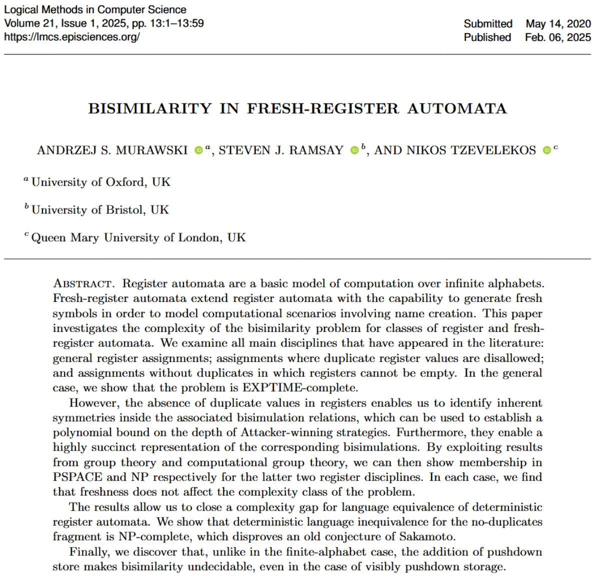
Bismilarity in Fresh-Register Automata

Paul Gauer

20.01.2026


Logical Methods in Computer Science
Volume 21, Issue 1, 2025, pp. 13:1–13:59
https://lmcs.episciences.org/

Submitted May 14, 2020
Published Feb. 06, 2025

BISIMILARITY IN FRESH-REGISTER AUTOMATA

ANDRZEJ S. MURAWSKI [a], STEVEN J. RAMSAY [b], AND NIKOS TZEVELEKOS [c]

[a] University of Oxford, UK

[b] University of Bristol, UK

[c] Queen Mary University of London, UK

Abstract. Register automata are a basic model of computation over infinite alphabets. Fresh-register automata extend register automata with the capability to generate fresh symbols in order to model computational scenarios involving name creation. This paper investigates the complexity of the bisimilarity problem for classes of register and fresh-register automata. We examine all main disciplines that have appeared in the literature: general register assignments; assignments where duplicate register values are disallowed; and assignments without duplicates in which registers cannot be empty. In the general case, we show that the problem is EXPTIME-complete.

However, the absence of duplicate values in registers enables us to identify inherent symmetries inside the associated bisimulation relations, which can be used to establish a polynomial bound on the depth of Attacker-winning strategies. Furthermore, they enable a highly succinct representation of the corresponding bisimulations. By exploiting results from group theory and computational group theory, we can then show membership in PSPACE and NP respectively for the latter two register disciplines. In each case, we find that freshness does not affect the complexity class of the problem.

The results allow us to close a complexity gap for language equivalence of deterministic register automata. We show that deterministic language inequivalence for the no-duplicates fragment is NP-complete, which disproves an old conjecture of Sakamoto.

Finally, we discover that, unlike in the finite-alphabet case, the addition of pushdown store makes bisimilarity undecidable, even in the case of visibly pushdown storage.

[1]

(Fresh-)Register Automata

Register Automata

Definition

An r-register automaton is a tuple $\mathcal{A} = (Q, \Sigma, \delta)$ where:

- Q is a set of states
- Σ is a finite alphabet
- δ is a transition function

$\mathcal{A}$ additionally has r registers it can access during a computation

Register Automata (RA)

Register Assignments

As input $\mathcal{A}$ reads tuples $(t, a) \in (\Sigma \times \mathcal{D})$ where $\mathcal{D}$ is an infinite alphabet

A register assignment is a function $\rho : \{1, ..., r\} \to \mathcal{D} \cup \{\#\}$

- ρ defines which value from $\mathcal{D}$ is stored in the register with index $i \in \{1, ..., r\}$
- If a register is empty, it is mapped to $\#$

Register Automata (RA)

Transitions

$\delta : Q \times \Sigma \times \mathcal{P}(\{1, ..., r\}) \times \{0, ..., r\} \times \mathcal{P}(\{1, ..., r\}) \to Q$

A transition, written $q \xrightarrow{t,X,i,Z} q'$, is applicable on input (t', a) if:

- $t = t'$
- X is the set of register indices so that $\rho(x) = a \Leftrightarrow x \in X$

Then:

- a is written into register i. (No write if $i = 0$)
- The content of the registers corresponding to the indices in Z is deleted. ($\rho(z)$ is set to $\#$ for all $z \in Z$)

Fresh-Register Automata (FRA)

Fresh Transitions

- For fresh-register automata additionally $X = \circledast$ is allowed.
- A transition $q \xrightarrow{t,\circledast,i,Z} q'$ is only applicable for inputs (t, a) where a is fresh. Meaning this is the first time that a appears in an input.
- (Register automata are a special case of fresh-register automata, where always $X \neq \circledast$.)

