Characterising REGEX Languages by Regular Languages Equipped with Factor-Referencing

Markus L. Schmid

Trier University, Germany

DLT 2014

The Problems of Context-Freeness

- “The world” is not context-free.

The Problems of Context-Freeness

- "The world" is not context-free.
- Context-sensitive languages are often too powerful (vast expressive power, parsing intractable, undecidability, etc.)

The Problems of Context-Freeness

- "The world" is not context-free.
- Context-sensitive languages are often too powerful (vast expressive power, parsing intractable, undecidability, etc.)
- We often need language classes with some "non-context-free features", while at the same time weaker than context-sensitive.

The Problems of Context-Freeness

- "The world" is not context-free.
- Context-sensitive languages are often too powerful (vast expressive power, parsing intractable, undecidability, etc.)
- We often need language classes with some "non-context-free features", while at the same time weaker than context-sensitive.

regulated rewriting (add control mechanisms to context-free grammars)

mildly context-sensitive (allow only a little bit of context-sensitivity).

Typical Non-Context-Free Features

Reduplication	$\{ww \mid w \in \Sigma^*\}$
Multiple agreements	$\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 1\}$
Crossed agreements	$\{\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m \mid n, m \geq 1\}$

We solely focus on reduplication.

Typical Non-Context-Free Features

Reduplication	$\{ww \mid w \in \Sigma^*\}$
Multiple agreements	$\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 1\}$
Crossed agreements	$\{\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m \mid n, m \geq 1\}$

We solely focus on reduplication.

Language descriptor/grammars models tailored to reduplication:

- L systems,
- pattern languages,
- H-systems,
- Wijngaarden grammars, macro grammars, Indian parallel grammars, deterministic iteration grammars,
- pattern expressions, synchronized regular expressions, EH-expressions, extended regular expressions with backreferences (REGEX).

Reference-Words - Idea

$$a\;\underbrace{b\;a\;\rlap{$\overbrace{c\;b}^{y}$}c}_{x}\;\;c\;\overbrace{x\;c\;b}^{z}\;z\;y\;a$$

Reference-Words - Idea

$$\mathtt{a}\;\underbrace{\mathtt{b\;a\;c}}_{x}\;\mathtt{b}\;\mathtt{c}\;\overbrace{x\;\mathtt{c}\;\mathtt{b}}^{z}\;z\;y\;\mathtt{a}$$

Reference-Words - Idea

$$\mathtt{a}\,\underbrace{\mathtt{b\,a\,}\mathrlap{\overbrace{\mathtt{c\,b}}^{y}}\phantom{\mathtt{c}}}_{x}\phantom{\mathtt{\,b}}\,\mathtt{c}\,\overbrace{\mathtt{b\,a\,c\,c\,b}}^{z}\,z\,y\,\mathtt{a}$$

Reference-Words - Idea

$$\mathtt{a\,b\,a}\,\overbrace{\mathtt{c\,b}}^{y}\,\mathtt{c}\,\overbrace{\mathtt{b\,a\,c\,c\,b}}^{z}\,z\,y\,\mathtt{a}$$

Reference-Words - Idea

$$\mathtt{aba}\overbrace{\mathtt{cb}}^{y}\mathtt{c}\overbrace{\mathtt{baccb}}^{z}\,z\,y\,\mathtt{a}$$

Reference-Words - Idea

$$\mathtt{a\,b\,a}\,\overbrace{\mathtt{c\,b}}^{y}\,\mathtt{c}\,\overbrace{\mathtt{b\,a\,c\,c\,b}}^{z}\,z\,\mathtt{c\,b\,a}$$

Reference-Words - Idea

$$\mathtt{a\,b\,a\,c\,b\,c}\,\overbrace{\mathtt{b\,a\,c\,c\,b}}^{z}\,z\,\mathtt{c\,b\,a}$$

Reference-Words - Idea

$$\mathtt{abacbc}\overbrace{\mathtt{baccb}}^{z}z\,\mathtt{cba}$$

Reference-Words - Idea

$$\mathtt{a\,b\,a\,c\,b\,c}\,\overbrace{\mathtt{b\,a\,c\,c\,b}}^{z}\,\mathtt{b\,a\,c\,c\,b\,c\,b\,a}$$

Reference-Words - Idea

`a b a c b c b a c c b b a c c b c b a`

Reference-Words - Formal Definition

- Σ is a finite alphabet.

