NATIONAL RESEARCH UNIVERSITY

On Emptiness and Membership Problems for Set Automata

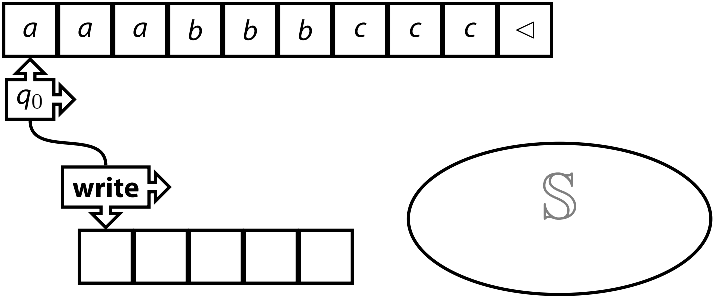
Alexander Rubtsov[1,2] Mikhail Vyalyi[3,1,2]

arubtsov@hse.ru vyalyi@gmail.com

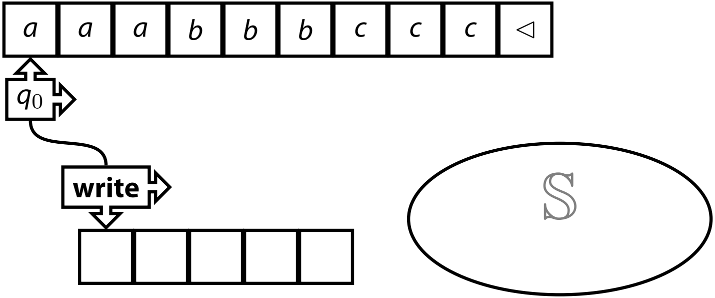
[1]Higher School of Economics
[2]Moscow Institute of Physics and Technology
[3]Dorodnicyn Computing Centre, FRC CSC RAS

CSR 2018

Set Automata



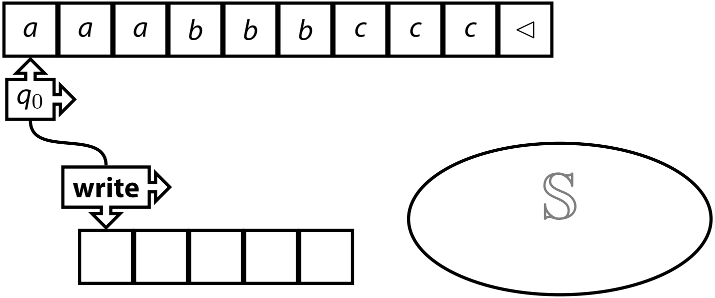
Set Automata



Operations

- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Set Automata



Operations

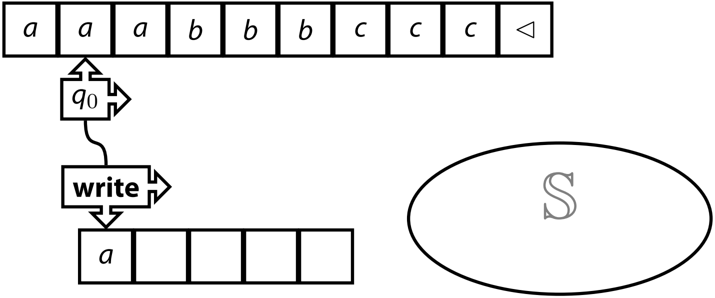
- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata



Operations

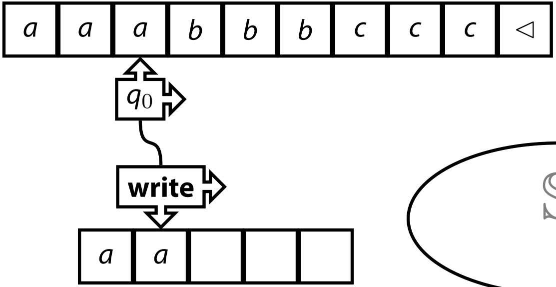
- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

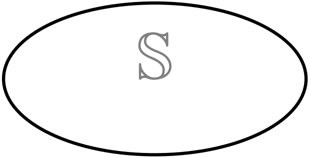
Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata





Operations

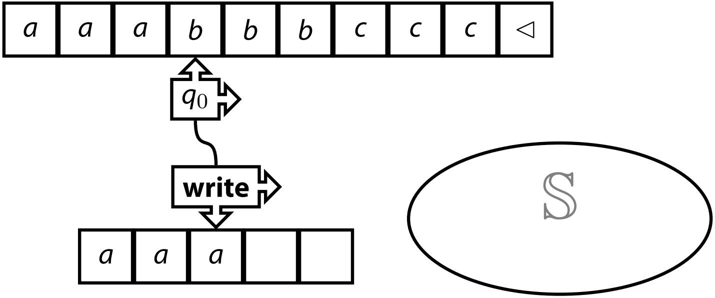
- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata



Operations

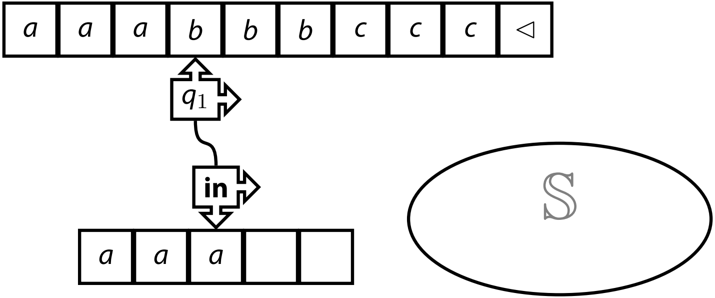
- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata



Operations

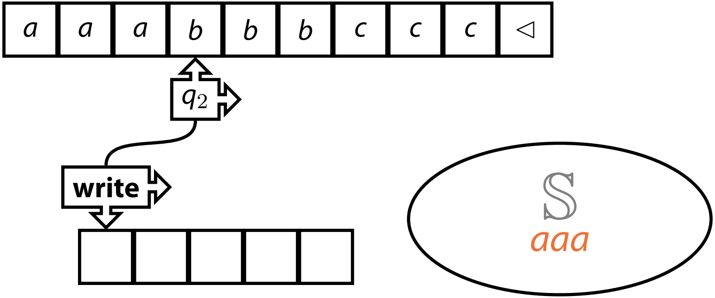
- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata



Operations

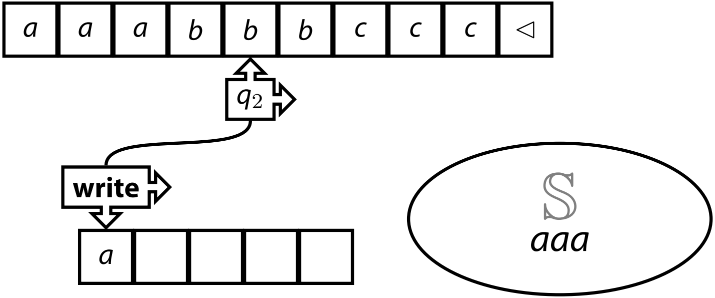
- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata



Operations

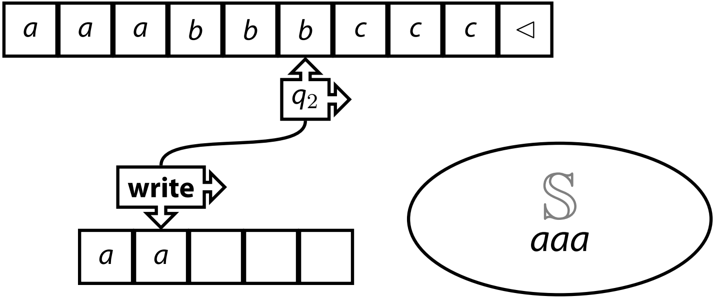
- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata



Operations

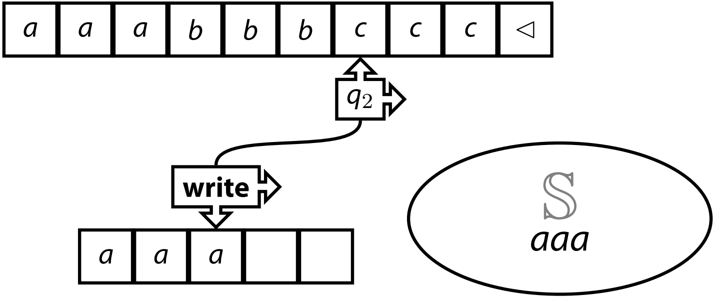
- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata



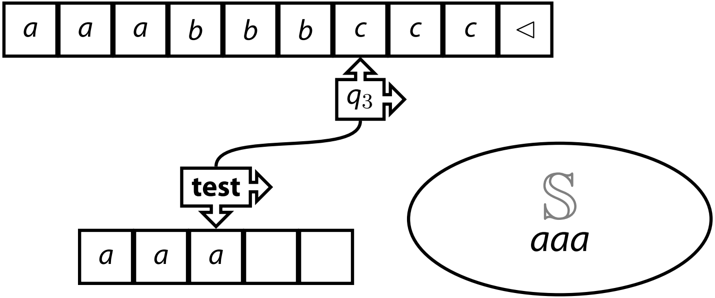
Operations

- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode



Operations

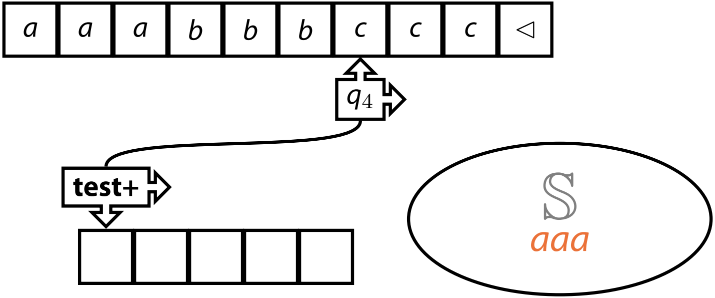
- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata



Operations

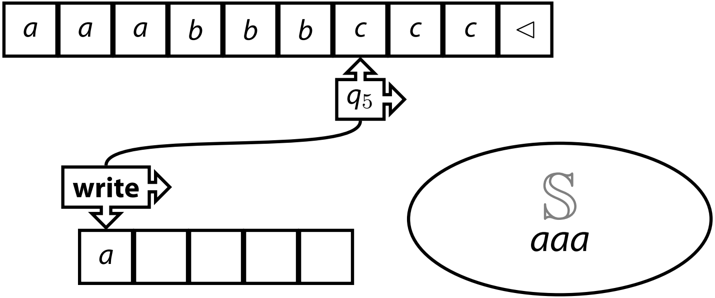
Callback

- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata



Operations

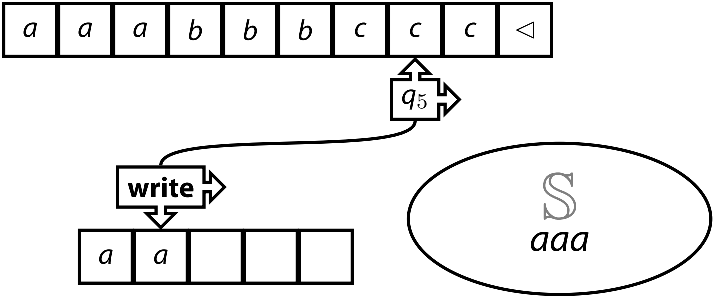
- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata



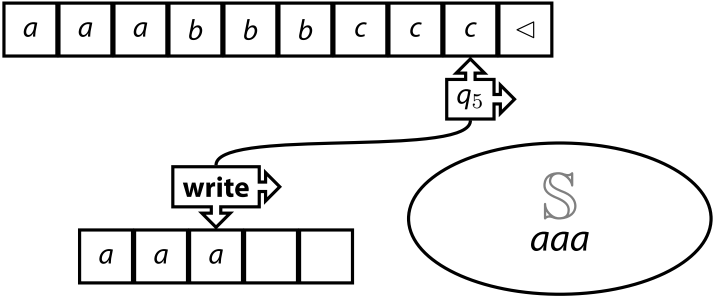
Operations

- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode



Operations

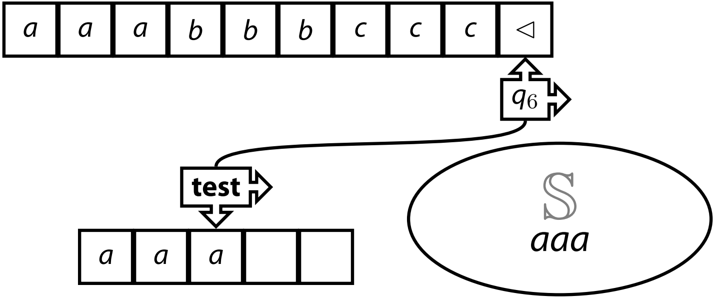
- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata



Operations

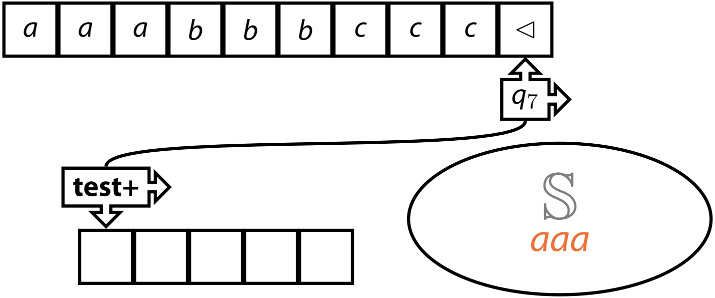
- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata



Operations

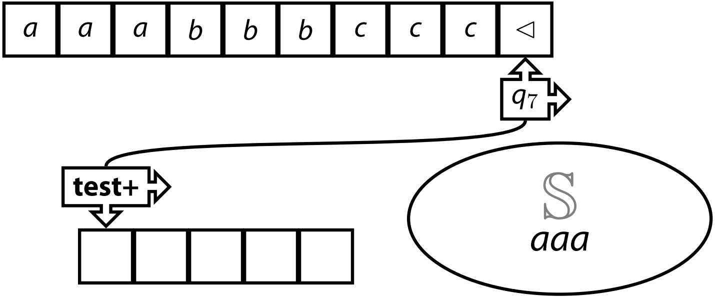
- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set Automata



Operations

- **in** : $\mathbb{S} \to \mathbb{S} \cup \{x\}$
- **out** : $\mathbb{S} \to \mathbb{S} \setminus \{x\}$
- **test** : $x \overset{?}{\in} \mathbb{S}$

Callback

- **test+**: $x \in \mathbb{S}$
- **test-** : $x \notin \mathbb{S}$

- **write** mode

Set automaton M is defined by the tuple

$$M = \langle S,\ \Sigma,\ \Gamma,\ \lhd,\ \delta,\ s_0,\ F\rangle, \text{ where}$$

