Shrinking One-Way Cellular Automata

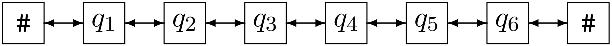
Martin Kutrib Andreas Malcher Matthias Wendlandt

Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
email: {kutrib,malcher,matthias.wendlandt}@informatik.uni-giessen.de

AUTOMATA 2015, Turku, Finland

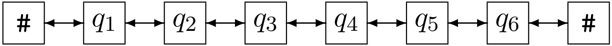
Cellular Automata and Iterative Arrays

A two-way cellular automaton (CA):

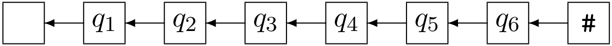


Cellular Automata and Iterative Arrays

A two-way cellular automaton (CA):

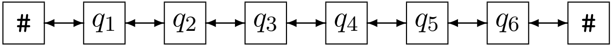


A one-way cellular automaton (OCA):

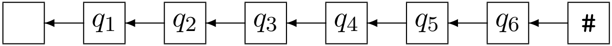


Cellular Automata and Iterative Arrays

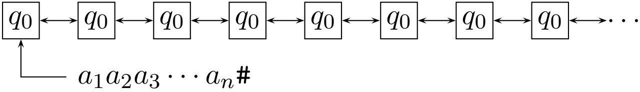
A two-way cellular automaton (CA):



A one-way cellular automaton (OCA):

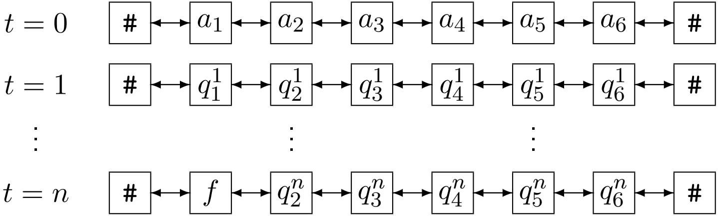


An iterative array (IA) is a cellular automaton with sequential input mode.



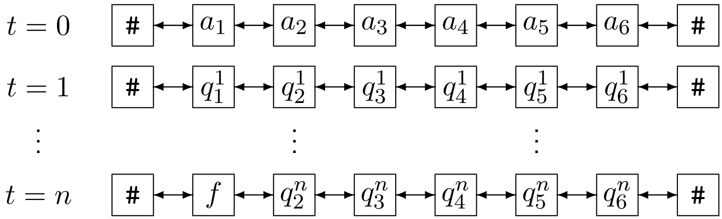
Recognizing Formal Languages With Cellular Automata

Input $u = a_1 a_2 \cdots a_6 \in A^+$



Recognizing Formal Languages With Cellular Automata

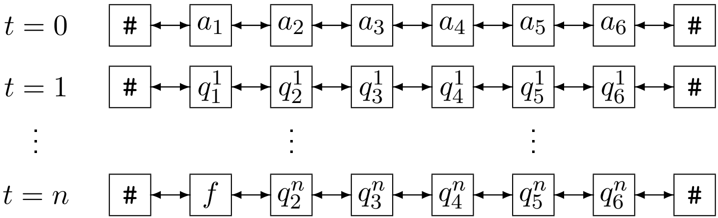
Input $u = a_1 a_2 \cdots a_6 \in A^+$



➜ $u \in A^+$ is accepted, if there exists a time step at which the first cell enters an accepting state.

Recognizing Formal Languages With Cellular Automata

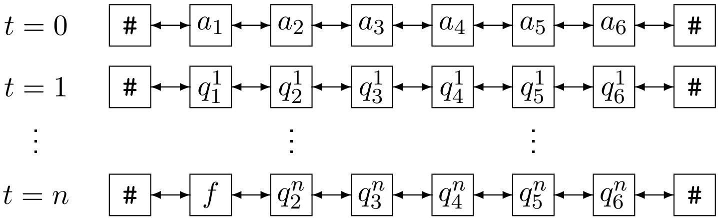
Input $u = a_1 a_2 \cdots a_6 \in A^+$



- $u \in A^+$ is accepted, if there exists a time step at which the first cell enters an accepting state.
- $L(M) = \{ u \in A^+ \mid u \text{ is accepted by } M \}$

Recognizing Formal Languages With Cellular Automata

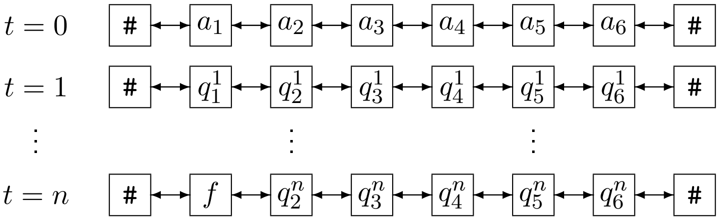
Input $u = a_1 a_2 \cdots a_6 \in A^+$



- $u \in A^+$ is accepted, if there exists a time step at which the first cell enters an accepting state.
- $L(M) = \{ u \in A^+ \mid u \text{ is accepted by } M \}$
- M has time complexity $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, if all $u \in L(M)$ are accepted within $t(|u|)$ time steps.