Bisimilarity

Labelled Transition Systems (LTS)

Definition

An LTS is a tuple $S = (\mathbb{C}, \mathit{Act}, \rightarrow)$ where:

- $\mathbb{C}$ is a set of configurations
- Act is a set of labels
- $\rightarrow$ is a transition relation of type $\mathbb{C} \times \mathit{Act} \times \mathbb{C}$

LTS for Fresh-Register Automata

Configurations

A configuration of an FRA $\mathcal{A}$ is a tuple (q, ρ, H) where:

- q is the current state
- ρ is the current register assignment
- H is the history of names read as input (only for FRA)

$\mathbb{C}_A$ denotes the set of all configurations of $\mathcal{A}$

If $\mathcal{A}$ takes a transition $q \xrightarrow{t,X,i,Z} q'$ reading input (t, d) then it passes from (q, ρ, H) to (q', ρ', H') where:

- ρ' is obtained from ρ by setting $\rho'(i) = d$ and $\rho'(z) = \#$ for all $z \in Z$
- $H' = H \cup \{d\}$

The LTS is then given by $S(\mathcal{A}) = (\mathbb{C}_{\mathcal{A}}, \Sigma \times \mathcal{D}, \to_A)$, where $\to_A$ the induced transition relation.

Bisimilarity

Definition:

A bisimulation is a relation $R \subset \mathbb{C} \times \mathbb{C}$ where for each $(c_1, c_2) \in R$ and each $\ell \in Act$:

- if $c_1 \xrightarrow{\ell} c_1'$, then there is some $c_2 \xrightarrow{\ell} c_2'$ with $(c_1', c_2') \in R$
- if $c_2 \xrightarrow{\ell} c_2'$, then there is some $c_1 \xrightarrow{\ell} c_1'$ with $(c_1', c_2') \in R$

Bisimulation Game:

- Attacker takes any valid transition from one of the current configurations
- Defender has to match this transition (same label) for the respective other configuration.

Problems regarding Bisimilarity in (Fresh-) Register Automata

Restrictions on the Register Assignment

Single (S) and Multiple (M) Assignments

Register assignment can either be Single (S) or Multiple (M)

- S : If a register is nonempty, its content has to be different to the content of all other registers
- M : Two different registers can hold the same content

Policies for empty registers

There are three different policies ($F/\#_0/\#$):

- F : All registers have to be filled at all times.
- $\#_0$: Registers can be initially empty.
- $\#$: The content of a register can be deleted

Bisimilarity Problems in (Fresh-) Register Automata

Let $X \in \{S, M\}$ and $Y \in \{F, \#_0, \#\}$.

$\sim FRA(XY)$

- Given: *FRA* with a register assignment policy that obeys the restrictions imposed by X and Y, two configurations $\kappa_1 = (q_1, \rho_1, H)$ and $\kappa_2 = (q_2, \rho_2, H)$
- Question: is $\kappa_1 \sim \kappa_2$?

$\sim RA(XY)$

- Given: *RA* with a register assignment policy that obeys the restrictions imposed by X and Y, two configurations κ_1 and κ_2
- Question: is $\kappa_1 \sim \kappa_2$?

Bisimilarity Problems in (Fresh-) Register Automata

$\sim FRA(SF)$	$\sim FRA(S\#_0)$	$\sim FRA(S\#)$	$\sim FRA(MF)$	$\sim FRA(M\#_0)$	$\sim FRA(M\#)$
$\sim RA(SF)$	$\sim RA(S\#_0)$	$\sim RA(S\#)$	$\sim RA(MF)$	$\sim RA(M\#_0)$	$\sim RA(M\#)$

Bisimilarity Problems in (Fresh-) Register Automata

$$\begin{array}{ccccccccccc}
\sim FRA(SF) & \leq & \sim FRA(S\#_0) & \leq & \sim FRA(S\#) & \leq & \sim FRA(MF) & \leq & \sim FRA(M\#_0) & \leq & \sim FRA(M\#) \\
\rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} \\
\sim RA(SF) & \leq & \sim RA(S\#_0) & \leq & \sim RA(S\#) & \leq & \sim RA(MF) & \leq & \sim RA(M\#_0) & \leq & \sim RA(M\#)
\end{array}$$

Bisimilarity Problems in (Fresh-) Register Automata

$$
\begin{array}{ccccccccccccc}
\text{NP-solvable} & & \text{PSPACE-Complete} & & & & & & \text{EXPTIME-Complete} & & & & \\
\sim FRA(SF) & \leq & \sim FRA(S\#_0) & \leq & \sim FRA(S\#) & \leq & \sim FRA(MF) & \leq & \sim FRA(M\#_0) & \leq & \sim FRA(M\#) \\
\rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} \\
\sim RA(SF) & \leq & \sim RA(S\#_0) & \leq & \sim RA(S\#) & \leq & \sim RA(MF) & \leq & \sim RA(M\#_0) & \leq & \sim RA(M\#)
\end{array}
$$

Alternating Linear Bounded Automata

Alternating Linear Bounded Automata

Definition

An alternating linear bounded automaton (ALBA) is a tuple $\mathcal{M} = \langle \Gamma, Q_\forall, Q_\exists, q_0, q_{acc}, q_{rej}, \delta \rangle$ where:

- Γ is a finite alphabet containing end of tape markers $\triangleright, \triangleleft \in \Gamma$
- $Q_\forall$ is a set of universal states
- $Q_\exists$ is a set of existential states
- $q_{acc} \neq q_{rej} \notin Q_\forall \cup Q_\exists$
- $\delta : (Q \setminus \{q_{acc}, q_{rej}\}) \times \Gamma \to P(Q \times \Gamma \times \{1, +1\})$ is a transition function which forbids moving beyond the end of tape markers

(Q refers to $(Q_\forall \uplus Q_\exists \uplus q_{acc} \uplus q_{rej})$)

Alternating Linear Bounded Automata

Computation Trees

A computation tree has configurations (q, k, t) of $\mathcal{M}$ as nodes and

- is rooted at $(q_0, 0, \triangleright w \triangleleft)$
- if $q \in Q_\forall$ then (q, k, t) has one child for each successor configuration
- if $q \in Q_\exists$ then (q, k, t) has exactly one child, which is any possible successor configuration
- The tree is accepting if it is finite and all leaves are of type (q_{acc}, k, t)

$\mathcal{M}$ accepts input $w \in \Gamma \setminus \{\triangleright, \triangleleft\}$ if there is an accepting computation tree w.