Reference-Words - Formal Definition

- Σ is a finite alphabet.
- $\Gamma = \{[_{x_i},]_{x_i}, x_i \mid i \in \mathbb{N}\}$.

Reference-Words - Formal Definition

- Σ is a finite alphabet.
- $\Gamma = \{[_{x_i},]_{x_i}, x_i \mid i \in \mathbb{N}\}$.
- $[_{x_i}$ and $]_{x_i}$ are called parentheses, x_i is called variable.

Reference-Words - Formal Definition

- Σ is a finite alphabet.
- $\Gamma = \{[_{x_i},]_{x_i}, x_i \mid i \in \mathbb{N}\}$.
- $[_{x_i}$ and $]_{x_i}$ are called parentheses, x_i is called variable.
- A ref-word over Σ is a word $w \in (\Sigma \cup \Gamma)^*$.

Reference-Words - Formal Definition

- Σ is a finite alphabet.
- $\Gamma = \{[_{x_i},]_{x_i}, x_i \mid i \in \mathbb{N}\}$.
- $[_{x_i}$ and $]_{x_i}$ are called parentheses, x_i is called variable.
- A ref-word over Σ is a word $w \in (\Sigma \cup \Gamma)^*$.
- A ref-word is valid if, for every $i \in \mathbb{N}$,
  - only well-formed, non-nested pairs of parentheses $[_{x_i},]_{x_i}$,
  - no x_i inside of $[_{x_i} \ldots]_{x_i}$.

Reference-Words - Formal Definition

- Σ is a finite alphabet.
- $\Gamma = \{[_{x_i},]_{x_i}, x_i \mid i \in \mathbb{N}\}$.
- $[_{x_i}$ and $]_{x_i}$ are called parentheses, x_i is called variable.
- A ref-word over Σ is a word $w \in (\Sigma \cup \Gamma)^*$.
- A ref-word is valid if, for every $i \in \mathbb{N}$,
  - only well-formed, non-nested pairs of parentheses $[_{x_i},]_{x_i}$,
  - no x_i inside of $[_{x_i} \ldots]_{x_i}$.
- $\Sigma^{[*]}$ is the set of valid ref-words (over Σ).

Reference-Words - Examples

- $[_x\ [_y\ \mathtt{b}\ ]_x\ \mathtt{c}x\ [_x\ \mathtt{b}\ ]_y\ zy\mathtt{b}\ [_y\ \mathtt{c}z\ ]_y\ z\ [_z\ \mathtt{cc}\ ]_x\ ]_z$,
- $[_x\ \mathtt{a}\ [_y\ \mathtt{b}\ [_z\ \mathtt{bba}\ ]_z\ \mathtt{c}\ ]_y\ \mathtt{b}y\mathtt{b}\ ]_x\ xy$.

Reference-Words - Examples

- $[_x\ [_y\ \mathtt{b}\ ]_x\ \mathtt{c}x\ [_x\ \mathtt{b}\ ]_y\ zy\mathtt{b}\ [_y\ \mathtt{c}z\ ]_y\ z\ [_z\ \mathtt{cc}\ ]_x\ ]_z$,
- $[_x\ \mathtt{a}\ [_y\ \mathtt{b}\ [_z\ \mathtt{bba}\ ]_z\ \mathtt{c}\ ]_y\ \mathtt{b}y\mathtt{b}\ ]_x\ xy$.

References for x with a value u: $[_x u]_x$.

Reference-Words - Examples

- $[_x \ [_y \ \mathtt{b} \ ]_x \ \mathtt{c}x \ [_x \ \mathtt{b} \ ]_y \ zy\mathtt{b} \ [_y \ \mathtt{c}z \ ]_y \ z \ [_z \ \mathtt{cc} \ ]_x \ ]_z$,
- $[_x \ \mathtt{a} \ [_y \ \mathtt{b} \ [_z \ \mathtt{bba} \ ]_z \ \mathtt{c} \ ]_y \ \mathtt{b}y\mathtt{b} \ ]_x \ xy$.

References for x with a value u: $[_x u]_x$.

An Occurrence of variable x refers to the reference for x, which precedes it.