- S is the finite set of states;
- Σ is the alphabet of the input tape;
- Γ is the alphabet of the work tape;
- $\lhd \notin \Sigma$ is the right endmarker;
- $s_0 \in S$ is the initial state;
- $F \subseteq S$ is the set of accepting states;
- δ is the transition relation:

$$\delta : S \times (\Sigma \cup \{\varepsilon, \lhd\}) \times [S \times (\Gamma^* \cup \{\mathbf{in}, \mathbf{out}\}) \cup S \times \{\mathbf{test}\} \times S]$$

Set automaton M is defined by the tuple

$$M = \langle S,\ \Sigma,\ \Gamma,\ \triangleleft,\ \delta,\ s_0,\ F\rangle, \text{ where}$$

- S is the finite set of states;
- Σ is the alphabet of the input tape;
- Γ is the alphabet of the work tape;
- $\triangleleft \notin \Sigma$ is the right endmarker;
- $s_0 \in S$ is the initial state;
- $F \subseteq S$ is the set of accepting states;
- δ is the transition function (for the deterministic case):

$$\delta : S \times (\Sigma \cup \{\varepsilon, \triangleleft\}) \to [S \times (\Gamma^* \cup \{\mathbf{in}, \mathbf{out}\}) \cup S \times \{\mathbf{test}\} \times S]$$

Known Results for DSA

M. Kutrib, A. Malcher, M. Wendlandt, 2014

Decidability Properties			
	DSA	CFL	DCFL
$L \stackrel{?}{=} \varnothing$	**+**	+	+
$L \stackrel{?}{\in} \text{REG}$	**+**	—	+
$L \stackrel{?}{=} R$	**+**	—	+
$\lvert L\rvert \stackrel{?}{<} \infty$	**+**	+	+

Known Results for DSA

M. Kutrib, A. Malcher, M. Wendlandt, 2014

Closure Properties			
	DSA	CFL	DCFL
$L_1 \cdot L_2$	—	+	—
$L_1 \cup L_2$	—	+	—
$L_1 \cap L_2$	—	—	—
$\Sigma^* \setminus L$	+	—	+
$L \cup R$	+	+	+
$L \cap R$	+	+	+

R stands for a regular language.

Known Results for DSA

M. Kutrib, A. Malcher, M. Wendlandt, 2014

Closure Properties			
	DSA	CFL	DCFL
$L_1 \cdot L_2$	—	+	—
$L_1 \cup L_2$	—	+	—
$L_1 \cap L_2$	—	—	—
$\Sigma^* \setminus L$	+	—	+
$L \cup R$	+	+	+
$L \cap R$	+	+	+

R stands for a regular language.

Theorem

The classes DCFL *and* DSA *are incomparable.*

Complexity Results

- DSA $\subseteq$ **P**
- SA $\subseteq$ **NP**
- Emptiness (SA) is **PSPACE**-hard
- Emptiness (SA) is **NP**-hard
  for 1-ry alphabet of the work tape

- There are **P**- and **NP**-complete languages.

Known results for SA

A.R., M.V. DLT-2017

Complexity Results

- DSA $\subseteq$ **P**
- SA $\subseteq$ **NP**
- Emptiness (SA) is **PSPACE**-hard
- Emptiness (SA) is **NP**-hard
  for 1-ry alphabet of the work tape

Structural Result

- SA is a rational cone

- There are **P**- and **NP**-complete languages.

Known results for SA
A.R., M.V. DLT-2017

Complexity Results

- DSA $\subseteq$ **P**
- SA $\subseteq$ **NP**
- Emptiness (SA) is **PSPACE**-hard $\uparrow$
- Emptiness (SA) is **NP**-hard $\uparrow$
  for 1-ry alphabet of the work tape

Structural Result

- SA is a rational cone

- There are **P**- and **NP**-complete languages.

Lange-Reinhardt Set Automata

K. J. Lange and K. Reinhardt introduced similar model in 1996.

- Only **test** operation
- Word is added in the case **test-**
- No ε-moves

Lange-Reinhardt Set Automata

K. J. Lange and K. Reinhardt introduced similar model in 1996.

- Only **test** operation
- Word is added in the case **test-**
- No ε-moves

Remark

They obtained similar complexity results, but the clue difference is ε-moves.

