# Synchronizing Data Words for Register Automata

Parvaneh Babari [1], Karin Quaas [1], Mahsa Shirmohammadi [2]

[1] Universität Leipzig, [2] University of Oxford

MFCS 2016

# Data Words and Data Languages

$\Sigma$...a finite alphabet
$D$...an infinite data domain, e.g., $\mathbb{N}$, $\mathbb{R}_{\geq 0}$, `ASCII strings`

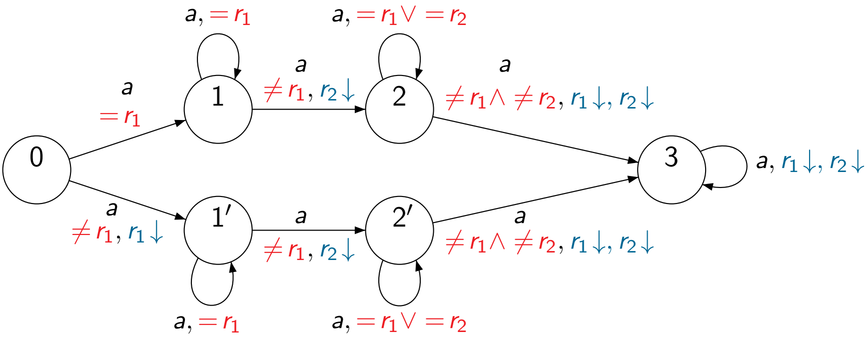
A **data word** is a finite sequence

$$(a_1, d_1)(a_2, d_2) \ldots (a_n, d_n)$$

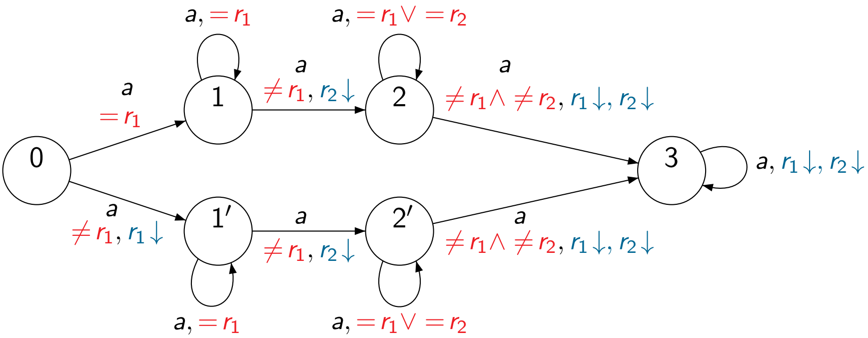
over $\Sigma \times D$.

A **data language** is a subset of $(\Sigma \times D)^*$.

# Register Automata (RA)



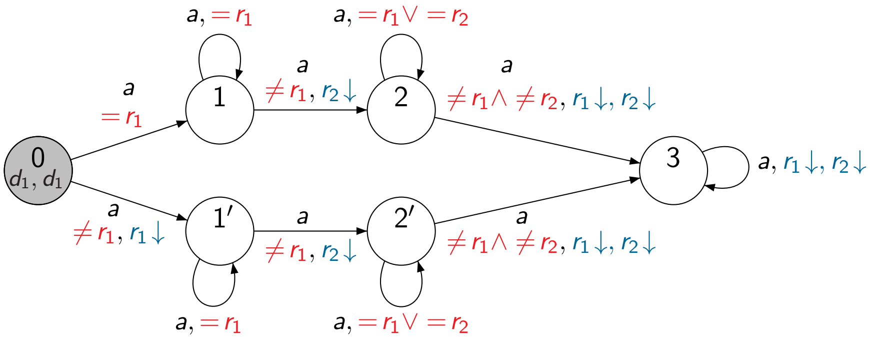
# Register Automata (RA)




$$w = (a, d_1)(a, d_2)(a, d_1)(a, d_3)$$

$$(0, d_1, d_1) \xrightarrow{(a,d_1)} (1, d_1, d_1) \xrightarrow{(a,d_2)} (2, d_1, d_2) \xrightarrow{(a,d_1)} (2, d_1, d_2) \xrightarrow{(a,d_3)} (3, d_3, d_3)$$