Reference-Words - Examples

- $[_x\ [_y\ \mathtt{b}\ ]_x\ \mathtt{c}x\ [_x\ \mathtt{b}\ ]_y\ zy\mathtt{b}\ [_y\ \mathtt{c}z\ ]_y\ z\ [_z\ \mathtt{cc}\ ]_x\ ]_z$,
- $[_x\ \mathtt{a}\ [_y\ \mathtt{b}\ [_z\ \mathtt{bba}\ ]_z\ \mathtt{c}\ ]_y\ \mathtt{b}y\mathtt{b}\ ]_x\ xy$.

References for x with a value u: $[_x u]_x$.

An Occurrence of variable x refers to the reference for x, which precedes it.

Undefined variables: x not preceded by a reference for x.

Reference-Words - Examples

- $[_x\ [_y\ \texttt{b}\ ]_x\ \texttt{c}x\ [_x\ \texttt{b}\ ]_y\ zy\texttt{b}\ [_y\ \texttt{c}z\ ]_y\ z\ [_z\ \texttt{cc}\ ]_x\ ]_z$,
- $[_x\ \texttt{a}\ [_y\ \texttt{b}\ [_z\ \texttt{bba}\ ]_z\ \texttt{c}\ ]_y\ \texttt{b}y\texttt{b}\ ]_x\ xy$.

References for x with a value u: $[_x u]_x$.

An Occurrence of variable x refers to the reference for x, which precedes it.

Undefined variables: x not preceded by a reference for x.

Nested references: $[_x \ldots [_y \ldots]_y \ldots]_x$.

Reference-Words - Examples

- $[_x\ [_y\ \mathtt{b}\ ]_x\ \mathtt{c}x\ [_x\ \mathtt{b}\ ]_y\ zy\mathtt{b}\ [_y\ \mathtt{c}z\ ]_y\ z\ [_z\ \mathtt{cc}\ ]_x\ ]_z$,
- $[_x\ \mathtt{a}\ [_y\ \mathtt{b}\ [_z\ \mathtt{bba}\ ]_z\ \mathtt{c}\ ]_y\ \mathtt{b}y\mathtt{b}\ ]_x\ xy$.

References for x with a value u: $[_x u]_x$.

An Occurrence of variable x refers to the reference for x, which precedes it.

Undefined variables: x not preceded by a reference for x.

Nested references: $[_x \ldots [_y \ldots]_y \ldots]_x$.

Overlapping references: $[_x \ldots [_y \ldots]_x \ldots]_y$.

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \to \Sigma^*$

Example:

$$z\mathtt{a}[_z x[_x y\mathtt{b}[_y \mathtt{c}]_x \mathtt{b} x[_x \mathtt{c}]_y \mathtt{b}]_x y\mathtt{c}]_z x\mathtt{c}z$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \to \Sigma^*$

Example:

$$z\mathtt{a}[_z x [_x y \mathtt{b} [_y \mathtt{c}]_x \mathtt{b} x [_x \mathtt{c}]_y \mathtt{b}]_x y \mathtt{c}]_z x \mathtt{c} z$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \to \Sigma^*$

Example:

$$\mathtt{a}[_z [_x \mathtt{b} [_y \mathtt{c}]_x \mathtt{b} x [_x \mathtt{c}]_y \mathtt{b}]_x y \mathtt{c}]_z x \mathtt{c} z$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \rightarrow \Sigma^*$

Example:

$$\mathtt{a}[_z [_x \mathtt{b} [_y \mathtt{c}]_x \mathtt{b} x [_x \mathtt{c}]_y \mathtt{b}]_x y \mathtt{c}]_z x \mathtt{c} z$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \to \Sigma^*$

Example:

$$\mathtt{a}[_z[_x\mathtt{b}[_y\mathtt{c}]_x\mathtt{bbc}[_x\mathtt{c}]_y\mathtt{b}]_x y\,\mathtt{c}]_z x\,\mathtt{c}\,z$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \to \Sigma^*$

Example:

$$\mathtt{a}[_z\mathtt{b}[_y\mathtt{cbbc}[_x\mathtt{c}]_y\mathtt{b}]_x y\,\mathtt{c}]_z x\,\mathtt{c}\,z$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \to \Sigma^*$

Example:

$$\mathtt{a}[_z \mathtt{b} [_y \mathtt{cbbc} [_x \mathtt{c}]_y \mathtt{b}]_x \, y \, \mathtt{c}]_z \, x \, \mathtt{c} \, z$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \rightarrow \Sigma^*$