Recognizing Formal Languages With Cellular Automata

Input $u = a_1 a_2 \cdots a_6 \in A^+$



- ➜ $u \in A^+$ is accepted, if there exists a time step at which the first cell enters an accepting state.
- ➜ $L(M) = \{ u \in A^+ \mid u \text{ is accepted by } M \}$
- ➜ M has time complexity $t : \mathbb{N} \to \mathbb{N}$, $t(n) \geq n$, if all $u \in L(M)$ are accepted within $t(|u|)$ time steps.
- ➜ $\mathscr{L}_t(\mathrm{CA}) = \{ L \mid L \text{ is accepted with time complexity } t \}$

Important Language Classes

- realtime-CA languages $\mathscr{L}_{rt}$(CA) ($t(|u|) = |u|$)
- lineartime-CA languages $\mathscr{L}_{lt}$(CA) ($t(|u|) = m \cdot |u|$, $m \in \mathbb{Q}$, $m \geq 1$)

The language classes for one-way cellular automata $\mathscr{L}_{rt}$(OCA), $\mathscr{L}_{lt}$(OCA) are defined analogously.

Important Language Classes

- realtime-CA languages $\mathscr{L}_{rt}(\mathsf{CA})$ $(t(|u|) = |u|)$
- lineartime-CA languages $\mathscr{L}_{lt}(\mathsf{CA})$ $(t(|u|) = m \cdot |u|$, $m \in \mathbb{Q}$, $m \geq 1)$

The language classes for one-way cellular automata $\mathscr{L}_{rt}(\mathsf{OCA})$, $\mathscr{L}_{lt}(\mathsf{OCA})$ are defined analogously.

For iterative arrays

- realtime-IA languages $\mathscr{L}_{rt}(\mathsf{IA})$ $(t(|u|) = |u| + 1)$
- lineartime-IA languages $\mathscr{L}_{lt}(\mathsf{IA})$ $(t(|u|) = m \cdot |u|$, $m \in \mathbb{Q}$, $m \geq 1)$

Computational Capacity

$$
\begin{array}{ccccccccc}
 & & \mathrm{DCS} & = & \mathscr{L}(\mathrm{CA}) & = & \mathscr{L}(\mathrm{IA}) & & \\
 & & & & \rotatebox[origin=c]{90}{$\subseteq$} & & & & \\
 & & \mathrm{CF} & \subset & \mathscr{L}(\mathrm{OCA}) & & \rotatebox[origin=c]{90}{$\subseteq$} & & \\
 & & & & \rotatebox[origin=c]{90}{$\subseteq$} & & & & \\
 & & & & \mathscr{L}_{lt}(\mathrm{CA}) & = & \mathscr{L}_{lt}(\mathrm{IA}) & & \\
 & & & & \rotatebox[origin=c]{90}{$\subseteq$} & & \rotatebox[origin=c]{90}{$\subset$} & & \\
 & & \mathrm{DCF} & \subset & \mathscr{L}_{rt}(\mathrm{CA}) & \supset & \mathscr{L}_{rt}(\mathrm{IA}) & \supset & \mathrm{DCF}_{\lambda} \\
 & & & & \parallel & & & & \\
 & & & & \mathscr{L}_{lt}(\mathrm{OCA})^{R} & & & & \\
 & & & & \rotatebox[origin=c]{90}{$\subset$} & & & & \\
\mathrm{REG} & \subset & \mathrm{LCF} & \subset & \mathscr{L}_{rt}(\mathrm{OCA}) & & & &
\end{array}
$$

The language classes $\mathscr{L}_{rt}$(OCA) and $\mathscr{L}_{rt}$(IA) are incomparable.
Both CF and $\mathscr{L}_{rt}$(OCA) and CF and $\mathscr{L}_{rt}$(IA) are incomparable.

Shrinking Cellular Automata – Definition

A shrinking one-way cellular automaton (SOCA) is a system $\langle S, F, A, \#, \delta \rangle$, where

- ➜ S is the finite set of cell states,
- ➜ $F \subseteq S$ is the set of accepting states,
- ➜ $A \subseteq S$ is the finite set of input symbols,
- ➜ $\# \notin S$ is the permanent boundary symbol,
- ➜ $\delta : S \times S_{\#} \to S \cup \{\text{dissolve}\}$ is the local transition function.