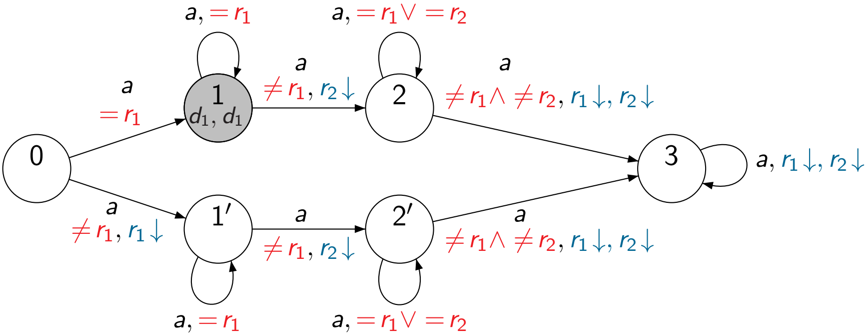
# Register Automata (RA)




$$w = (a, d_1)(a, d_2)(a, d_1)(a, d_3)$$

$$(0, d_1, d_1) \xrightarrow{(a,d_1)} (1, d_1, d_1) \xrightarrow{(a,d_2)} (2, d_1, d_2) \xrightarrow{(a,d_1)} (2, d_1, d_2) \xrightarrow{(a,d_3)} (3, d_3, d_3)$$

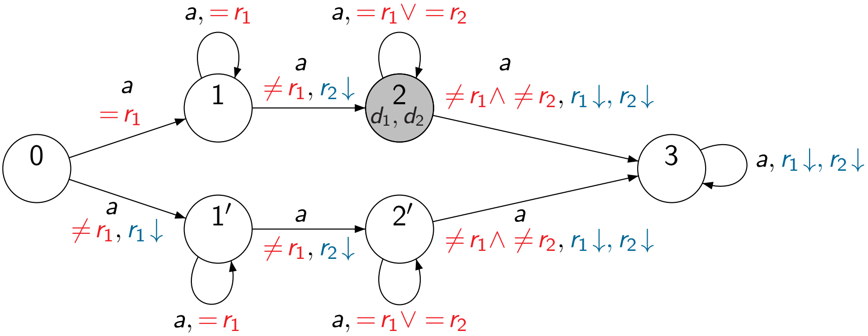
# Register Automata (RA)




$$w = (a, d_1)(a, d_2)(a, d_1)(a, d_3)$$

$$(0, d_1, d_1) \xrightarrow{(a,d_1)} \boxed{(1, d_1, d_1)} \xrightarrow{(a,d_2)} (2, d_1, d_2) \xrightarrow{(a,d_1)} (2, d_1, d_2) \xrightarrow{(a,d_3)} (3, d_3, d_3)$$

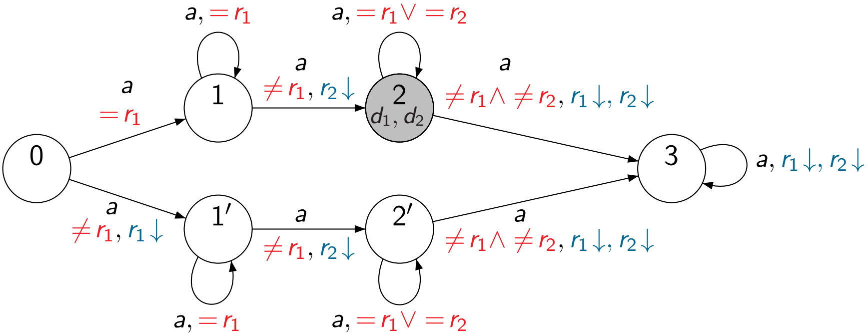
# Register Automata (RA)




$$w = (a, d_1)(a, d_2)(a, d_1)(a, d_3)$$

$$(0, d_1, d_1) \xrightarrow{(a,d_1)} (1, d_1, d_1) \xrightarrow{(a,d_2)} \boxed{(2, d_1, d_2)} \xrightarrow{(a,d_1)} (2, d_1, d_2) \xrightarrow{(a,d_3)} (3, d_3, d_3)$$