Main results

- Emptiness (SA)
  - **PSPACE**-hard (A.R., M.V., [DLT'17])
  - in **PSPACE** (this work)

↑ Membership (SA)

- Membership (DSA, 1-ry alphabet of the work tape)
  - **PSPACE**-hard (this work)*
- Membership (DSA without ε-transitions)
  - in **P** [DLT'17]

Main results

- Emptiness (SA)
    - **PSPACE**-hard (A.R., M.V., [DLT'17])
    - in **PSPACE** (this work)

↑ Membership (SA)

- Membership (DSA, 1-ry alphabet of the work tape)
    - **PSPACE**-hard (this work)*
- Membership (DSA without ε-transitions)
    - in **P** [DLT'17]

* Improves the result [DLT'17]:

- Emptiness (SA, 1-ry alphabet of the work tape)
    - **NP**-hard

Main results

- Emptiness (SA)
  - **PSPACE**-hard (A.R., M.V., [DLT'17])
  - in **PSPACE** (this work)

↑ Membership (SA)

- Membership (DSA, 1-ry alphabet of the work tape)
  - **PSPACE**-hard (this work)*
- Membership (DSA without ε-transitions)
  - in **P** [DLT'17]

Main result

Membership and Emptiness problems are **PSPACE**-complete even for DSA with 1-ry alphabet of the work tape.

Membership (DSA) is **PSPACE**-hard

TM simulation by DSA

$k = 3$

TM input tape

1	0	1	0	0	0	1

q

DSA input tape: doesn't matter

$(3, (1, q), \ldots)$

Set content: # ### #######

Membership and Emptiness

Proposition

Membership is polynomially time reduced to non-Emptiness:

$$L(M') = L(M) \cap \{w\}$$

Membership and Emptiness

Proposition

Membership is polynomially time reduced to non-Emptiness:

$$L(M') = L(M) \cap \{w\}$$

Plan

- ✓ Membership is **PSPACE**-hard
- ✓ Membership $\leqslant_{\mathbf{P}}$ non-Emptines

Membership and Emptiness

Proposition

Membership is polynomially time reduced to non-Emptiness:

$$L(M') = L(M) \cap \{w\}$$

Plan

- ✓ Membership is **PSPACE**-hard
- ✓ Membership $\leqslant_{\mathbf{P}}$ non-Emptines
  - ✓↑ non-Emptiness is **PSPACE**-hard

Membership and Emptiness

Proposition

Membership is polynomially time reduced to non-Emptiness:

$$L(M') = L(M) \cap \{w\}$$

Plan

- ✓ Membership is **PSPACE**-hard
- ✓ Membership $\leqslant_{\mathbf{P}}$ non-Emptines
  - ✓↑ non-Emptiness is **PSPACE**-hard
- □ non-Emptiness in **PSPACE**

Membership and Emptiness

Proposition

Membership is polynomially time reduced to non-Emptiness:

$$L(M') = L(M) \cap \{w\}$$

Plan

- ✓ Membership is **PSPACE**-hard
- ✓ Membership $\leqslant_{\mathbf{P}}$ non-Emptines
  - ✓↑ non-Emptiness is **PSPACE**-hard
- ☐ non-Emptiness in **PSPACE**
  - ☐ ↑↑ Emptiness is **PSPACE**-complete

Membership and Emptiness

Proposition

Membership is polynomially time reduced to non-Emptiness:

$$L(M') = L(M) \cap \{w\}$$

Plan

- $\checkmark$ Membership is **PSPACE**-hard
- $\checkmark$ Membership $\leqslant_{\mathbf{P}}$ non-Emptines
  - $\checkmark\uparrow$ non-Emptiness is **PSPACE**-hard
- $\square$ non-Emptiness in **PSPACE**
  - $\square$ $\uparrow\uparrow$ Emptiness is **PSPACE**-complete
  - $\square$ $\uparrow\uparrow$ Membership is **PSPACE**-complete

*All results hold even for DSA with 1-ry alphabet of the work tape!

Regular Realizability Problem

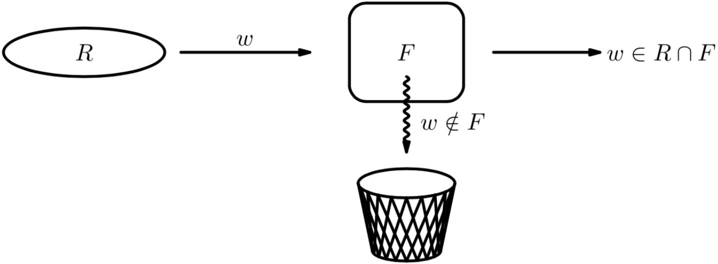
- We fix a language $F \subseteq \Sigma^*$, which we call the filter.

Regular Realizability Problem

- We fix a language $F \subseteq \Sigma^*$, which we call the filter.
- $L(\mathcal{A}) \in \mathrm{REG}$ is the input of the problem, where $\mathcal{A}$ is NFA.

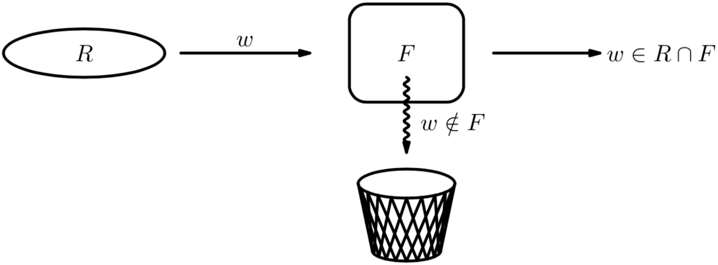
Regular Realizability Problem

- We fix a language $F \subseteq \Sigma^*$, which we call the filter.
- $L(\mathcal{A}) \in$ REG is the input of the problem, where $\mathcal{A}$ is NFA.