Shrinking Cellular Automata – Definition

A shrinking one-way cellular automaton (SOCA) is a system $\langle S, F, A, \#, \delta \rangle$, where

- ➜ S is the finite set of cell states,
- ➜ $F \subseteq S$ is the set of accepting states,
- ➜ $A \subseteq S$ is the finite set of input symbols,
- ➜ $\# \notin S$ is the permanent boundary symbol,
- ➜ $\delta : S \times S_{\#} \to S \cup \{\text{dissolve}\}$ is the local transition function.

Shrinking Cellular Automata – Definition

A shrinking one-way cellular automaton (SOCA) is a system $\langle S, F, A, \#, \delta\rangle$, where

- ➜ S is the finite set of cell states,
- ➜ $F \subseteq S$ is the set of accepting states,
- ➜ $A \subseteq S$ is the finite set of input symbols,
- ➜ $\# \notin S$ is the permanent boundary symbol,
- ➜ $\delta : S \times S_{\#} \to S \cup \{\text{dissolve}\}$ is the local transition function.

Shrinking Cellular Automata – Definition

A shrinking one-way cellular automaton (SOCA) is a system $\langle S, F, A, \#, \delta \rangle$, where

- ➜ S is the finite set of cell states,
- ➜ $F \subseteq S$ is the set of accepting states,
- ➜ $A \subseteq S$ is the finite set of input symbols,
- ➜ $\# \notin S$ is the permanent boundary symbol,
- ➜ $\delta : S \times S_{\#} \to S \cup \{\text{dissolve}\}$ is the local transition function.

Shrinking Cellular Automata – Definition

A shrinking one-way cellular automaton (SOCA) is a system $\langle S, F, A, \#, \delta \rangle$, where

- ➜ S is the finite set of cell states,
- ➜ $F \subseteq S$ is the set of accepting states,
- ➜ $A \subseteq S$ is the finite set of input symbols,
- ➜ $\# \notin S$ is the permanent boundary symbol,
- ➜ $\delta : S \times S_{\#} \to S \cup \{\text{dissolve}\}$ is the local transition function.

Shrinking Cellular Automata – Definition

A shrinking one-way cellular automaton (SOCA) is a system $\langle S, F, A, \#, \delta \rangle$, where

- ➜ S is the finite set of cell states,
- ➜ $F \subseteq S$ is the set of accepting states,
- ➜ $A \subseteq S$ is the finite set of input symbols,
- ➜ $\# \notin S$ is the permanent boundary symbol,
- ➜ $\delta : S \times S_{\#} \to S \cup \{\text{dissolve}\}$ is the local transition function.

We define realtime-SOCA languages $\mathscr{L}_{rt}(\text{SOCA})$ with $t(|u|) = |u|$ as usual.

Example

$L = \{\, \$w \mid w \in \{a,b\}^* \text{ and } |w|_a = |w|_b \,\} \in \mathscr{L}_{rt}(\mathsf{SOCA})$.

Example

$L = \{\, \$w \mid w \in \{a,b\}^* \text{ and } |w|_a = |w|_b \,\} \in \mathscr{L}_{rt}(\mathsf{SOCA})$.

$	a	b	b	a	b	b	a	a	#

Example

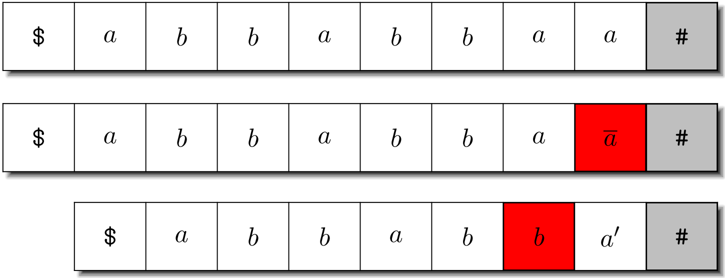
$L = \{ \$w \mid w \in \{a,b\}^* \text{ and } |w|_a = |w|_b \} \in \mathscr{L}_{rt}(\text{SOCA})$.

\$	a	b	b	a	b	b	a	a	#

\$	a	b	b	a	b	b	a	$\overline{a}$	#

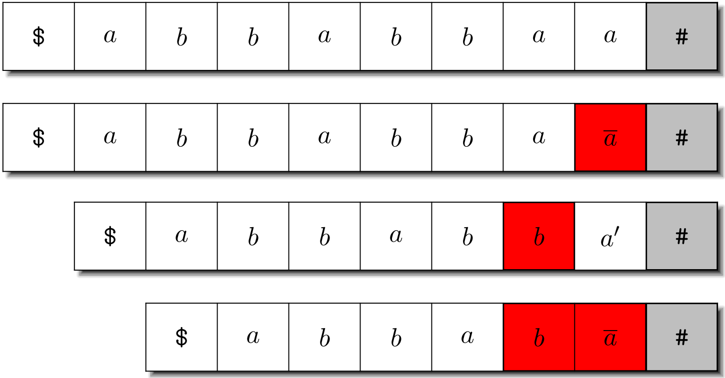
Example

$L = \{\, \$w \mid w \in \{a,b\}^* \text{ and } |w|_a = |w|_b \,\} \in \mathscr{L}_{rt}(\mathsf{SOCA})$.