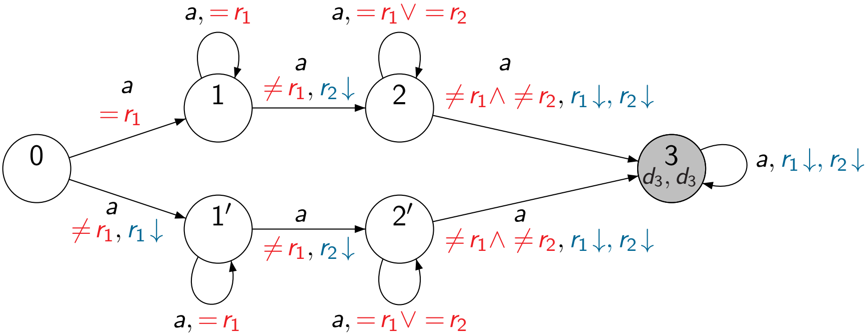
# Register Automata (RA)




$$w = (a, d_1)(a, d_2)(a, d_1)(a, d_3)$$

$$(0, d_1, d_1) \xrightarrow{(a,d_1)} (1, d_1, d_1) \xrightarrow{(a,d_2)} (2, d_1, d_2) \xrightarrow{(a,d_1)} (2, d_1, d_2) \xrightarrow{(a,d_3)} (3, d_3, d_3)$$

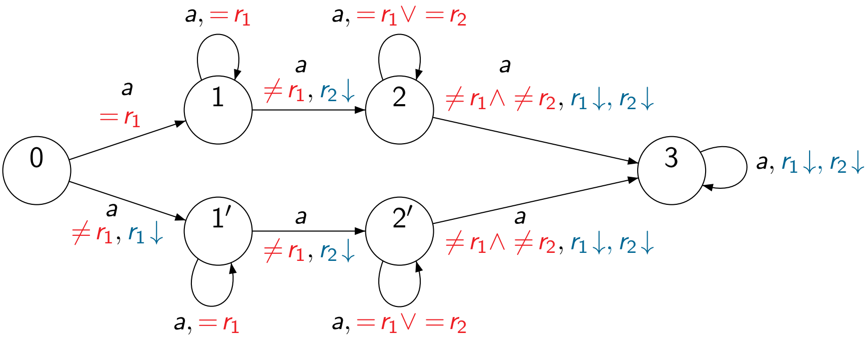
# Register Automata (RA)




$$w = (a, d_1)(a, d_2)(a, d_1)(a, d_3)$$

$$(0, d_1, d_1) \xrightarrow{(a,d_1)} (1, d_1, d_1) \xrightarrow{(a,d_2)} (2, d_1, d_2) \xrightarrow{(a,d_1)} (2, d_1, d_2) \xrightarrow{(a,d_3)} \boxed{(3, d_3, d_3)}$$

# Register Automata (RA)




$$w = (a, d_1)(a, d_2)(a, d_1)(a, d_3)$$

$$(0, d_1, d_1) \xrightarrow{(a,d_1)} (1, d_1, d_1) \xrightarrow{(a,d_2)} (2, d_1, d_2) \xrightarrow{(a,d_1)} (2, d_1, d_2) \xrightarrow{(a,d_3)} (3, d_3, d_3)$$

# Facts about Register Automata

### The Non-Emptiness Problem

| | |
|---|---|
| Instance: | A register automaton $\mathcal{A}$. |
| Question: | Does $\mathcal{A}$ accept a data word? |

# Facts about Register Automata

### The Non-Emptiness Problem

Instance: A register automaton $\mathcal{A}$.

Question: Does $\mathcal{A}$ accept a data word?

- PSPACE-complete (Demri & Lazic, 2008)

# Facts about Register Automata

### The Non-Emptiness Problem

Instance: A register automaton $\mathcal{A}$.
Question: Does $\mathcal{A}$ accept a data word?

- PSPACE-complete (Demri & Lazic, 2008)

### The Non-Universality Problem

Instance: A register automaton $\mathcal{A}$.
Question: Does there exist some data word that $\mathcal{A}$ does not accept?

# Facts about Register Automata

### The Non-Emptiness Problem

Instance: A register automaton $\mathcal{A}$.
Question: Does $\mathcal{A}$ accept a data word?