Regular Realizability Problem

- We fix a language $F \subseteq \Sigma^*$, which we call the filter.
- $L(\mathcal{A}) \in$ REG is the input of the problem, where $\mathcal{A}$ is NFA.



Defnition

$$\mathrm{NRR}(F) = \{\mathcal{A} \mid \mathcal{A} \in \mathrm{NFA},\ L(\mathcal{A}) \cap F \neq \varnothing\}$$

non-Emptines v.s. Regular Realizability Problem

Lemma (A.R., M.V. DLT'17)

non-Emptiness problem is equivalent to the NRR-problem:

$$\mathsf{NRR}(L(M')) \leqslant_{\log} (L(M) \overset{?}{\neq} \varnothing)$$

non-Emptines v.s. Regular Realizability Problem

Lemma (A.R., M.V. DLT'17)

non-Emptiness problem is equivalent to the NRR-problem:

$$\mathrm{NRR}(L(M')) \leqslant_{\mathrm{log}} (L(M) \overset{?}{\neq} \varnothing) \leqslant_{\mathrm{log}} \mathrm{NRR}(\text{SA-PROT})$$

*We define language SA-PROTof correct protocols on the next slide

Protocols

Definition

- A protocol – is a word of form

$$\#u_1\#\operatorname{op}_1\#u_2\#\operatorname{op}_2\#\cdots\#u_n\#\operatorname{op}_n,$$

where $u_i \in \Gamma^*$, $\# \notin \Gamma$, and $\operatorname{op}_i \in \{\textbf{in}, \textbf{out}, \textbf{test+}, \textbf{test-}\}$.

Protocols

Definition

- A protocol – is a word of form

$$\#u_1\#\,\mathrm{op}_1\#u_2\#\,\mathrm{op}_2\#\cdots\#u_n\#\,\mathrm{op}_n,$$

where $u_i \in \Gamma^*$, $\# \notin \Gamma$, and $\mathrm{op}_i \in \{\textbf{in}, \textbf{out}, \textbf{test+}, \textbf{test-}\}$.

- We say that p is a correct protocol for SA M on an input $w \in L(M)$, if there exists a run of M on the input w such that M performs the operation op_i with the word u_i.

Protocols

Definition

- A protocol – is a word of form

$$\#u_1\#\operatorname{op}_1\#u_2\#\operatorname{op}_2\#\cdots\#u_n\#\operatorname{op}_n,$$

where $u_i \in \Gamma^*$, $\# \notin \Gamma$, and $\operatorname{op}_i \in \{\textbf{in}, \textbf{out}, \textbf{test+}, \textbf{test-}\}$.

- We say that p is a correct protocol for SA M on an input $w \in L(M)$, if there exists a run of M on the input w such that M performs the operation op_i with the word u_i.

- SA-PROT is the language of all correct protocols over the alphabet of the work tape $\Gamma = \{a, b\}$.

Correctness and Support

$\#u\#$**in** $\#v\#$**in** $\#u\#$**test+** $\#v\#$**out** $\#v\#$**test-** $\#w\#$**test-**

standalone

Lemma

A protocol is correct iff each **test+** segment is supported and each **test-** segment is either supported or standalone.

NRR-problem: Steps of the Solution

Protocols' transformation

$$q_0 \xrightarrow[L_{q_0,q_1}]{\#u_1\#\textbf{in}} q_1 \xrightarrow[L_{q_1,q_2}]{\#u_2\#\textbf{in}} q_2 \xrightarrow[L_{q_2,q_3}]{\#u_3\#\textbf{test+}} q_3 \xrightarrow[L_{q_3,q_4}]{\#u_4\#\textbf{out}} q_4 \xrightarrow[L_{q_4,q_5}]{\#u_5\#\textbf{test-}} q_5 \xrightarrow[L_{q_5,q_6}]{\#u_6\#\textbf{test-}} q_6$$

NRR-problem: Steps of the Solution

Protocols' transformation

$$q_0 \xrightarrow[L_{q_0,q_1}]{\#u_1\#\textbf{in}} q_1 \xrightarrow[L_{q_1,q_2}]{\#u_2\#\textbf{in}} q_2 \xrightarrow[L_{q_2,q_3}]{\#u_3\#\textbf{test+}} q_3 \xrightarrow[L_{q_3,q_4}]{\#u_4\#\textbf{out}} q_4 \xrightarrow[L_{q_4,q_5}]{\#u_5\#\textbf{test-}} q_5 \xrightarrow[L_{q_5,q_6}]{\#u_6\#\textbf{test-}} q_6$$

$$\Downarrow$$

$$q_0 \xrightarrow[L_{q_0,q_1}]{\#u'_1\#\textbf{in}} q_1 \xrightarrow[L_{q_1,q_2}]{\#u'_2\#\textbf{in}} q_2 \xrightarrow[L_{q_2,q_3}]{\#u'_3\#\textbf{test+}} q_3 \xrightarrow[L_{q_3,q_4}]{\#u'_4\#\textbf{out}} q_4 \xrightarrow[L_{q_4,q_5}]{\#u'_5\#\textbf{test-}} q_5 \xrightarrow[L_{q_5,q_6}]{\#u'_6\#\textbf{test-}} q_6$$