Example:

$$\mathtt{a}[_z\mathtt{b}[_y\mathtt{cbbc}[_x\mathtt{c}]_y\mathtt{b}]_x\mathtt{cbbccc}]_z x\mathtt{c}z$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \rightarrow \Sigma^*$

Example:

$$\mathtt{a}[_z\mathtt{bcbbc}[_x\mathtt{cb}]_x\mathtt{cbbccc}]_z x\mathtt{c}z$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \to \Sigma^*$

Example:

$$\mathtt{a}[_z\mathtt{bcbbc}[_x\mathtt{cb}]_x\mathtt{cbbccc}]_z\, x\,\mathtt{c}\, z$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \to \Sigma^*$

Example:

$$\mathtt{a}[_z\mathtt{bcbbc}[_x\mathtt{cb}]_x\mathtt{cbbccc}]_z\mathtt{cbc}z$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \to \Sigma^*$

Example:

$$\mathtt{a}[_z\mathtt{bcbbccbcbbccc}]_z\mathtt{cbc}z$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \rightarrow \Sigma^*$

Example:

$$\mathtt{a}[_z\mathtt{bcbbccbcbbccc}]_z\mathtt{cbc}z$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \rightarrow \Sigma^*$

Example:

$$\mathtt{a}[_z\mathtt{bcbbccbcbbccc}]_z\mathtt{cbcbcbbccbcbbccc}$$

Reference-Words - Dereference Function

$\mathcal{D} : \Sigma^{[*]} \rightarrow \Sigma^*$

Example:

`abcbbccbcbbccccbcbcbbccbcbbccc`

Ref-Languages and Ref-Regular Languages

L is a ref-language (over Σ) if

- $L \subseteq \Sigma^{[*]}$ and
- $L \subseteq (\Sigma \cup \{[_{x_i},]_{x_i}, x_i \mid i \leq k\})^*$, for some $k \in \mathbb{N}$.

Ref-Languages and Ref-Regular Languages

L is a ref-language (over Σ) if

- $L \subseteq \Sigma^{[*]}$ and
- $L \subseteq (\Sigma \cup \{[_{x_i},]_{x_i}, x_i \mid i \leq k\})^*$, for some $k \in \mathbb{N}$.

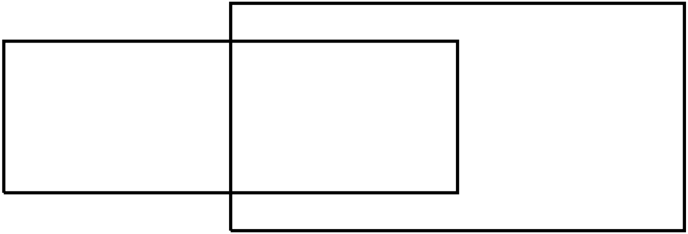
REG

Ref-Languages and Ref-Regular Languages

L is a ref-language (over Σ) if

- $L \subseteq \Sigma^{[*]}$ and
- $L \subseteq (\Sigma \cup \{[_{x_i},]_{x_i}, x_i \mid i \leq k\})^*$, for some $k \in \mathbb{N}$.

ref-lang.