EXPTIME-Complete Bisimilarity Problems

EXPTIME-Complete Bisimilarity Problems

$$\begin{array}{ccccccc}
\sim FRA(S\#) & \leq & \sim FRA(MF) & \leq & \sim FRA(M\#_0) & \leq & \sim FRA(M\#) \\
\rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} \\
\sim RA(S\#) & \leq & \sim RA(MF) & \leq & \sim RA(M\#_0) & \leq & \sim RA(M\#)
\end{array}$$

EXPTIME-Complete Bisimilarity Problems

$$\begin{array}{ccccccc}
 & & & & & & \in \text{EXPTIME} \\
\sim FRA(S\#) & \leq & \sim FRA(MF) & \leq & \sim FRA(M\#_0) & \leq & \boxed{\sim FRA(M\#)} \\
\rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} \\
\sim RA(S\#) & \leq & \sim RA(MF) & \leq & \sim RA(M\#_0) & \leq & \sim RA(M\#)
\end{array}$$

EXPTIME-Complete Bisimilarity Problems

$$\begin{tikzcd}
 & & & \in \text{EXPTIME} \\
\sim FRA(S\#) \arrow[r, phantom, "\leq"] & \sim FRA(MF) \arrow[r, phantom, "\leq"] & \sim FRA(M\#_0) \arrow[r, phantom, "\leq"] & \boxed{\sim FRA(M\#)} \\
\boxed{\sim RA(S\#)} \arrow[u, phantom, sloped, "\leq"] \arrow[r, phantom, "\leq"] & \sim RA(MF) \arrow[u, phantom, sloped, "\leq"] \arrow[r, phantom, "\leq"] & \sim RA(M\#_0) \arrow[u, phantom, sloped, "\leq"] \arrow[r, phantom, "\leq"] & \sim RA(M\#) \arrow[u, phantom, sloped, "\leq"] \\
\text{EXPTIME-hard} & & &
\end{tikzcd}$$

EXPTIME-Complete Bisimilarity Problems

$$\begin{tikzcd}
 & & & & & & \in \text{EXPTIME} \\
\sim FRA(S\#) \arrow[r, phantom, "\leq"] \arrow[d, phantom, "\rotatebox{90}{$\geq$}"] & \sim FRA(MF) \arrow[r, phantom, "\leq"] \arrow[d, phantom, "\rotatebox{90}{$\geq$}"] & \sim FRA(M\#_0) \arrow[r, phantom, "\leq"] \arrow[d, phantom, "\rotatebox{90}{$\geq$}"] & \boxed{\sim FRA(M\#)} \arrow[d, phantom, "\rotatebox{90}{$\geq$}"] \\
\boxed{\sim RA(S\#)} \arrow[r, phantom, "\leq"] & \sim RA(MF) \arrow[r, phantom, "\leq"] & \sim RA(M\#_0) \arrow[r, phantom, "\leq"] & \sim RA(M\#) \\
\text{EXPTIME-hard}
\end{tikzcd}$$

$\Rightarrow$ EXPTIME-Completeness

$\sim FRA(M\#) \in$ EXPTIME

Reminder $\sim FRA(M\#)$:

- fresh register automaton
- different registers can hold same value
- register content can be deleted

General idea:

- Use a finite subset $N \subseteq \mathcal{D}$ with $|N| = 2r + 2$ to capture the full bisimulation.
- Determine the winner of this bounded bisimulation in alternating polynomial time and use the fact that $APSPACE = EXPTIME$

$\sim$ $FRA(M\#) \in$ EXPTIME

Given (q_1, ρ_1, H) and (q_2, ρ_2, H)

For transitions in a bisimulation only the information

- which sets of registers from ρ_1 and ρ_2 contain the same data
- is $H \subseteq (rng(\rho_1) \cup rng(\rho_2))$

is of importance

$\sim FRA(M\#) \in$ EXPTIME

- Let $\langle A, (q_1, \rho_1, H_0), (q_2, \rho_2, H_0)\rangle$ be an instance of $\sim FRA(M\#)$
- Choose a set $N = \{d_1, ..., d_{2r+2}\}$ with:
  - $N \subseteq H_0$, if $|H_0| < 2r + 2$
  - $rng(\rho_1) \cup rng(\rho_2) \subseteq N \subseteq H_0$, otherwise

$\sim FRA(M\#) \in$ EXPTIME

To do so, we define N-Bisimilarity for Configurations from:

$$\mathbb{C}_{\mathcal{A},\mathcal{N}} = \{(q, \rho, H) \in \mathbb{C}_A | H \subsetneq N\}$$

that handle histories which are trimmed to contain at most $2r + 1$ elements:

$$\lceil H \rceil^{N}_{\rho_1,\rho_2} = \begin{cases} H & \text{if } H \subsetneq N \\ H \setminus \{\min(N \setminus (rng(\rho_1) \cup rng(\rho_2)))\} & \text{if } H = N \end{cases}$$