- PSPACE-complete (Demri & Lazic, 2008)

### The Non-Universality Problem

Instance: A register automaton $\mathcal{A}$.
Question: Does there exist some data word that $\mathcal{A}$ does not accept?

- PSPACE-complete for deterministic RA (DRA) (Demri & Lazic, 2008)

# Facts about Register Automata

### The Non-Emptiness Problem

Instance: A register automaton $\mathcal{A}$.
Question: Does $\mathcal{A}$ accept a data word?

- PSPACE-complete (Demri & Lazic, 2008)

### The Non-Universality Problem

Instance: A register automaton $\mathcal{A}$.
Question: Does there exist some data word that $\mathcal{A}$ does not accept?

- PSPACE-complete for deterministic RA (DRA) (Demri & Lazic, 2008)
- undecidable for non-deterministic RA (NRA) (Neven, Schwentick & Vianu, 2004)

# Facts about Register Automata

### The Non-Emptiness Problem

Instance: A register automaton $\mathcal{A}$.
Question: Does $\mathcal{A}$ accept a data word?

- PSPACE-complete (Demri & Lazic, 2008)

### The Non-Universality Problem

Instance: A register automaton $\mathcal{A}$.
Question: Does there exist some data word that $\mathcal{A}$ does not accept?

- PSPACE-complete for deterministic RA (DRA) (Demri & Lazic, 2008)
- undecidable for non-deterministic RA (NRA) (Neven, Schwentick & Vianu, 2004)
- decidable for NRA with one register (1-NRA), Ackermann-complete (Demri & Lazic, 2008)

# Synchronizing Data Words for Register Automata

# Synchronizing Data Words for Register Automata

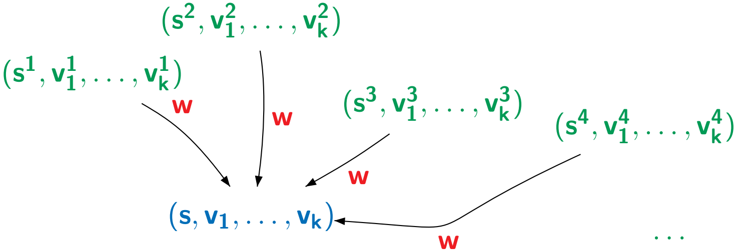
Let $\mathcal{A}$ be a RA with $k$ registers.

A data word $\mathbf{w}$ is called **synchronizing for** $\mathcal{A}$ if there exists a configuration $(\mathbf{s}, \mathbf{v_1}, \ldots, \mathbf{v_k})$ such that $\mathcal{A}$ is in $(\mathbf{s}, \mathbf{v_1}, \ldots, \mathbf{v_k})$ after processing $\mathbf{w}$,
no matter from which configuration $\mathbf{c}$ $\mathcal{A}$ starts from.

# Synchronizing Data Words for Register Automata

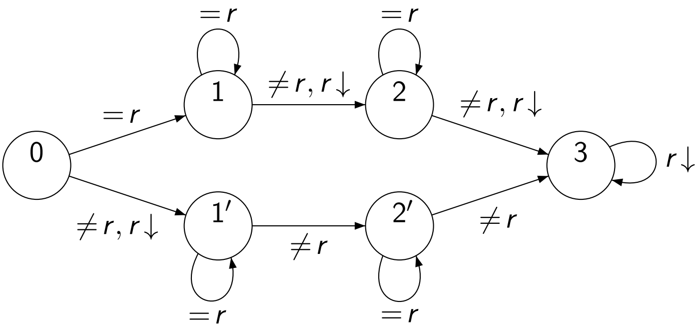
Let $\mathcal{A}$ be a RA with $k$ registers.

A data word $\mathbf{w}$ is called **synchronizing for** $\mathcal{A}$ if
there exists a configuration $(\mathbf{s}, \mathbf{v_1}, \ldots, \mathbf{v_k})$ such that $\mathcal{A}$ is in $(\mathbf{s}, \mathbf{v_1}, \ldots, \mathbf{v_k})$ after processing $\mathbf{w}$,
no matter from which configuration $\mathbf{c}$ $\mathcal{A}$ starts from.



