			c_2			c_3					

Proof outline of Theorem 1

$\triangleright$	c_1	c_2	c_3	b_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	3	a_4	1	$\triangleleft$
	1	b_1	2	3	b_2	3	2	b_3	1		b_4		
		c_1			c_2			c_3					

Proof outline of Theorem 1

$\triangleright$	c_1	c_2	c_3	c_4	$\triangleleft$

W_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3	L_4	W_4	R_4	W_5
$\triangleright$	1	a_1	1	2	a_2	2	1	a_3	1	3	a_4	1	$\triangleleft$
	1	b_1	2	3	b_2	3	2	b_3	1	3	b_4	1	
		c_1			c_2			c_3			c_4		

These sequences can be used to reconstruct the computation of the stl-det-ORWW.

Proof outline (cont.)

There is no valid computation for the following sequences.

L_0	W_0	R_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3
Λ	$\triangleright$	Λ	1	a_1	2	2	a_2	1	Λ	$\triangleleft$	Λ
				b_1			b_2				

Compatibility of two finite non-decreasing sequences of integers

$R = (r_1, \ldots, r_k)$ and $L = (\ell_1, \ldots, \ell_s)$, where $k, s \geq 0$,

$$\mathrm{order}(R, L) = \{\, r_i + i - 1 \mid i = 1, \ldots, k \,\} \cup \{\, \ell_j + j - 1 \mid j = 1, \ldots, s \,\}.$$

(R, L) is called consistent, if $\mathrm{order}(R, L) = \{1, 2, \ldots, k + s\}$.

Proof outline (cont.)

There is no valid computation for the following sequences.

L_0	W_0	R_0	L_1	W_1	R_1	L_2	W_2	R_2	L_3	W_3	R_3
Λ	$\triangleright$	Λ	1	a_1	2	2	a_2	1	Λ	$\triangleleft$	Λ
				b_1			b_2				

Compatibility of two finite non-decreasing sequences of integers

$R = (r_1, \ldots, r_k)$ and $L = (\ell_1, \ldots, \ell_s)$, where $k, s \geq 0$,

$$\mathrm{order}(R, L) = \{\, r_i + i - 1 \mid i = 1, \ldots, k \,\} \cup \{\, \ell_j + j - 1 \mid j = 1, \ldots, s \,\}.$$

(R, L) is called consistent, if $\mathrm{order}(R, L) = \{1, 2, \ldots, k + s\}$.

Proof outline of Theorem 1 (cont.)

As a first step we construct an NFA B for the characteristic language $L_C(M) = \{\, w \in \Gamma^* \mid (\lambda, \triangleright w \triangleleft) \vdash_M^* \text{Accept} \,\}$ of M, which consists of all words over Γ that M accepts.

The set Q contains

- the initial state q_0,
- the final state q_F and
- all pairs of triples of the form $((L_1, W_1, R_1), (L_2, W_2, R_2))$, which satisfy certain conditions.
  - The sequence of letters are ordered by the partial ordering $<$
  - L_i and R_i is a non-decreasing sequence of positive integers.
  - The sequences R_1 and L_2 are consistent.

The transition function Δ_B ensures that all MVR and rewrite instructions exist.

Proof outline of Theorem 1 (cont.)

As a first step we construct an NFA B for the characteristic language $L_C(M) = \{\, w \in \Gamma^* \mid (\lambda, \triangleright w \triangleleft) \vdash_M^* \text{Accept} \,\}$ of M, which consists of all words over Γ that M accepts.

The set Q contains

- the initial state q_0,
- the final state q_F and
- all pairs of triples of the form $((L_1, W_1, R_1), (L_2, W_2, R_2))$, which satisfy certain conditions.
  - The sequence of letters are ordered by the partial ordering $<$
  - L_i and R_i is a non-decreasing sequence of positive integers.
  - The sequences R_1 and L_2 are consistent.

The transition function Δ_B ensures that all MVR and rewrite instructions exist.

Proof outline of Theorem 1 (cont.)

As a first step we construct an NFA B for the characteristic language $L_C(M) = \{\, w \in \Gamma^* \mid (\lambda, \rhd w \lhd) \vdash_M^* \mathrm{Accept} \,\}$ of M, which consists of all words over Γ that M accepts.

The set Q contains

- the initial state q_0,
- the final state q_F and
- all pairs of triples of the form $((L_1, W_1, R_1), (L_2, W_2, R_2))$, which satisfy certain conditions.
  - The sequence of letters are ordered by the partial ordering $<$
  - L_i and R_i is a non-decreasing sequence of positive integers.
  - The sequences R_1 and L_2 are consistent.

The transition function Δ_B ensures that all MVR and rewrite instructions exist.

Proof outline of Theorem 1 (cont.)