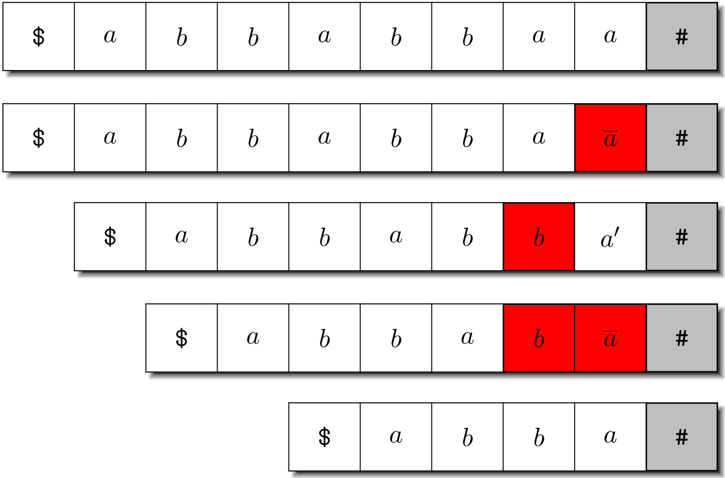
Example

$L = \{ \$w \mid w \in \{a,b\}^* \text{ and } |w|_a = |w|_b \} \in \mathscr{L}_{rt}(\mathsf{SOCA})$.



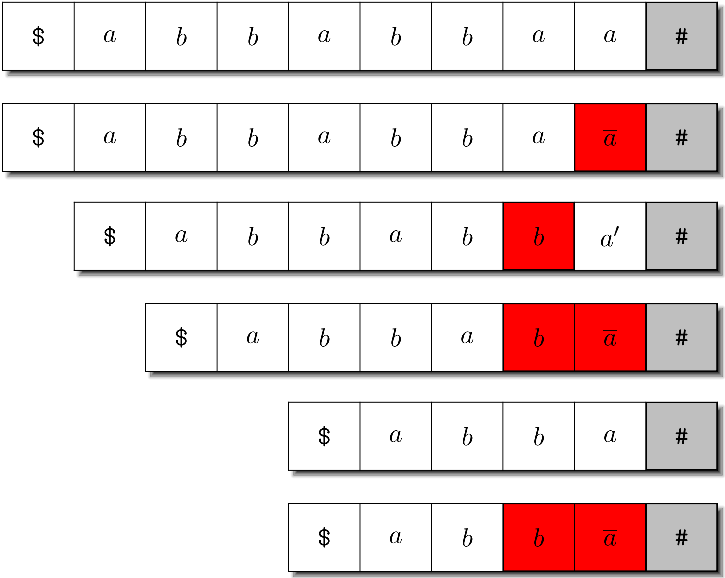
Example

$L = \{\,\$w \mid w \in \{a,b\}^* \text{ and } |w|_a = |w|_b\,\} \in \mathscr{L}_{rt}(\text{SOCA})$.



Example

$L = \{\, \$w \mid w \in \{a,b\}^* \text{ and } |w|_a = |w|_b \,\} \in \mathscr{L}_{rt}(\mathsf{SOCA})$.



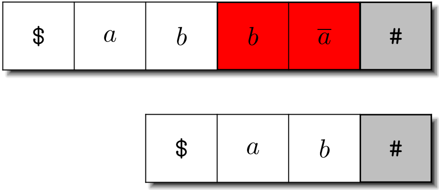
Example (2)

$L = \{\, \$w \mid w \in \{a,b\}^* \text{ and } |w|_a = |w|_b \,\} \in \mathscr{L}_{rt}(\mathsf{SOCA})$.

\$	a	b	b	$\overline{a}$	#

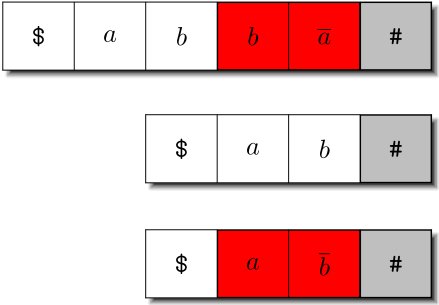
Example (2)

$L = \{\, \$w \mid w \in \{a,b\}^* \text{ and } |w|_a = |w|_b \,\} \in \mathscr{L}_{rt}(\mathsf{SOCA})$.