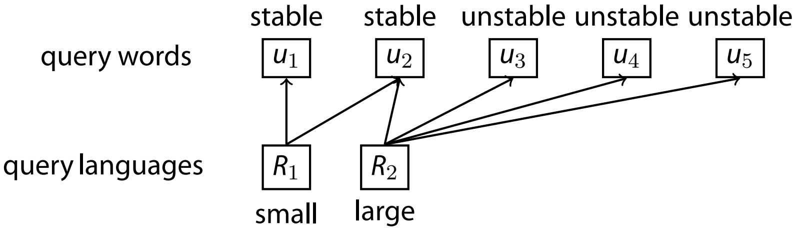
- ✓ $u_i, u'_i \in L_{q_{i-1},q_i}$
- ✓ $|\mathbb{S}|$ is small at each step

- ☐ **PSPACE** algorithm guesses and verifies the modified protocol.

NRR-problem: Steps of the Solution

Verification of the correctness

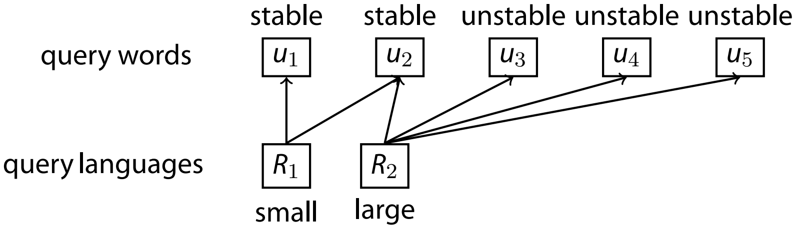
$$q_0 \xrightarrow[L_{q_0,q_1}]{\#u_1\#\mathbf{in}} q_1 \xrightarrow[L_{q_1,q_2}]{\#u_2\#\mathbf{in}} q_2 \xrightarrow[L_{q_2,q_3}]{\#u_3\#\mathbf{test+}} q_3 \xrightarrow[L_{q_3,q_4}]{\#u_4\#\mathbf{out}} q_4 \xrightarrow[L_{q_4,q_5}]{\#u_5\#\mathbf{test-}} q_5 \xrightarrow[L_{q_5,q_6}]{\#u_6\#\mathbf{test-}} q_6$$



NRR-problem: Steps of the Solution

Verification of the correctness

$$q_0 \xrightarrow[L_{q_0,q_1}]{\#u_1\#\textbf{in}} q_1 \xrightarrow[L_{q_1,q_2}]{\#u_2\#\textbf{in}} q_2 \xrightarrow[L_{q_2,q_3}]{\#u_3\#\textbf{test+}} q_3 \xrightarrow[L_{q_3,q_4}]{\#u_4\#\textbf{out}} q_4 \xrightarrow[L_{q_4,q_5}]{\#u_5\#\textbf{test-}} q_5 \xrightarrow[L_{q_5,q_6}]{\#u_6\#\textbf{test-}} q_6$$



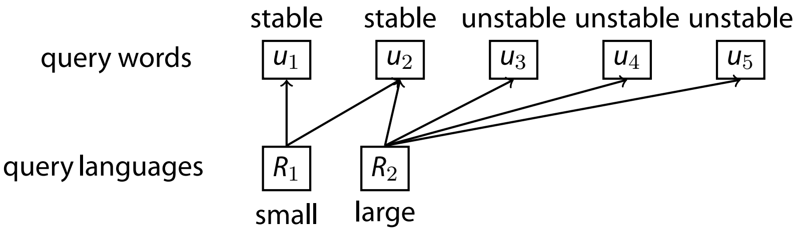
Stable words belong to small languages

Unstable words don’t

NRR-problem: Steps of the Solution

Verification of the correctness

$$q_0 \xrightarrow[L_{q_0,q_1}]{\#u_1\#\textbf{in}} q_1 \xrightarrow[L_{q_1,q_2}]{\#u_2\#\textbf{in}} q_2 \xrightarrow[L_{q_2,q_3}]{\#u_3\#\textbf{test+}} q_3 \xrightarrow[L_{q_3,q_4}]{\#u_4\#\textbf{out}} q_4 \xrightarrow[L_{q_4,q_5}]{\#u_5\#\textbf{test-}} q_5 \xrightarrow[L_{q_5,q_6}]{\#u_6\#\textbf{test-}} q_6$$



Stable words belong to small languages

Unstable words don't

NRR-problem: Steps of the Solution

Protocols' transformation

$$q_0 \xrightarrow[L_{q_0,q_1}]{\#u'_1\#\textbf{in}} q_1 \xrightarrow[L_{q_1,q_2}]{\#u'_2\#\textbf{in}} q_2 \xrightarrow[L_{q_2,q_3}]{\#u'_3\#\textbf{test+}} q_3 \xrightarrow[L_{q_3,q_4}]{\#u'_4\#\textbf{out}} q_4 \xrightarrow[L_{q_4,q_5}]{\#u'_5\#\textbf{test-}} q_5 \xrightarrow[L_{q_5,q_6}]{\#u'_6\#\textbf{test-}} q_6$$

Each L_{q_{i-1},q_i} is either large or small.