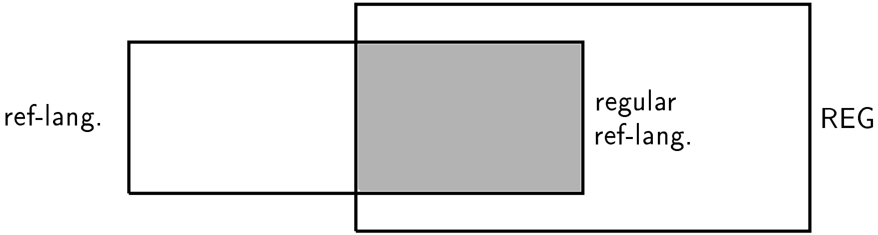


REG

Ref-Languages and Ref-Regular Languages

L is a ref-language (over Σ) if

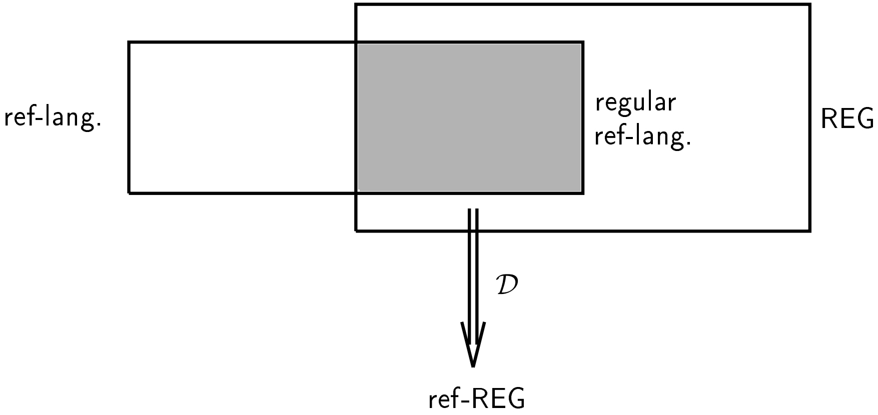
- $L \subseteq \Sigma^{[*]}$ and
- $L \subseteq (\Sigma \cup \{[_{x_i},]_{x_i}, x_i \mid i \leq k\})^*$, for some $k \in \mathbb{N}$.



Ref-Languages and Ref-Regular Languages

L is a ref-language (over Σ) if

- $L \subseteq \Sigma^{[*]}$ and
- $L \subseteq (\Sigma \cup \{[_{x_i},]_{x_i}, x_i \mid i \leq k\})^*$, for some $k \in \mathbb{N}$.



Ref-Languages and Ref-Regular Languages

- REG $\subset$ ref-REG $\subset$ CS,

Ref-Languages and Ref-Regular Languages

- REG $\subset$ ref-REG $\subset$ CS,
- $L_c = \{[_x\, w\,]_x\, x \mid w \in \Sigma^*\}$ is a regular ref-languages.
  $\mathcal{D}(L_c) = \{ww \mid w \in \Sigma^*\}$ (copy language).

Ref-Languages and Ref-Regular Languages

- REG $\subset$ ref-REG $\subset$ CS,
- $L_c = \{[_x\, w\,]_x\, x \mid w \in \Sigma^*\}$ is a regular ref-languages.
  $\mathcal{D}(L_c) = \{ww \mid w \in \Sigma^*\}$ (copy language).
- $\{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\} \notin$ ref-REG.

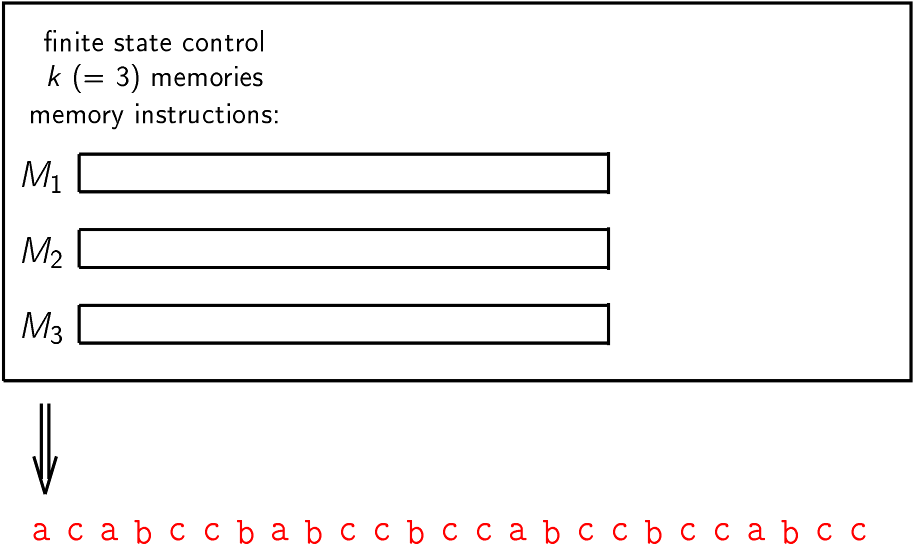
Memory Automata (MFA)