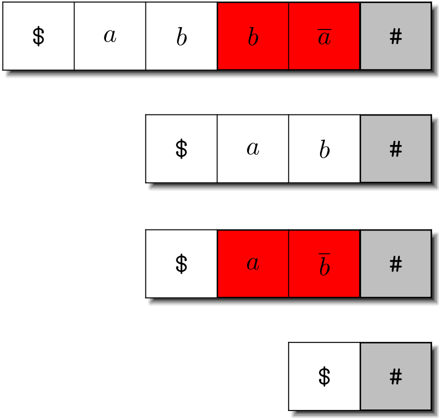
Example (2)

$L = \{ \$w \mid w \in \{a, b\}^* \text{ and } |w|_a = |w|_b \} \in \mathscr{L}_{rt}(\mathsf{SOCA})$.



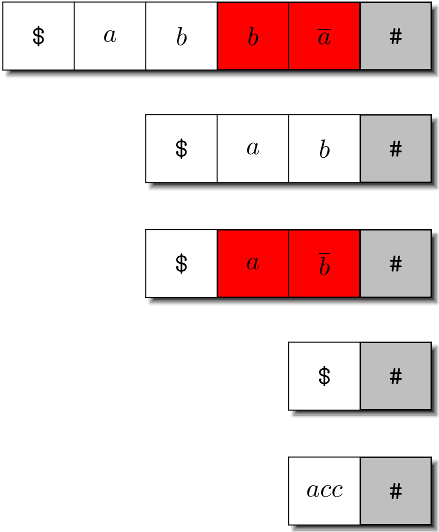
Example (2)

$L = \{ \$w \mid w \in \{a,b\}^* \text{ and } |w|_a = |w|_b \} \in \mathscr{L}_{rt}(\text{SOCA})$.



Example (2)

$L = \{\, \$w \mid w \in \{a,b\}^* \text{ and } |w|_a = |w|_b \,\} \in \mathscr{L}_{rt}(\text{SOCA})$.



Computational Capacity

A valuable tool concerns embeddings. Let $L \subseteq A^*$, \$ $\notin A$, and $\mathrm{emb} : A^* \to A_{\$}^*$, where $\mathrm{emb}(a) = a\$$ for $a \in A$.

Computational Capacity

A valuable tool concerns embeddings. Let $L \subseteq A^*$, $\$ \notin A$, and $\mathrm{emb} : A^* \to A_{\$}^*$, where $\mathrm{emb}(a) = a\$$ for $a \in A$.

Lemma

Let $r : \mathbb{N} \to \mathbb{N}$ be an increasing function so that $r(O(n)) \leq O(r(n))$. A language L belongs to the family $\mathscr{L}_{n+r(n)}(\mathrm{OCA})$ if and only if $\mathrm{emb}(L)$ belongs to $\mathscr{L}_{n+r(n)}(\mathrm{OCA})$.

Computational Capacity

A valuable tool concerns embeddings. Let $L \subseteq A^*$, $\$ \notin A$, and $\mathrm{emb} : A^* \to A_{\$}^*$, where $\mathrm{emb}(a) = a\$$ for $a \in A$.

Lemma

Let $r : \mathbb{N} \to \mathbb{N}$ be an increasing function so that $r(O(n)) \leq O(r(n))$. A language L belongs to the family $\mathscr{L}_{n+r(n)}(\mathrm{OCA})$ if and only if $\mathrm{emb}(L)$ belongs to $\mathscr{L}_{n+r(n)}(\mathrm{OCA})$.

- Let M be an OCA accepting $\mathrm{emb}(L)$. In an OCA accepting L each cell simulates two adjacent cells of M and the OCA is finally sped up suitably.

Computational Capacity

A valuable tool concerns embeddings. Let $L \subseteq A^*$, $\$ \notin A$, and $\mathrm{emb} : A^* \to A_\$^*$, where $\mathrm{emb}(a) = a\$$ for $a \in A$.

Lemma

Let $r : \mathbb{N} \to \mathbb{N}$ be an increasing function so that $r(O(n)) \leq O(r(n))$. A language L belongs to the family $\mathscr{L}_{n+r(n)}(\mathrm{OCA})$ if and only if $\mathrm{emb}(L)$ belongs to $\mathscr{L}_{n+r(n)}(\mathrm{OCA})$.

- Let M be an OCA accepting $\mathrm{emb}(L)$. In an OCA accepting L each cell simulates two adjacent cells of M and the OCA is finally sped up suitably.

- Let M be an OCA accepting L. An OCA accepting $\mathrm{emb}(L)$ is first sped up suitably. Then, two adjacent cells simulate in two time steps one transition of M.

Computational Capacity (2)

Lemma

Let L be a language from $\mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$. Then $\mathrm{emb}(L)$ belongs to $\mathscr{L}_{rt}(\mathrm{SOCA})$.