As a first step we construct an NFA B for the characteristic language $L_C(M) = \{\, w \in \Gamma^* \mid (\lambda, \triangleright w \triangleleft) \vdash_M^* \mathsf{Accept}\,\}$ of M, which consists of all words over Γ that M accepts.

The set Q contains

- the initial state q_0,
- the final state q_F and
- all pairs of triples of the form $((L_1, W_1, R_1), (L_2, W_2, R_2))$, which satisfy certain conditions.
  - The sequence of letters are ordered by the partial ordering $<$
  - L_i and R_i is a non-decreasing sequence of positive integers.
  - The sequences R_1 and L_2 are consistent.

The transition function Δ_B ensures that all MVR and rewrite instructions exist.

Proof outline (cont.)

- $L_C(M) \subseteq L(B)$:
  - Construct the sequences
  - Obviously they yield an accepting computation of B.
- $L(B) \subseteq L_C(M)$:
  - Extract a computation of M
  - Proof that all conditions are met by induction over the number of rewrites.

Proof outline (cont.)

- $L_C(M) \subseteq L(B)$:
  - Construct the sequences
  - Obviously they yield an accepting computation of B.
- $L(B) \subseteq L_C(M)$:
  - Extract a computation of M
  - Proof that all conditions are met by induction over the number of rewrites.

Proof outline of Theorem 1 (cont.)

$|Q| \in 2^{O(|\Gamma|)}$

The sequence L can be interpreted as multiset. The number of possible sequences L is bounded from above by the expression

$$\sum_{r=0}^{n-1} \binom{n+r-1}{r} \leq \sum_{r=0}^{n-1} \binom{2n}{r} \leq \sum_{r=0}^{2n} \binom{2n}{r} = 2^{2n},$$

and the same is true for the number of possible sequences R.

- at most $2^{2n} \cdot 2^{n} \cdot 2^{2n} = 2^{5n}$ different triples of the form (L, W, R)
- number of states of B is bounded from above by 2^{10n}

Proof outline of Theorem 1 (cont.)

- B is of size $2^{O(n)}$
- obtain an NFA A for the language $L(M) = L_C(M) \cap \Sigma^*$ by simply deleting all transitions from Δ_B that read a letter $x \in (\Gamma \setminus \Sigma)$.
- A is an unambiguous NFA of size $2^{O(n)}$ for $L(A) = L(B) \cap \Sigma^* = L(M)$

$\square$

- $U_n = \{a^{2^n}\}$ is accepted by a stl-det-ORWW-automaton with $3n - 1$ letters
- each NFA for U_n needs at least $2^n + 1$ states
- the bound given in Theorem 1 is sharp up to the O-notation

Corollary 3

For each stl-det-ORWW-automaton M with alphabet of size n, there exists a DFA C of size $2^{2^{O(n)}}$ such that $L(C) = L(M)$ holds.

Proof outline of Theorem 1 (cont.)

- B is of size $2^{O(n)}$
- obtain an NFA A for the language $L(M) = L_C(M) \cap \Sigma^*$ by simply deleting all transitions from Δ_B that read a letter $x \in (\Gamma \smallsetminus \Sigma)$.
- A is an unambiguous NFA of size $2^{O(n)}$ for $L(A) = L(B) \cap \Sigma^* = L(M)$

□

- $U_n = \{a^{2^n}\}$ is accepted by a stl-det-ORWW-automaton with $3n - 1$ letters
- each NFA for U_n needs at least $2^n + 1$ states
- the bound given in Theorem 1 is sharp up to the O-notation

Corollary 3

For each stl-det-ORWW-automaton M with alphabet of size n, there exists a DFA C of size $2^{2^{O(n)}}$ such that $L(C) = L(M)$ holds.

Proof outline of Theorem 1 (cont.)

- B is of size $2^{O(n)}$
- obtain an NFA A for the language $L(M) = L_C(M) \cap \Sigma^*$ by simply deleting all transitions from Δ_B that read a letter $x \in (\Gamma \smallsetminus \Sigma)$.
- A is an unambiguous NFA of size $2^{O(n)}$ for $L(A) = L(B) \cap \Sigma^* = L(M)$

$\square$

- $U_n = \{a^{2^n}\}$ is accepted by a stl-det-ORWW-automaton with $3n - 1$ letters
- each NFA for U_n needs at least $2^n + 1$ states
- the bound given in Theorem 1 is sharp up to the O-notation

Corollary 3

For each stl-det-ORWW-automaton M with alphabet of size n, there exists a DFA C of size $2^{2^{O(n)}}$ such that $L(C) = L(M)$ holds.