finite state control

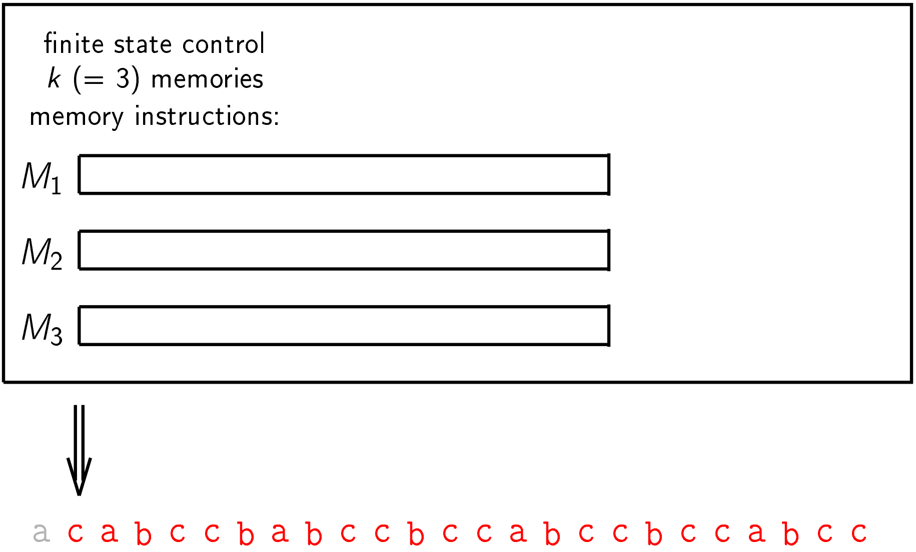
k $(= 3)$ memories

memory instructions:

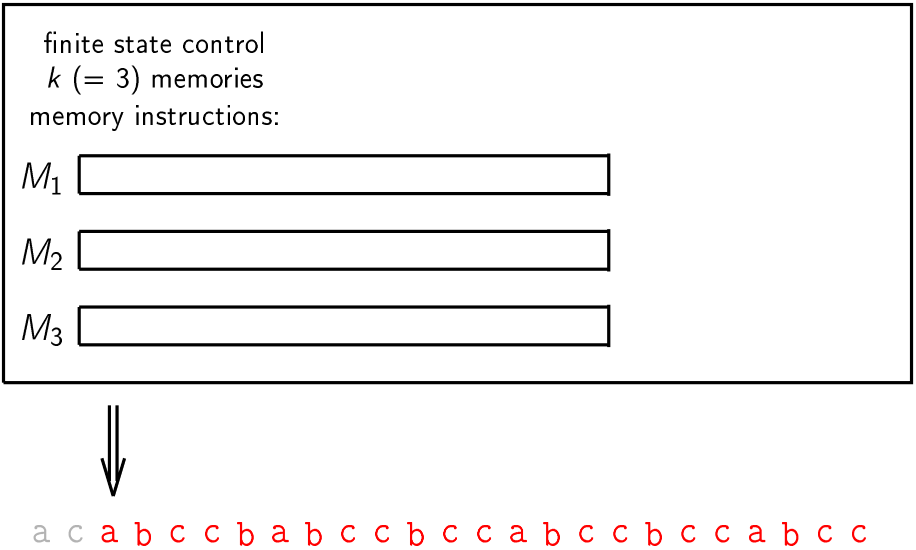
Memory Automata (MFA)



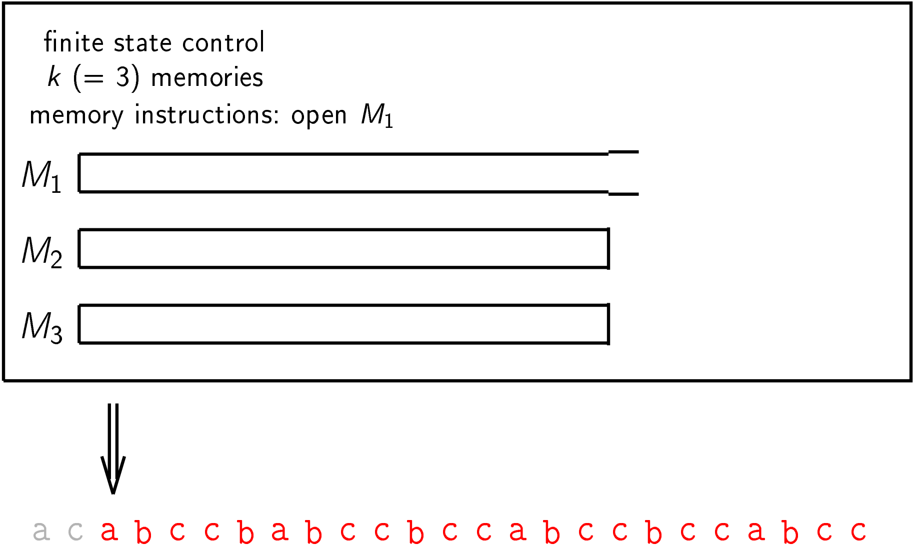
Memory Automata (MFA)