Computational Capacity (2)

Lemma

Let L be a language from $\mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$. Then $\mathrm{emb}(L)$ belongs to $\mathscr{L}_{rt}(\mathrm{SOCA})$.

- ➜ $L \in \mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$ can be sped up to be accepted in time $2n - 2$.

Computational Capacity (2)

Lemma

Let L be a language from $\mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$. Then $\mathrm{emb}(L)$ belongs to $\mathscr{L}_{rt}(\mathrm{SOCA})$.

- ➜ $L \in \mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$ can be sped up to be accepted in time $2n - 2$.
- ➜ An OCA for $\mathrm{emb}(L)$ checks in the first time step the correct format of the input.

Computational Capacity (2)

Lemma

Let L be a language from $\mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$. Then $\mathrm{emb}(L)$ belongs to $\mathscr{L}_{rt}(\mathrm{SOCA})$.

- ➜ $L \in \mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$ can be sped up to be accepted in time $2n - 2$.
- ➜ An OCA for $\mathrm{emb}(L)$ checks in the first time step the correct format of the input.
- ➜ In the second time step, all \$-cells are dissolved.

Computational Capacity (2)

Lemma

Let L be a language from $\mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$. Then $\mathrm{emb}(L)$ belongs to $\mathscr{L}_{rt}(\mathrm{SOCA})$.

- $L \in \mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$ can be sped up to be accepted in time $2n - 2$.
- An OCA for $\mathrm{emb}(L)$ checks in the first time step the correct format of the input.
- In the second time step, all $-cells are dissolved.
- In the remaining time, L is simulated.

Computational Capacity (2)

Lemma

Let L be a language from $\mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$. Then $\mathrm{emb}(L)$ belongs to $\mathscr{L}_{rt}(\mathrm{SOCA})$.

- $L \in \mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$ can be sped up to be accepted in time $2n - 2$.
- An OCA for $\mathrm{emb}(L)$ checks in the first time step the correct format of the input.
- In the second time step, all $-cells are dissolved.
- In the remaining time, L is simulated.

Theorem

Let L be a language from $\mathscr{L}_{lt}(\mathrm{OCA}) \setminus \mathscr{L}_{rt}(\mathrm{OCA})$. Then $\mathrm{emb}(L)$ belongs to $\mathscr{L}_{rt}(\mathrm{SOCA})$ but does not belong to $\mathscr{L}_{rt}(\mathrm{OCA})$. In particular, the family $\mathscr{L}_{rt}(\mathrm{OCA})$ is properly included in $\mathscr{L}_{rt}(\mathrm{SOCA})$.

Real-Time SOCA and Iterative Arrays

Theorem

Let L belong to $\mathscr{L}_{rt}(\mathrm{IA})$. Then $\{\, w \mid w^R \in L$ and $|w|$ is even $\}$ is accepted by a real-time SOCA.

Real-Time SOCA and Iterative Arrays

Theorem

Let L belong to $\mathscr{L}_{rt}(\mathrm{IA})$. Then $\{\, w \mid w^R \in L$ and $|w|$ is even $\}$ is accepted by a real-time SOCA.

- ➜ Embedding of the two-way computation into a one-way device.

Real-Time SOCA and Iterative Arrays

Theorem

Let L belong to $\mathscr{L}_{rt}(\mathrm{IA})$. Then $\{\, w \mid w^R \in L$ and $|w|$ is even $\}$ is accepted by a real-time SOCA.

- ➜ Embedding of the two-way computation into a one-way device.
- ➜ Speed-up of the computation.

Real-Time SOCA and Iterative Arrays

Theorem

Let L belong to $\mathscr{L}_{rt}(\mathrm{IA})$. Then $\{\, w \mid w^R \in L$ and $|w|$ is even $\}$ is accepted by a real-time SOCA.

- ➜ Embedding of the two-way computation into a one-way device.
- ➜ Speed-up of the computation.
- ➜ Dissolving of cells to gain additional time.

Real-Time SOCA and Iterative Arrays

Theorem

Let L belong to $\mathscr{L}_{rt}(\mathrm{IA})$. Then $\{\, w \mid w^R \in L \text{ and } |w| \text{ is even} \,\}$ is accepted by a real-time SOCA.

- Embedding of the two-way computation into a one-way device.
- Speed-up of the computation.
- Dissolving of cells to gain additional time.

Theorem

Let L belong to $\mathscr{L}_{rt}(\mathrm{IA})$. Then $\{\, w \mid w^R \in L \text{ and } |w| \text{ is odd} \,\}$ is accepted by a real-time SOCA.