4. Decision Problems for stl-det-ORWW-Automata

- The emptiness problem for an NFA $A = (Q, \Sigma, \delta, q_0, F)$ of size $|Q| = m$ is decidable nondeterministically in space $O(\log m)$
- NFA-Emptiness $\in$ DSPACE$((\log |Q|)^2)$ (Savitch's Theorem)

Theorem 4

The emptiness problem for stl-det-ORWW-automata is PSPACE-complete.

Proof

- Let $M = (\Sigma, \Gamma, \triangleright, \triangleleft, \delta, >)$ be a stl-det-ORWW-automaton ($|\Gamma| = n$).
- There exists an NFA A of size $2^{O(n)}$ such that $L(A) = L(M)$.
- We can check emptiness of $L(A)$ deterministically using space $(\log(2^{O(n)}))^2 = O(n^2)$.
- Thus, we see that stl-det-ORWW-Emptiness $\in \mathrm{PSPACE}$.

4. Decision Problems for stl-det-ORWW-Automata

- The emptiness problem for an NFA $A = (Q, \Sigma, \delta, q_0, F)$ of size $|Q| = m$ is decidable nondeterministically in space $O(\log m)$
- NFA-Emptiness $\in$ DSPACE$((\log |Q|)^2)$ (Savitch's Theorem)

Theorem 4

The emptiness problem for stl-det-ORWW-automata is PSPACE-complete.

Proof

- Let $M = (\Sigma, \Gamma, \triangleright, \triangleleft, \delta, >)$ be a stl-det-ORWW-automaton ($|\Gamma| = n$).
- There exists an NFA A of size $2^{O(n)}$ such that $L(A) = L(M)$.
- We can check emptiness of $L(A)$ deterministically using space $(\log(2^{O(n)}))^2 = O(n^2)$.
- Thus, we see that stl-det-ORWW-Emptiness $\in \mathrm{PSPACE}$.

Proof (cont.)

- Let $A_1, \ldots, A_t$ be $t \geq 2$ DFAs over a common input alphabet Σ of size k such that A_i has n_i states, $1 \leq i \leq t$.
- From these DFAs we can construct a stl-det-ORWW-automaton M with a tape aphabet of size $k \cdot (1 + n_1 + \cdots + n_{t-1}) + n_t$ such that $L(M) = \bigcap_{i_1}^{t} L(A_i)$ (Otto, DCFS2014).
- M has at most $O((k \cdot \sum_{i=1}^{t} n_i)^3)$ transitions and can be computed from $A_1, \ldots, A_t$ in polynomial time.
- $L(M) \neq \emptyset$ iff $L(A_1) \cap \cdots \cap L(A_t) \neq \emptyset$
- polynomial-time reduction from the DFA-Intersection-Emptiness Problem (PSPACE-complete) to stl-det-ORWW-Emptiness
- stl-det-ORWW-Emptiness is PSPACE-hard

Corollary 5

For stl-det-ORWW-automata, universality, finiteness, inclusion, and equivalence are PSPACE-complete.

Many subfamilies of REG have been studied:

- nilpotent
- combinatorial
- circular
- suffix-closed
- prefix-closed
- strictly locally testable

Corollary 5

For stl-det-ORWW-automata, universality, finiteness, inclusion, and equivalence are PSPACE-complete.

Many subfamilies of REG have been studied:

- nilpotent
- combinatorial
- circular
- suffix-closed
- prefix-closed
- strictly locally testable

Corollary 5

For stl-det-ORWW-automata, universality, finiteness, inclusion, and equivalence are PSPACE-complete.

Many subfamilies of REG have been studied:

- nilpotent
- combinatorial
- circular
- suffix-closed
- prefix-closed
- strictly locally testable

Definition 6

A language $L \subseteq \Sigma^*$ is strictly k-testable for some $k \geq 1$ if $L \cap \Sigma^k \cdot \Sigma^* = (A \cdot \Sigma^* \cap \Sigma^* \cdot B) \smallsetminus \Sigma^+ \cdot (\Sigma^k \smallsetminus C) \cdot \Sigma^+$ for some finite sets $A, B, C \subseteq \Sigma^k$.

Theorem 7

The following problem is PSPACE-complete for each $k \geq 1$:

INSTANCE:	*A stl-det-ORWW-automaton M.*
QUESTION:	*Is the language $L(M)$ strictly locally k-testable?*

Strictly locally testability is at least PSPACE-hard.

5. Concluding Remarks

- stl-det-ORWW-automata although being deterministic can provide exponentially more succinct representations than NFAs
- Many decision problems of interest are PSPACE-complete.

Some open problems remain:

- Can the given upper bounds be further improved by providing small constants in the exponents?
- Is the problem of deciding whether the language $L(M)$ that is accepted by a given stl-det-ORWW-automaton M is strictly locally testable decidable in polynomial space?
- Characterize $\mathcal{L}$(ORWW)

Thank you for your attention!