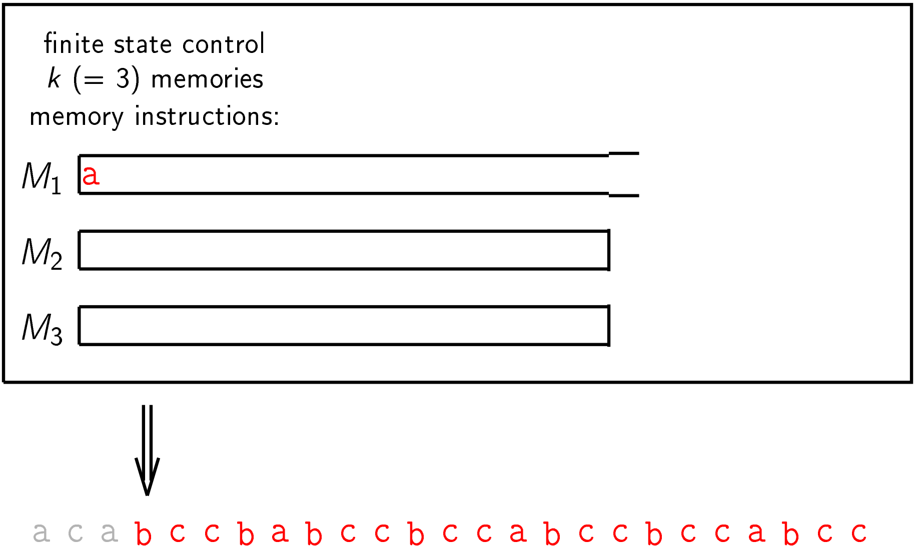
Memory Automata (MFA)



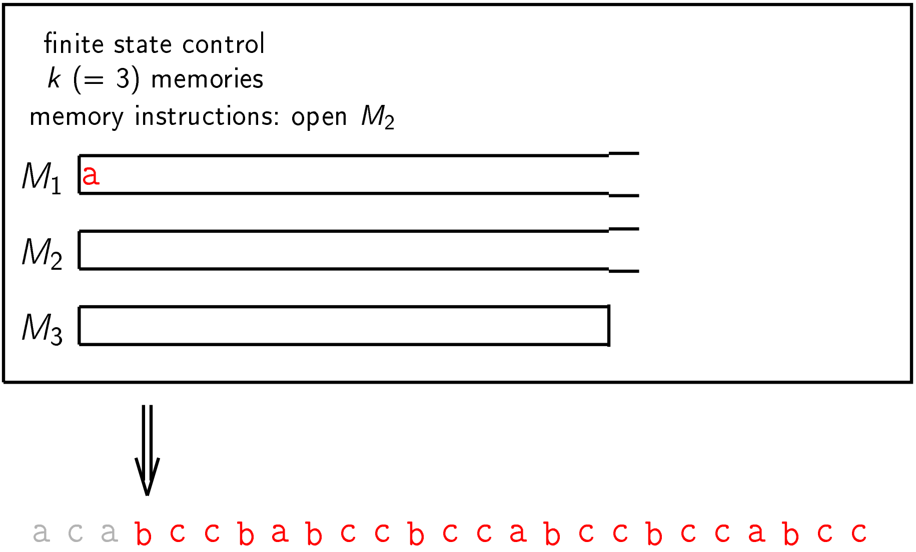
Memory Automata (MFA)