Applications

- ➜ $\{\, a^{2^n} \mid n \geq 1 \,\} \in \mathscr{L}_{rt}(\mathrm{SOCA})$

Applications

- $\{ a^{2^n} \mid n \geq 1 \} \in \mathscr{L}_{rt}(\mathrm{SOCA})$
- $\{ a^p \mid p > 2$ and p is prime $\} \in \mathscr{L}_{rt}(\mathrm{SOCA})$

Applications

- $\{ a^{2^n} \mid n \geq 1 \} \in \mathscr{L}_{rt}(\mathrm{SOCA})$
- $\{ a^p \mid p > 2$ and p is prime $\} \in \mathscr{L}_{rt}(\mathrm{SOCA})$
- $\{ w \mid w \in \{a, b\}^*$ and $|w|_a = |w|_b \}$

Applications

- $\{ a^{2^n} \mid n \geq 1 \} \in \mathscr{L}_{rt}(\mathrm{SOCA})$
- $\{ a^p \mid p > 2$ and p is prime $\} \in \mathscr{L}_{rt}(\mathrm{SOCA})$
- $\{ w \mid w \in \{a, b\}^*$ and $|w|_a = |w|_b \}$

Theorem

Let $L \subseteq A^*$ be a language from $\mathscr{L}_{rt}(\mathrm{IA})$ and $\$ \notin A$ be a letter. Then $\{ \$w \mid w^R \in L \}$ is accepted by a real-time SOCA.

Applications

- $\{\, a^{2^n} \mid n \geq 1 \,\} \in \mathscr{L}_{rt}(\mathrm{SOCA})$
- $\{\, a^p \mid p > 2$ and p is prime $\} \in \mathscr{L}_{rt}(\mathrm{SOCA})$
- $\{\, w \mid w \in \{a, b\}^*$ and $|w|_a = |w|_b \,\}$

Theorem

Let $L \subseteq A^*$ be a language from $\mathscr{L}_{rt}(\mathrm{IA})$ and $\$ \notin A$ be a letter. Then $\{\, \$w \mid w^R \in L \,\}$ is accepted by a real-time SOCA.

- Remember $\mathrm{DCF}_\lambda \subset \mathscr{L}_{rt}(\mathrm{IA})$.

Dissolving versus Time

- Let M be an SOCA and $f : \mathbb{N} \to \mathbb{N}$ be an increasing function.

Dissolving versus Time

- Let M be an SOCA and $f : \mathbb{N} \to \mathbb{N}$ be an increasing function.
- If all $w \in L(M)$ are accepted with computations where the number of dissolved cells is bounded by $f(|w|)$, then M is said to be dissolving bounded by f.

Dissolving versus Time

- Let M be an SOCA and $f : \mathbb{N} \to \mathbb{N}$ be an increasing function.
- If all $w \in L(M)$ are accepted with computations where the number of dissolved cells is bounded by $f(|w|)$, then M is said to be dissolving bounded by f.
- The corresponding class of SOCA is denoted by f-SOCA.

Dissolving versus Time

- Let M be an SOCA and $f : \mathbb{N} \to \mathbb{N}$ be an increasing function.
- If all $w \in L(M)$ are accepted with computations where the number of dissolved cells is bounded by $f(|w|)$, then M is said to be dissolving bounded by f.
- The corresponding class of SOCA is denoted by f-SOCA.

Theorem

Let M be an f-SOCA working in real time. Then an equivalent conventional OCA M' working in time $n + f(n)$ can effectively be constructed.

Dissolving versus Time

- Let M be an SOCA and $f : \mathbb{N} \to \mathbb{N}$ be an increasing function.
- If all $w \in L(M)$ are accepted with computations where the number of dissolved cells is bounded by $f(|w|)$, then M is said to be dissolving bounded by f.
- The corresponding class of SOCA is denoted by f-SOCA.

Theorem

Let M be an f-SOCA working in real time. Then an equivalent conventional OCA M' working in time $n + f(n)$ can effectively be constructed.

- Freeze cells instead of dissolving them.
- Slow down the computation.

Dissolving versus Time (2)

- What about the other way around?

Dissolving versus Time (2)

- What about the other way around?
- Is there a general approach to trade time for dissolving of cells?

Dissolving versus Time (2)

- What about the other way around?
- Is there a general approach to trade time for dissolving of cells?
- There exists an infinite time hierarchy $\mathscr{L}_{n+r_2(n)}(\mathsf{OCA}) \subset \mathscr{L}_{n+r_1(n)}(\mathsf{OCA})$ in between real-time and linear-time.

Dissolving versus Time (2)

- What about the other way around?
- Is there a general approach to trade time for dissolving of cells?
- There exists an infinite time hierarchy $\mathscr{L}_{n+r_2(n)}$(OCA) $\subset \mathscr{L}_{n+r_1(n)}$(OCA) in between real-time and linear-time.