We replace u_i by u'_i such a way that

- $u'_i = u_i$ if u^i is stable

NRR-problem: Steps of the Solution

Protocols' transformation

$$q_0 \xrightarrow[L_{q_0,q_1}]{\#u'_1\#\mathbf{in}} q_1 \xrightarrow[L_{q_1,q_2}]{\#u'_2\#\mathbf{in}} q_2 \xrightarrow[L_{q_2,q_3}]{\#u'_3\#\mathbf{test+}} q_3 \xrightarrow[L_{q_3,q_4}]{\#u'_4\#\mathbf{out}} q_4 \xrightarrow[L_{q_4,q_5}]{\#u'_5\#\mathbf{test-}} q_5 \xrightarrow[L_{q_5,q_6}]{\#u'_6\#\mathbf{test-}} q_6$$

Each L_{q_{i-1},q_i} is either large or small.

We replace u_i by u'_i such a way that

- $u'_i = u_i$ if u^i is stable
- all $u'_i \notin \mathbb{S}$ for $\#u_i\#$**test-** and $\#u_i\#$**out** if u_i is unstable

NRR-problem: Steps of the Solution

Protocols' transformation

$$q_0 \xrightarrow[L_{q_0,q_1}]{\#u'_1\#\mathbf{in}} q_1 \xrightarrow[L_{q_1,q_2}]{\#u'_2\#\mathbf{in}} q_2 \xrightarrow[L_{q_2,q_3}]{\#u'_3\#\mathbf{test+}} q_3 \xrightarrow[L_{q_3,q_4}]{\#u'_4\#\mathbf{out}} q_4 \xrightarrow[L_{q_4,q_5}]{\#u'_5\#\mathbf{test-}} q_5 \xrightarrow[L_{q_5,q_6}]{\#u'_6\#\mathbf{test-}} q_6$$

Each L_{q_{i-1},q_i} is either large or small.

We replace u_i by u'_i such a way that

- $u'_i = u_i$ if u^i is stable
- all $u'_i \notin \mathbb{S}$ for $\#u_i\#$**test-** and $\#u_i\#$**out** if u_i is unstable
- for each large L_{q_i,q_j} there is at most one word in $\mathbb{S}$.

Another problem

- u_i may be long.

NRR-problem: Steps of the Solution

Another problem

- u_i may be long. So we describe u_i by the language R_I:

$$R_I = \bigcap_{k \in I} R_k \cap \bigcap_{k \notin I} \overline{R_k}, \quad I \subseteq \{1, 2, \ldots, N\}.$$

NRR-problem: Steps of the Solution

Another problem

- u_i may be long. So we describe u_i by the language R_I:

$$R_I = \bigcap_{k \in I} R_k \cap \bigcap_{k \notin I} \overline{R_k}, \quad I \subseteq \{1, 2, \ldots, N\}.$$

- There are exponentialy many I

NRR-problem: Steps of the Solution

Another problem

- u_i may be long. So we describe u_i by the language R_I:

$$R_I = \bigcap_{k\in I} R_k \cap \bigcap_{k\notin I} \overline{R_k}, \quad I \subseteq \{1, 2, \ldots, N\}.$$

- There are exponentialy many I

NRR-problem: Steps of the Solution

Another problem

- u_i may be long. So we describe u_i by the language R_I:

$$R_I = \bigcap_{k \in I} R_k \cap \bigcap_{k \notin I} \overline{R_k}, \quad I \subseteq \{1, 2, \dots, N\}.$$

- There are exponentialy many I, but not in the modified protocol since the number of unstable words is small.

The Plan is Complete

- ✓ Membership is **PSPACE**-hard
- ✓ Membership $\leqslant_{\mathbf{P}}$ non-Emptines
  - ✓↑ non-Emptiness is **PSPACE**-hard
- ✓ non-Emptiness in **PSPACE**
  - ✓ ↑↑ Emptiness is **PSPACE**-complete
  - ✓ ↑↑ Membership is **PSPACE**-complete

The Plan is Complete

- $\checkmark$ Membership is **PSPACE**-hard
- $\checkmark$ Membership $\leqslant_{\mathbf{P}}$ non-Emptines
  - $\checkmark\uparrow$ non-Emptiness is **PSPACE**-hard
- $\checkmark$ non-Emptiness in **PSPACE**
  - $\checkmark$ $\uparrow\uparrow$ Emptiness is **PSPACE**-complete
  - $\checkmark$ $\uparrow\uparrow$ Membership is **PSPACE**-complete

Thank you!