$\sim$ $FRA(M\#) \in$ EXPTIME

N-Bisimilarity

N-Bisimilarity is a relation $\sim_N \subseteq \mathbb{C}_{\mathcal{A},\mathcal{N}} \times \mathbb{C}_{\mathcal{A},\mathcal{N}}$ where for all $(q_1, \rho_1, H_1) \sim_N (q_2, \rho_2, H_2)$:

- $H_1 = H_2$
- for all transitions $(q_1, \rho_1, H_1) \xrightarrow{(t,d)} (q_1', \rho_1', H_1')$ with $d \in N$, where either:
  - $d \in rng(\rho_1) \cup rng(\rho_2)$
  - $rng(\rho_1) \cup rng(\rho_2) \subsetneq H$ and $d = \min(H \setminus (rng(\rho_1) \cup rng(\rho_2)))$
  - $d = \min(N \setminus H)$

  there is $(q_2, \rho_2, H_2) \xrightarrow{(t,d)} (q_2', \rho_2', H_2')$ and $(q_1', \rho_1', \lceil H_1' \rceil^N_{\rho_1,\rho_2}) \sim_N (q_2', \rho_2', \lceil H_2' \rceil^N_{\rho_1,\rho_2})$

$\sim FRA(M\#) \in$ EXPTIME

Lemma:

$(q_1, \rho_1, H_0) \sim (q_2, \rho_2, H_0) \Leftrightarrow (q_1, \rho_1, \hat{H}_0) \sim_N (q_2, \rho_2, \hat{H}_0)$ where $\hat{H}_0 = \lceil H_0 \cap N \rceil^N_{\rho_1,\rho_2}$

$$\mathcal{D} = (rng(\rho_1) \cup rng(\rho_2)) \uplus (H \setminus (rng(\rho_1) \cup rng(\rho_2))) \uplus (D \setminus H)$$

$\Rightarrow$ Every transition $(q, \rho, H) \xrightarrow{(t, d \in \mathcal{D})} (q', \rho', H')$ in the bisimulation game can be captured by a transition $(t, d' \in N)$ in the N-bisimulation game:

- if $d \in rng(\rho_1) \cup rng(\rho_2)$ this is clear
- if $d \in H \setminus (rng(\rho_1) \cup rng(\rho_2))$ then $d' = \min(\hat{H} \setminus (rng(\rho_1) \cup rng(\rho_2)))$
- if $d \in \mathcal{D} \setminus H$ then $d' = \min(N \setminus \hat{H})$

$\sim FRA(M\#) \in$ EXPTIME

$(q_{01}, \rho_{01}, \hat{H}) \not\sim_N (q_{02}, \rho_{02}, \hat{H})$ can be decided in *APSPACE* by the following algorithm:

```
1: $(q_1, \rho_1, q_2, \rho_2, H) \leftarrow (q_{01}, \rho_{01}, q_{02}, \rho_{02}, \hat{H}_0)$
2: **repeat**
3:     Existentially choose $i \in \{1, 2\}$ and a valid transition $(q_i, \rho_i, H) \xrightarrow{(t,d)} (q'_i, \rho'_i, H')$ or return FALSE, if no such transition exists
4:     Universally choose a valid transition $(q_{3-i}, \rho_{3-i}, H) \xrightarrow{(t,d)} (q'_{3-i}, \rho'_{3-i}, H')$ or return TRUE, if no such transition exists
5:     $(q_1, \rho_1, q_2, \rho_2, H) \leftarrow (q'_1, \rho'_1, q'_2, \rho'_2, \lceil H' \rceil^N_{\rho'_1, \rho'_2})$
6: **until** termination
```

The algorithm accepts as soon as the defender cannot defend himself, meaning the two configurations are not bisimilar. The space used is in $\mathcal{O}(2r(\log(2r+2)) + (2r+2) + \log(|Q|))$

EXPTIME-Complete Bisimilarity Problems

$$\begin{array}{ccccccc}
 & & & & & & \in \text{EXPTIME} \checkmark \\
\sim FRA(S\#) & \leq & \sim FRA(MF) & \leq & \sim FRA(M\#_0) & \leq & \boxed{\sim FRA(M\#)} \\
\rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} \\
\sim RA(S\#) & \leq & \sim RA(MF) & \leq & \sim RA(M\#_0) & \leq & \sim RA(M\#)
\end{array}$$

EXPTIME-Complete Bisimilarity Problems

$$\begin{tikzcd}
 & & & & & & \color{red}{\in \text{EXPTIME} \checkmark} \\
\sim FRA(S\#) \arrow[r, phantom, "\leq"] & \sim FRA(MF) \arrow[r, phantom, "\leq"] & \sim FRA(M\#_0) \arrow[r, phantom, "\leq"] & \boxed{\sim FRA(M\#)} \\
\sim RA(S\#) \arrow[u, phantom, "\rotatebox{90}{$\leq$}"] \arrow[r, phantom, "\leq"] & \sim RA(MF) \arrow[u, phantom, "\rotatebox{90}{$\leq$}"] \arrow[r, phantom, "\leq"] & \sim RA(M\#_0) \arrow[u, phantom, "\rotatebox{90}{$\leq$}"] \arrow[r, phantom, "\leq"] & \sim RA(M\#) \arrow[u, phantom, "\rotatebox{90}{$\leq$}"]
\end{tikzcd}$$