# Example Synchronizing Data Words

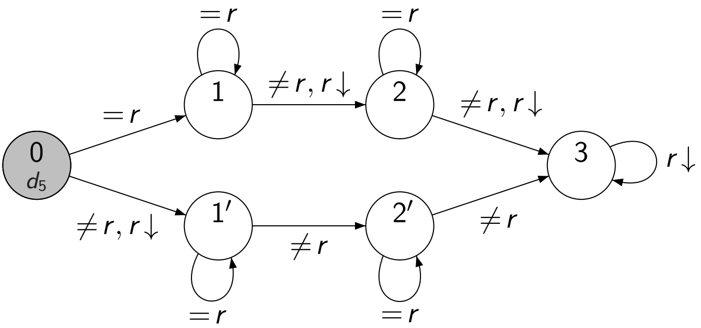
The data word $w = (a, d_1)(a, d_2)(a, d_3)(a, d_4)$ is synchronizing for




because this RA is in $(3, d_4)$ after processing $w$, no matter from which configuration $c$ $\mathcal{A}$ starts from.

# Example Synchronizing Data Words

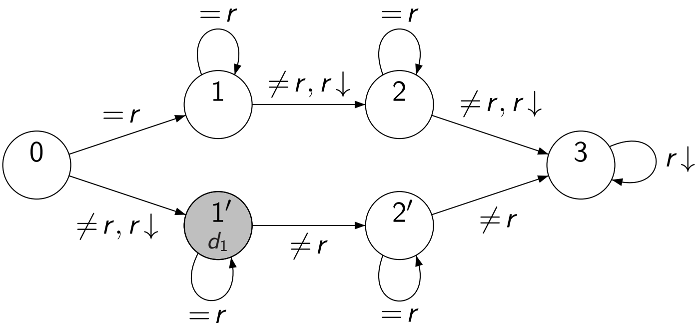
The data word $w = (a, d_1)(a, d_2)(a, d_3)(a, d_4)$ is synchronizing for




because this RA is in $(3, d_4)$ after processing $w$, no matter from which configuration $c$ $\mathcal{A}$ starts from.

# Example Synchronizing Data Words

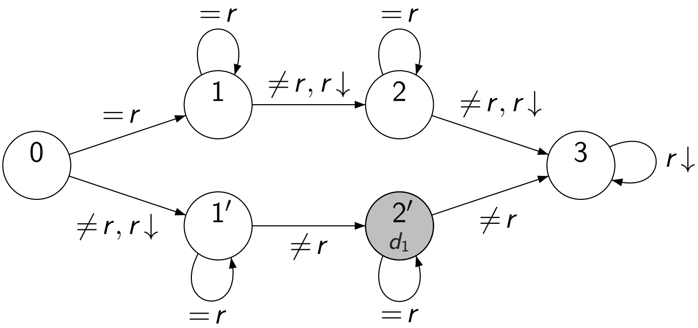
The data word $w = (a, d_1)(a, d_2)(a, d_3)(a, d_4)$ is synchronizing for




because this RA is in $(3, d_4)$ after processing $w$, no matter from which configuration $c$ $\mathcal{A}$ starts from.

# Example Synchronizing Data Words

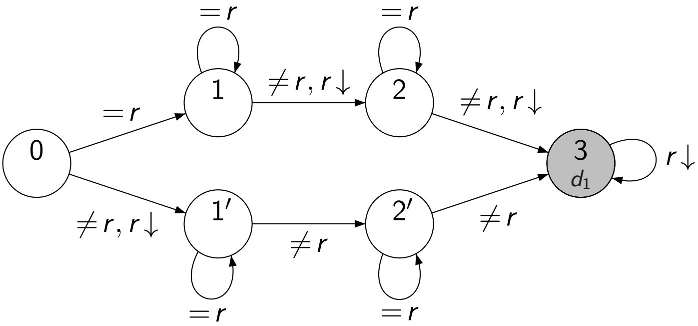
The data word $w = (a, d_1)(a, d_2)(a, d_3)(a, d_4)$ is synchronizing for




because this RA is in $(3, d_4)$ after processing $w$, no matter from which configuration $c$ $\mathcal{A}$ starts from.