Theorem

Let $r_1, r_2 : \mathbb{N} \to \mathbb{N}$ be two increasing functions. If r_1^{-1} is OCA-constructible, $r_2(O(n)) \leq O(r_2(n))$, and $r_2 \cdot \log(r_2) \in o(r_1)$, then

$$\mathscr{L}_{rt}(r_2\text{-SOCA}) \subset \mathscr{L}_{rt}(r_1\text{-SOCA}).$$

Dissolving versus Time (3)

Steps of the proof:

- Let L_{r_1} be the witness language of the time hierarchy.

Dissolving versus Time (3)

Steps of the proof:

- Let L_{r_1} be the witness language of the time hierarchy.
- Define $L_{\$,r_1}$ such that exactly one \$ is inserted after each of the rightmost $r_1(|w|)$ symbols of $w \in L_{r_1}$.

Dissolving versus Time (3)

Steps of the proof:

- Let L_{r_1} be the witness language of the time hierarchy.
- Define $L_{\$,r_1}$ such that exactly one \$ is inserted after each of the rightmost $r_1(|w|)$ symbols of $w \in L_{r_1}$.
- Show that $L_{\$,r_1}$ is accepted by a real-time $r_1\text{-SOCA}$.

Dissolving versus Time (3)

Steps of the proof:

- Let L_{r_1} be the witness language of the time hierarchy.
- Define $L_{\$,r_1}$ such that exactly one \$ is inserted after each of the rightmost $r_1(|w|)$ symbols of $w \in L_{r_1}$.
- Show that $L_{\$,r_1}$ is accepted by a real-time $r_1\text{-SOCA}$.
- Assume that $L_{\$,r_1}$ is accepted by a real-time $r_2\text{-SOCA}$.

Dissolving versus Time (3)

Steps of the proof:

- Let L_{r_1} be the witness language of the time hierarchy.
- Define $L_{\$,r_1}$ such that exactly one \$ is inserted after each of the rightmost $r_1(|w|)$ symbols of $w \in L_{r_1}$.
- Show that $L_{\$,r_1}$ is accepted by a real-time r_1-SOCA.
- Assume that $L_{\$,r_1}$ is accepted by a real-time r_2-SOCA.
- Then, $L_{\$,r_1}$ is accepted by an $n + r_2(n)$-time OCA.

Dissolving versus Time (3)

Steps of the proof:

- Let L_{r_1} be the witness language of the time hierarchy.
- Define $L_{\$,r_1}$ such that exactly one \$ is inserted after each of the rightmost $r_1(|w|)$ symbols of $w \in L_{r_1}$.
- Show that $L_{\$,r_1}$ is accepted by a real-time r_1-SOCA.
- Assume that $L_{\$,r_1}$ is accepted by a real-time r_2-SOCA.
- Then, $L_{\$,r_1}$ is accepted by an $n + r_2(n)$-time OCA.
- Construct an $n + r_2(n)$-time OCA accepting L_{r_1}.

Dissolving versus Time (3)

Steps of the proof:

- Let L_{r_1} be the witness language of the time hierarchy.
- Define $L_{\$,r_1}$ such that exactly one \$ is inserted after each of the rightmost $r_1(|w|)$ symbols of $w \in L_{r_1}$.
- Show that $L_{\$,r_1}$ is accepted by a real-time $r_1\text{-SOCA}$.
- Assume that $L_{\$,r_1}$ is accepted by a real-time $r_2\text{-SOCA}$.
- Then, $L_{\$,r_1}$ is accepted by an $n + r_2(n)$-time OCA.
- Construct an $n + r_2(n)$-time OCA accepting L_{r_1}.

Applications:

- $\mathscr{L}_{rt}(n^p\text{-SOCA}) \subset \mathscr{L}_{rt}(n^q\text{-SOCA})$ for two rational numbers $0 \leq p < q \leq 1$.

Dissolving versus Time (3)

Steps of the proof:

- Let L_{r_1} be the witness language of the time hierarchy.
- Define $L_{\$,r_1}$ such that exactly one \$ is inserted after each of the rightmost $r_1(|w|)$ symbols of $w \in L_{r_1}$.
- Show that $L_{\$,r_1}$ is accepted by a real-time r_1-SOCA.
- Assume that $L_{\$,r_1}$ is accepted by a real-time r_2-SOCA.
- Then, $L_{\$,r_1}$ is accepted by an $n + r_2(n)$-time OCA.
- Construct an $n + r_2(n)$-time OCA accepting L_{r_1}.

Applications:

- $\mathscr{L}_{rt}(n^p\text{-SOCA}) \subset \mathscr{L}_{rt}(n^q\text{-SOCA})$ for two rational numbers $0 \leq p < q \leq 1$.
- $\mathscr{L}_{rt}(\log^{[j]}\text{-SOCA}) \subset \mathscr{L}_{rt}(\log^{[i]}\text{-SOCA})$ where $\log^{[i]}$ denotes the i-fold iterated logarithms and $0 < i < j$ are integers.