$\boxed{\sim RA(S\#)}$

EXPTIME-hard

$\sim RA(S\#)$ is EXPTIME-hard

Reminder $\sim RA(S\#)$:

- register automaton (without fresh transitions)
- all registers must contain different values (or be empty)
- register content can be deleted

General idea:

- Reduction from EXPTIME-hard problem ALBA-MEM to $\sim RA(S\#)$
- Simulate computation of the ALBA twice, where one simulation behaves different to the other only if the input is rejected

$\sim$ $RA(S\#)$ is EXPTIME-hard

Given an instance of an ALBA-MEM problem $\langle M, w\rangle$

- wlog: $\Gamma \setminus \{\triangleright, \triangleleft\} = \{0,1\}$, $|\delta(q,a)| \leq 2$ for all (q,a)

Build $RA(S\#)$ $\mathcal{A}_M^w = (Q', \Sigma, \delta)$ with $2w + 1$ registers as follows:

- $Q' = (Q \times [0, |w| + 1], \{L, R\}) \uplus Q_{\text{aux}}$
- $|\Sigma| = 1$ (all tags are equal $\rightarrow$ will be left out from here on)
- δ, so that $(q_0, 0, L) \sim (q_0, 0, R) \Leftrightarrow M$ accepts w

$\sim RA(S\#)$ is EXPTIME-hard

Transitions

- δ will assure that the bisimulation game for $(q_0, 0, L) \sim (q_0, 0, R)$ simulates the computation of M with input w.
- If M is in a configuration, (q, k, t) then the bisimulation is in a configuration $((q, k, L), \rho), ((q, k, R), \rho)$ where:
  - $\rho(2k) = \# \Leftrightarrow \rho(2k+1) \neq \# \Leftrightarrow$ cell k on t contains 0
  - $\rho(2k) \neq \# \Leftrightarrow \rho(2k+1) = \# \Leftrightarrow$ cell k on t contains 1

  $\Rightarrow$ add transitions depending on $\delta(q, a)$

$\sim RA(S\#)$ is EXPTIME-hard

Case 1: $\delta(q, a) = \emptyset$: No need to add any transitions

Case 2: $\delta(q, a) = \{(q', b, z)\}$: Add the following transitions:

$$(q, k, C) \xrightarrow{\ell_a} \cdot \xrightarrow{\ell'_b} (q', k + z, C)$$

where:

$$\begin{array}{ll} \ell_0 = (\{2k + 1\}, 0, \{2k + 1\}) & \ell'_0 = (\emptyset, \{2k + 1\}, \emptyset) \\ \ell_1 = (\{2k\}, 0, \{2k\}) & \ell'_1 = (\emptyset, \{2k\}, \emptyset) \end{array}$$

$\sim RA(S\#)$ is EXPTIME-hard

Case 3: $\delta(q,a) = \{(q_1,b_1,z_1),(q_2,b_2,z_2)\}$:

If q is a universal state add transitions

$$(q_1, k+z_1, C) \xleftarrow{\ell'_{b_1}} \cdot \xleftarrow{A} \cdot \xleftarrow{\ell_a} (q,k,C) \xrightarrow{\ell_a} \cdot \xrightarrow{B} \cdot \xrightarrow{\ell'_{b_2}} (q_2, k+z_2, C)$$

$\sim RA(S\#)$ is EXPTIME-hard

Case 3: $\delta(q,a) = \{(q_1, b_1, z_1), (q_2, b_2, z_2)\}$:

If q is a universal state add transitions

$$(q_1, k+z_1, C) \xleftarrow{\ell'_{b_1}} \cdot \xleftarrow{A} \cdot \xleftarrow{\ell_a} (q, k, C) \xrightarrow{\ell_a} \cdot \xrightarrow{B} \cdot \xrightarrow{\ell'_{b_2}} (q_2, k+z_2, C)$$

If q is an existential state, use defender forcing gadget

$\sim$ $RA(S\#)$ is EXPTIME-hard

Defender forcing gadget

$$\begin{tikzcd}
 & & q_L \arrow[dl, "\ell"'] \arrow[dr, "\ell"] \arrow[drrr, "\ell"] & & q_R \arrow[dl, "\ell"'] \arrow[dr, "\ell"] & & \\
 & \cdot \arrow[dl, "\ell_1"'] \arrow[dr, "\ell_2"] & & \cdot \arrow[dl, "\ell_2"'] \arrow[dr, "\ell_1"] & & \cdot \arrow[dlllll, "\ell_1"] \arrow[dr, "\ell_2"] & \\
p_L & & p'_L & & p_R & & p'_R
\end{tikzcd}$$

$\sim RA(S\#)$ is EXPTIME-hard

End of the simulation:

- No transitions from states (q_{acc}, k, C) $\Rightarrow$ (q_{acc}, k, L) and (q_{acc}, k, R) are trivially bisimilar
- For q_{rej} add transition $(q_{rej}, k, L) \xrightarrow{(\{1\},0,\emptyset)} (q_{rej}, k, L)$ only in the Left copy $\Rightarrow$ (q_{rej}, k, L) and (q_{rej}, k, R) are not bisimilar