# Example Synchronizing Data Words

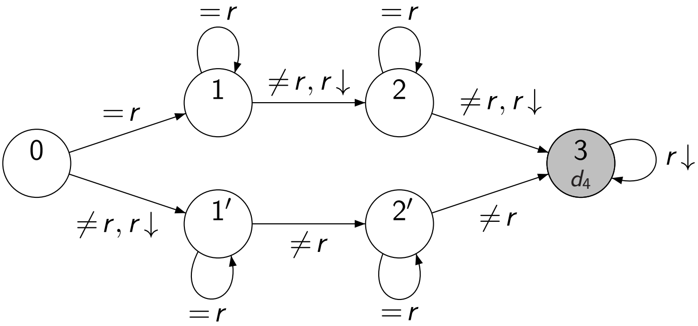
The data word $w = (a, d_1)(a, d_2)(a, d_3)(a, d_4)$ is synchronizing for




because this RA is in $(3, d_4)$ after processing $w$, no matter from which configuration $c$ $\mathcal{A}$ starts from.

# Example Synchronizing Data Words

The data word $w = (a, d_1)(a, d_2)(a, d_3)(a, d_4)$ is synchronizing for




because this RA is in $(3, d_4)$ after processing $w$, no matter from which configuration $c$ $\mathcal{A}$ starts from.

# How to prove the Synchronizing Property

$$
\begin{array}{c}
\{0, 1, 2, 1', 2', 3\} \times D \\
\Big\downarrow (a, d_1) \\
\{(1, d_1), (1', d_1), (2, d_1), (3, d_1), (2', d_1)\} \cup (\{2', 3\} \times D \backslash \{d_1\}) \\
\Big\downarrow (a, d_2) \\
\{(2, d_2), (2', d_1), (3, d_2), (2', d_2), (3, d_1)\} \cup (\{3\} \times D \backslash \{d_1, d_2\}) \\
\Big\downarrow (a, d_3) \\
\{(3, d_1), (3, d_2), (3, d_3)\} \\
\Big\downarrow (a, d_4) \\
\{(3, d_4)\}
\end{array}
$$

# How to prove the Synchronizing Property

$$
\begin{tikzcd}
\{0,1,2,1',2',3\} \times D \arrow[d, "{(a, d_1)}"] \\
\{(1,d_1),(1',d_1),(2,d_1),(3,d_1),(2',d_1)\} \cup (\{2',3\} \times D \backslash \{d_1\}) \arrow[d, "{(a, d_2)}"] \\
\{(2,d_2),(2',d_1),(3,d_2),(2',d_2),(3,d_1)\} \cup (\{3\} \times D \backslash \{d_1,d_2\}) \arrow[d, "{(a, d_3)}"] \\
\{(3,d_1),(3,d_2),(3,d_3)\} \arrow[d, "{(a, d_4)}"] \\
\{(3,d_4)\}
\end{tikzcd}
$$

# Synchronizing Problem for Register Automata

**The Synchronizing Problem**

Instance: A register automaton $\mathcal{A}$.

Question: Does there exist a synchronizing data word for $\mathcal{A}$ ?

# Some Notes on the Synchronizing Problem

- Synchronizing words for finite automata (FA) (Černy 1964, Pin 1978)

# Some Notes on the Synchronizing Problem

- Synchronizing words for finite automata (FA) (Černy 1964, Pin 1978)
- Synchronizing problem for deterministic FA is in PTIME

# Some Notes on the Synchronizing Problem

- Synchronizing words for finite automata (FA) (Černy 1964, Pin 1978)
- Synchronizing problem for deterministic FA is in PTIME
- Černy Conjecture: The length of a shortest synchronizing word for deterministic FA with $n$ states is at most $(n-1)^2$.