EXPTIME-Complete Bisimilarity Problems

$$\begin{array}{ccccccc}
 & & & & & & \in \text{EXPTIME} \checkmark \\
\sim FRA(S\#) & \leq & \sim FRA(MF) & \leq & \sim FRA(M\#_0) & \leq & \boxed{\sim FRA(M\#)} \\
\rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} \\
\boxed{\sim RA(S\#)} & \leq & \sim RA(MF) & \leq & \sim RA(M\#_0) & \leq & \sim RA(M\#) \\
\text{EXPTIME-hard} \checkmark & & & & & &
\end{array}$$

Bisimilarity Problems in (Fresh-) Register Automata

$$\begin{array}{ccccccccccc}
\text{NP-solvable} & & \text{PSPACE-Complete} & & \multicolumn{7}{c}{\text{EXPTIME-Complete } \checkmark} \\
\boxed{\begin{array}{c} \sim FRA(SF) \\ \rotatebox{90}{$\leq$} \\ \sim RA(SF) \end{array}} & \begin{array}{c} \leq \\ \\ \leq \end{array} & \boxed{\begin{array}{c} \sim FRA(S\#_0) \\ \rotatebox{90}{$\leq$} \\ \sim RA(S\#_0) \end{array}} & \begin{array}{c} \leq \\ \\ \leq \end{array} & \begin{array}{c} \sim FRA(S\#) \\ \rotatebox{90}{$\leq$} \\ \sim RA(S\#) \end{array} & \begin{array}{c} \leq \\ \\ \leq \end{array} & \begin{array}{c} \sim FRA(MF) \\ \rotatebox{90}{$\leq$} \\ \sim RA(MF) \end{array} & \begin{array}{c} \leq \\ \\ \leq \end{array} & \begin{array}{c} \sim FRA(M\#_0) \\ \rotatebox{90}{$\leq$} \\ \sim RA(M\#_0) \end{array} & \begin{array}{c} \leq \\ \\ \leq \end{array} & \begin{array}{c} \sim FRA(M\#) \\ \rotatebox{90}{$\leq$} \\ \sim RA(M\#) \end{array}
\end{array}$$

$\sim RA(S\#_0)$ is PSPACE-complete

$\sim RA(S\#_0)$ is PSPACE-complete

Reminder $\sim RA(S\#_0)$:

- register automaton (without fresh transitions)
- all registers must contain different values (or be empty)
- register content cannot be deleted (only updated)

Notation:

There are only two types of transitions:

- $p \xrightarrow{(t,i)} q$ (short for $p \xrightarrow{(t,\{i\},0,\emptyset)} q$)
- $p \xrightarrow{(t,i^\bullet)} q$ (short for $p \xrightarrow{(t,\emptyset,i,\emptyset)} q$)

$\sim RA(S\#_0) \in PSPACE$

General idea:

- Define symbolic bisimulation which tracks the set of active registers in two configurations and relates those registers from the configurations that hold the same value
- Define indexed bisimilarity, which represents how many rounds are needed at most for an attacker to show that two configurations are not bisimilar
- Show that there is a polynomial upper bound for indexed bisimilarity $\Rightarrow$ suffices to play the bisimulation game for polynomially many rounds

$\sim RA(S\#_0) \in PSPACE$

Symbolic Bisimulations

$\mathcal{U} = \{(q_1, S_1, \sigma, q_2, S_2) | q_1, q_2 \in Q, S_1, S_2 \subseteq [1, r], \sigma \in \mathcal{IS}_r \subseteq S_1 \times S_2\}$
$R \subseteq \mathcal{U}$ is a symbolic simulation if all members satisfy the (SyS) conditions:

- for all $q_1 \xrightarrow{(t,i)} q_1'$
  - if $i \in dom(\sigma)$ then there is some $q_2 \xrightarrow{(t,\sigma(i))} q_2'$ with $(q_1', S_1) R_\sigma (q_2', S_2)$
  - if $i \in S_1 \setminus dom(\sigma)$ then there is some $q_2 \xrightarrow{(t,j^\bullet)} q_2'$ with $(q_1', S_1) R_{\sigma[i \to j]} (q_2', S_2[j])$
- for all $q_1 \xrightarrow{(t,i^\bullet)} q_1'$
  - there is some $q_2 \xrightarrow{(t,j^\bullet)} q_2'$ with $(q_1', S_1[i]) R_{\sigma[i \to j]} (q_2', S_2[j])$
  - for all $j \in S_2 \setminus rng(\sigma)$, there is some $q_2 \xrightarrow{(t,j)} q_2'$ with $(q_1', S_1[i]) R_{\sigma[i \to j]} (q_2', S_2)$.

R is a symbolic bisimulation if R and R^{-1} are symbolic simulations.