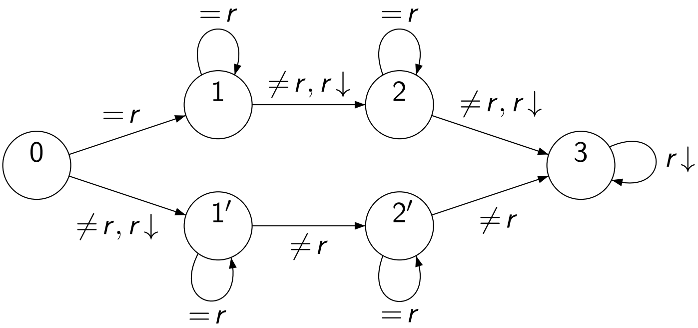
# Some Notes on the Synchronizing Problem

- Synchronizing words for finite automata (FA) (Černy 1964, Pin 1978)
- Synchronizing problem for deterministic FA is in PTIME
- Černy Conjecture: The length of a shortest synchronizing word for deterministic FA with $n$ states is at most $(n-1)^2$.
- Overview article by Volkov, 2008.

# Synchronizing Data Words for DRA

# Synchronizing Data Words for DRA

The data word $w = (a, d_1)(a, d_2)(a, d_3)(a, d_4)$ is synchronizing for




because $\mathcal{A}$ is in $(3, d_4)$ after processing $w$, no matter from which configuration $c$ $\mathcal{A}$ starts from.

# Synchronizing Data Words for DRA

$$\{0, 1, 2, 1', 2', 3\} \times D$$

$$\big\downarrow (a, d_1)$$

$$\{(1, d_1), (1', d_1), (2, d_1), (3, d_1), (2', d_1)\} \cup (\{2', 3\} \times D \backslash \{d_1\})$$

$$\big\downarrow (a, d_2)$$

$$\{(2, d_2), (2', d_1), (3, d_2), (2', d_2), (3, d_1)\} \cup (\{3\} \times D \backslash \{d_1, d_2\})$$

$$\big\downarrow (a, d_3)$$

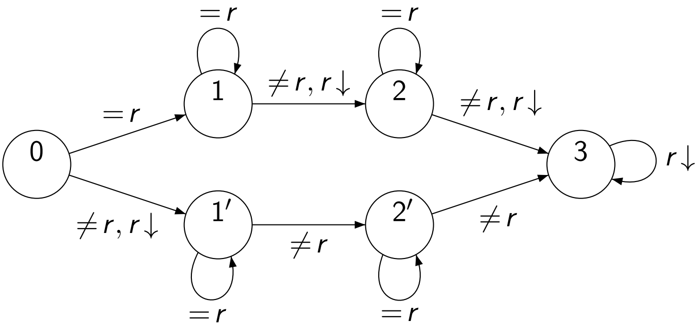
$$\{(3, d_1), (3, d_2), (3, d_3)\}$$

$$\big\downarrow (a, d_4)$$

$$\{(3, d_4)\}$$

# Synchronizing Data Words for DRA

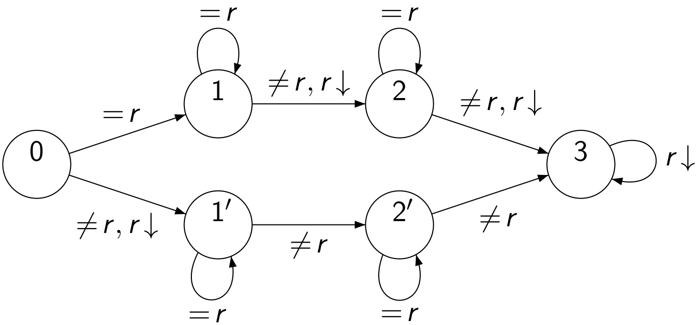
The data word $w = (a, d_1)(a, d_2)(a, d_3)(a, d_4)$ is synchronizing for




because $\mathcal{A}$ is in $(3, d_4)$ after processing $w$, no matter from which configuration $c$ $\mathcal{A}$ starts from.

# Synchronizing Data Words for DRA

The data word $w = (a, d_1)(a, d_2)(a, d_3)(a, d_4)$ is synchronizing for




because $\mathcal{A}$ is in $(3, d_4)$ after processing $w$, no matter from which configuration $c$ $\mathcal{A}$ starts from.