$\sim RA(S\#_0) \in PSPACE$

Notation:

- $\overset{s}{\sim}$ denotes the union of all symbolic Bisimulations.
- We write $(q_1, \rho_1) \overset{s}{\sim} (q_2, \rho_2)$ if $(q_1, dom(\rho_1), \rho_1; \rho_2^{-1}, q_2, dom(\rho_2)) \in \overset{s}{\sim}$

Lemma

$(q_1, \rho_1) \sim (q_2, \rho_2) \Leftrightarrow (q_1, \rho_1) \overset{s}{\sim} (q_2, \rho_2)$

$\sim RA(S\#_0) \in PSPACE$

Indexed Bisimilarity

- $\overset{0}{\sim} = \mathcal{U}$
- $\overset{i+1}{\sim} = \{(q_1, S_1, \sigma, q_2, S_2) |$
  $(q_1, S_1, \sigma, q_2, S_2)$ satisfies the (SyS) conditions in $\overset{i}{\sim}$
  $(q_2, S_2, \sigma^{-1}, q_1, S_1)$ satisfies the (SyS) conditions in $\overset{i}{\sim}\}$

$\Rightarrow$ If $c \in \overset{i}{\sim}$ but $c \notin \overset{i+1}{\sim}$, then there is a strategy for an Attacker to win the bisimulation game in $i + 1$ rounds

Goal: show that there is an upper bound B so that $\overset{B}{\sim} = \overset{s}{\sim}$

$\sim RA(S\#_0) \in PSPACE$

Lemma

- $\overset{s}{\sim} = CL(\overset{s}{\sim})$
- for all i: $\overset{i}{\sim} = CL(\overset{i}{\sim})$

$CL(R)$ refers to the smallest relation R' containing R which is closed under the rules:

$$\frac{}{(q, S, \mathrm{id}_S, q, S) \in R'} \text{ (ID)}$$

$$\frac{(q_1, S_1, \sigma_1, q_2, S_2) \in R' \qquad (q_2, S_2, \sigma_2, q_3, S_3) \in R'}{(q_1, S_1, \sigma_1; \sigma_2, q_3, S_3) \in R'} \text{ (TR)}$$

$$\frac{(q_1, S_1, \sigma, q_2, S_2) \in R'}{(q_2, S_2, \sigma^{-1}, q_1, S_1) \in R'} \text{ (SYM)}$$

$$\frac{(q_1, S_1, \sigma, q_2, S_2) \in R' \qquad \sigma \leq_{S_1, S_2} \sigma'}{(q_1, S_1, \sigma', q_2, S_2) \in R'} \text{ (EXT)}$$

$\sim RA(S\#_0) \in PSPACE$

Characteristic Sets and Groups

Let $p \in Q, S \subseteq [1, r], R \subseteq \mathcal{U}$ closed.

- $\mathcal{X}_S^p(R) = \bigcap\{X \subseteq S | (p, S) R_{id_X} (p, S)\}$
- $\mathcal{G}_S^p(R) = \{\sigma \subseteq X_S^p(R) \times X_S^p(R) | (p, S) R_\sigma (p, S)\}$

$\sim RA(S\#_0) \in PSPACE$

Lemma

For fixed $S_1, S_2 \subseteq [1, r]$, the sub-chain
$\{\overset{i}{\sim} \mid (\overset{i+1}{\sim} \cap \mathcal{U}_{S_1,S_2}) \subsetneq (\overset{i}{\sim} \cap \mathcal{U}_{S_1,S_2}\}$ has size at most
$2r|Q| + 2r|Q|(2r-2) + |Q|^2$

Proof (sketch):

If $(\overset{i+1}{\sim} \cap \mathcal{U}_{S_1,S_2}) \subsetneq (\overset{i}{\sim} \cap \mathcal{U}_{S_1,S_2}\}$ then this is because one of three reasons:

- $\mathcal{X}^q_{S_k}(\overset{i+1}{\sim}) \subsetneq \mathcal{X}^q_{S_k}(\overset{i}{\sim})$
- $\mathcal{G}^q_{S_k}(\overset{i+1}{\sim})$ is a strict subgroup of $\mathcal{G}^q_{S_k}(\overset{i}{\sim})$
- there are configurations $(q_1, S_1), (q_2, S_2)$ that are unseparated in $(\overset{i}{\sim})$ and become separated in $(\overset{i+1}{\sim})$

$\sim RA(S\#_0) \in PSPACE$

Lemma

Let $B = (2r+1)\cdot(2r|Q| + 2r|Q|(2r-2) + |Q|^2)$. Then

$$\overset{B}{\sim} \cap \mathcal{U}_{S_1,S_2} = \overset{s}{\sim} \cap \mathcal{U}_{S_1,S_2}$$

for any S_1, S_2

Therefore it suffices to play the bisimulation game for polynomially many steps. This can be done via an alternating Turing machine, and the PSPACE bound follows from APTIME = PSPACE.

$\sim RA(S\#_0)$ is PSPACE-hard

- Reduce from PSPACE-hard problem checking the validity of totally quantified boolean formulas in prenex conjunctive normal form
- Construct write once ALBA that guesses a truth assignment in alternating moves according to the quantifiers and verifies its correctness
- Use previous construction to build a $RA(S\#_0)$
- Because the ALBA is write once, the resulting RA obeys $\#_0$

Bisimilarity Problems in (Fresh-) Register Automata

NP-solvable

PSPACE-Complete (✓)