The data word $(a, d_1)(a, d_1)(a, d_1)(a, d_2)(a, d_1)(a, d_2)(a, d_1)$ is also synchronizing for $\mathcal{A}$.

$$
\begin{array}{c}
\{0,1,2,1',2',3\} \times D \\
\Big\downarrow (a, d_1) \\
\{(1,d_1),(1',d_1),(2,d_1),(3,d_1),(2',d_1)\} \cup (\{2',3\} \times D \backslash \{d_1\}) \\
\Big\downarrow (a, d_1) \\
\{(1,d_1),(1',d_1),(2,d_1),(3,d_1),(2',d_1)\} \cup (\{3\} \times D \backslash \{d_1\}) \\
\Big\downarrow (a, d_1) \\
\{(1,d_1),(1',d_1),(2,d_1),(3,d_1),(2',d_1)\} \\
\Big\downarrow (a, d_2) \\
\{(2,d_2),(2',d_1),(3,d_2),(3,d_1)\} \\
\Big\downarrow (a, d_1) \\
\{(3,d_1),(2',d_1)\} \\
\Big\downarrow (a, d_2) \\
\{(3,d_2),(3,d_1)\} \\
\Big\downarrow (a, d_1) \\
\{(3,d_1)\}
\end{array}
$$

# Synchronizing Data Words for DRA

**Lemma**

*If for a $k$-DRA $\mathcal{A}$ there exists some synchronizing data word, then there exists also some synchronizing data word for $\mathcal{A}$ with at most $2k + 1$ distinct data values.*

# Synchronizing Data Words for DRA

**Lemma**

*If for a k-DRA $\mathcal{A}$ there exists some synchronizing data word, then there exists also some synchronizing data word for $\mathcal{A}$ with at most $2k + 1$ distinct data values.*

**Theorem**

*The synchronizing problem for DRA is* PSPACE*-complete.*

# Synchronizing Data Words for NRA

**Theorem**

*The synchronizing problem for NRA is undecidable.*

**Proof Sketch.**

By reduction of the non-universality problem for NRA (preserving the number of registers). $\square$

# Synchronizing Data Words for NRA

**Theorem**

*The synchronizing problem for NRA is undecidable.*

**Proof Sketch.**

By reduction of the non-universality problem for NRA (preserving the number of registers). □

**Theorem**

*For* 1-*NRA the synchronizing problem is* Ackermann-*complete.*

**Proof Sketch.**

By reduction to the non-universality problem for 1-NRA. □

### The Length-Bounded Synchronizing Problem

Instance: An NRA $\mathcal{A}$, a natural number $N \in \mathbb{N}$.

Question: Does there exist a synchronizing data word for $\mathcal{A}$ with length at most $N$ ?

# Length-Bounded Synchronizing Data Words for NRA

**The Length-Bounded Synchronizing Problem**

Instance: An NRA $\mathcal{A}$, a natural number $N \in \mathbb{N}$.

Question: Does there exist a synchronizing data word for $\mathcal{A}$ with length at most $N$ ?

**Theorem**

*The length-bounded synchronizing problem for NRA is* NEXPTIME*-complete.*

**Proof Sketch.**

# Length-Bounded Synchronizing Data Words for NRA

**The Length-Bounded Synchronizing Problem**

Instance: An NRA $\mathcal{A}$, a natural number $N \in \mathbb{N}$.

Question: Does there exist a synchronizing data word for $\mathcal{A}$ with length at most $N$ ?

**Theorem**

*The length-bounded synchronizing problem for NRA is* NEXPTIME*-complete.*

**Proof Sketch.**

NEXPTIME-hardness by reduction from the bounded universality problem for **regular-like expressions** (built from atomic expressions and $\cdot$, $+$, and $^2$). $\square$

# Conclusion and Open Problems

The synchronizing problem is

- PSPACE-complete for DRA
- undecidable for NRA
- Ackermann-complete for 1-NRA

# Conclusion and Open Problems

The synchronizing problem is

- PSPACE-complete for DRA
- undecidable for NRA
- Ackermann-complete for 1-NRA

The length-bounded synchronizing problem for NRA is NEXPTIME-complete.

# Conclusion and Open Problems

The synchronizing problem is

- PSPACE-complete for DRA
- undecidable for NRA
- Ackermann-complete for 1-NRA

The length-bounded synchronizing problem for NRA is NEXPTIME-complete.

Open: Precise complexity for the length-bounded synchronizing problem for DRA.