EXPTIME-Complete ✓

$$
\begin{array}{ccccccccccccc}
\sim FRA(SF) & \leq & \sim FRA(S\#_0) & \leq & \sim FRA(S\#) & \leq & \sim FRA(MF) & \leq & \sim FRA(M\#_0) & \leq & \sim FRA(M\#) \\
\rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} \\
\sim RA(SF) & \leq & \sim RA(S\#_0) & \leq & \sim RA(S\#) & \leq & \sim RA(MF) & \leq & \sim RA(M\#_0) & \leq & \sim RA(M\#)
\end{array}
$$

$$\sim RA(SF) \in \mathrm{NP}$$

$\sim RA(SF) \in \mathrm{NP}$

Reminder $\sim RA(SF)$:

- register automaton (without fresh transitions)
- all registers must contain different values
- all registers must always be filled

Notation:

We are only going to consider $\overset{s}{\sim}$ and $S_1 = S_2 = [1, r]$ for all elements $(q_1, S_1, \sigma, q_2, S_2) \in \mathcal{U}$. Therefore, we write:

- $\mathcal{X}^p$ for $\mathcal{X}^p_{[1,r]}(\overset{s}{\sim})$
- $\mathcal{G}^p$ for $\mathcal{G}^p_{[1,r]}(\overset{s}{\sim})$

$\sim RA(SF) \in \mathrm{NP}$

Lemma

$\overset{s}{\sim}$ can be generated by polynomially many elements

Idea:

- Partition Q into equivalence classes $(p \sim q \Leftrightarrow \exists \sigma.(p, [1, r], \sigma, q, [1, r]) \in \overset{s}{\sim}$

$\sim RA(SF) \in \mathrm{NP}$

Lemma

$\overset{s}{\sim}$ can be generated by polynomially many elements

Idea:

- Partition Q into equivalence classes $(p \sim q \Leftrightarrow \exists \sigma.(p, [1, r], \sigma, q, [1, r]) \in \overset{s}{\sim}$
- Pick random member p_i for each equivalence class P_i

$\sim RA(SF) \in \mathrm{NP}$

Lemma

$\overset{s}{\sim}$ can be generated by polynomially many elements

Idea:

- Partition Q into equivalence classes $(p \sim q \Leftrightarrow \exists \sigma.(p, [1, r], \sigma, q, [1, r]) \in \overset{s}{\sim}$
- Pick random member p_i for each equivalence class P_i
- Select sets G^{p_i} of linearly many generators for $\mathcal{G}^{p_i}$

$\sim RA(SF) \in$ NP

Lemma

$\overset{s}{\sim}$ can be generated by polynomially many elements

Idea:

- Partition Q into equivalence classes $(p \sim q \Leftrightarrow \exists \sigma.(p, [1, r], \sigma, q, [1, r]) \in \overset{s}{\sim}$
- Pick random member p_i for each equivalence class P_i
- Select sets G^{p_i} of linearly many generators for $\mathcal{G}^{p_i}$
- For $q \in P_i \setminus \{p_i\}$ there exists σ so that $(p_i, [1, r], \sigma, q, [1, r]) \in \overset{s}{\sim}$. Set $\mathrm{ray}_q^{p_i} = \sigma \cap (X^{p_i} \times [1, r])$

$\sim RA(SF) \in$ NP

Lemma

$\overset{s}{\sim}$ can be generated by polynomially many elements

Idea:

- Partition Q into equivalence classes $(p \sim q \Leftrightarrow \exists \sigma.(p, [1, r], \sigma, q, [1, r]) \in \overset{s}{\sim}$
- Pick random member p_i for each equivalence class P_i
- Select sets G^{p_i} of linearly many generators for $\mathcal{G}^{p_i}$
- For $q \in P_i \setminus \{p_i\}$ there exists σ so that $(p_i, [1, r], \sigma, q, [1, r]) \in \overset{s}{\sim}$. Set $\text{ray}_q^{p_i} = \sigma \cap (X^{p_i} \times [1, r])$
- Then $\overset{s}{\sim} = CL(\{(p_i, [1, r], \sigma, p_i, [1, r]) | \sigma \in \mathcal{G}^{p_i}\} \cup \{(p_i, [1, r], \text{ray}_q^{p_i}, q, [1, r]) | q \in P_i\})$

$\sim RA(SF) \in$ NP

Theorem

$\sim RA(SF) \in$ NP

Proof

- Guess a generating system in polynomial time
- Test if the so generated system is a bisimultaion $\rightarrow$ test (SyS) conditions for elements in the generating system
- If it is a bisimulation, then test if the given problem instance is part of the generated system.

Bisimilarity Problems in (Fresh-) Register Automata

NP-solvable (✓) PSPACE-Complete (✓) EXPTIME-Complete ✓

$$\begin{array}{ccccccccccc}
\sim FRA(SF) & \leq & \sim FRA(S\#_0) & \leq & \sim FRA(S\#) & \leq & \sim FRA(MF) & \leq & \sim FRA(M\#_0) & \leq & \sim FRA(M\#) \\
\rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} & & \rotatebox{90}{$\leq$} \\
\sim RA(SF) & \leq & \sim RA(S\#_0) & \leq & \sim RA(S\#) & \leq & \sim RA(MF) & \leq & \sim RA(M\#_0) & \leq & \sim RA(M\#)
\end{array}$$

Future Work

- Can the NP bound for $\sim (F)RA(SF)$ be improved?

References I

A. S. Murawski, S. J. Ramsay, and N. Tzevelekos.
Bisimilarity in fresh-register automata.
Log. Methods Comput. Sci., 21(1), 2025.