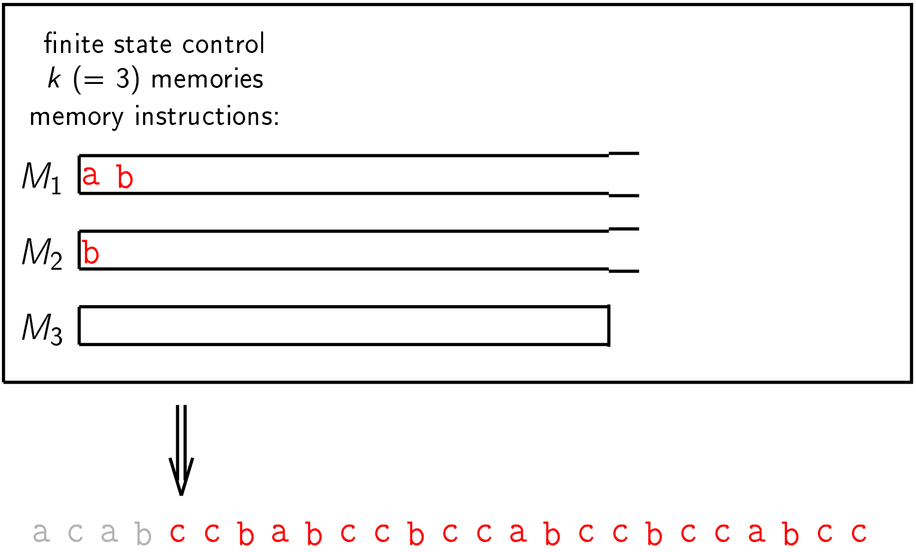
Memory Automata (MFA)



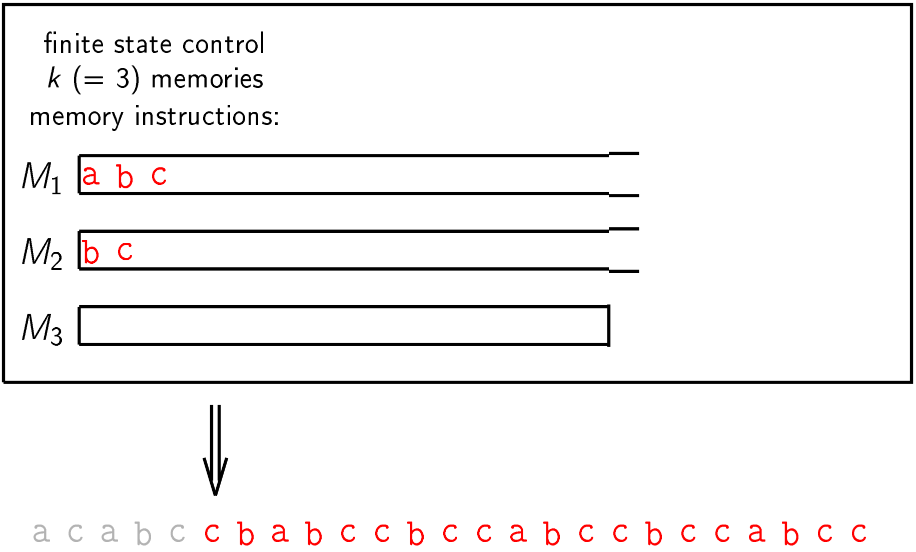
Memory Automata (MFA)



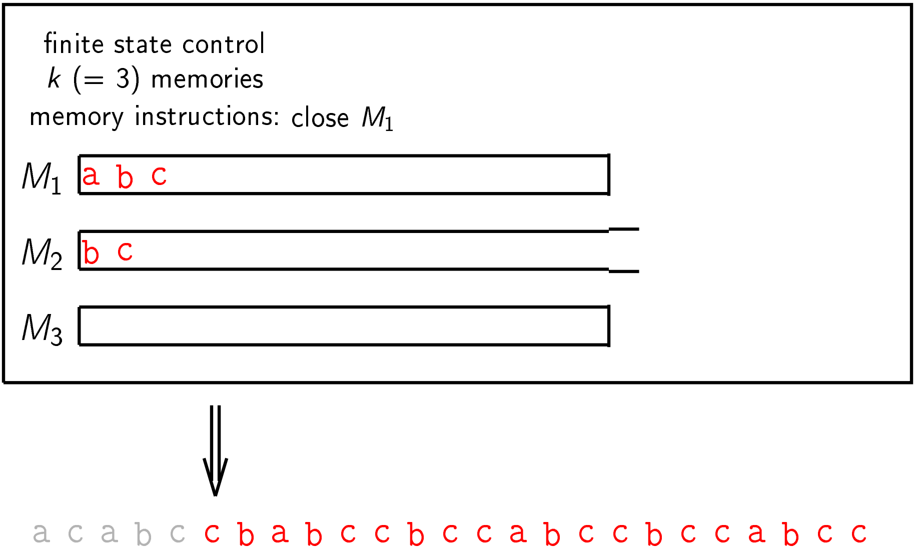
Memory Automata (MFA)



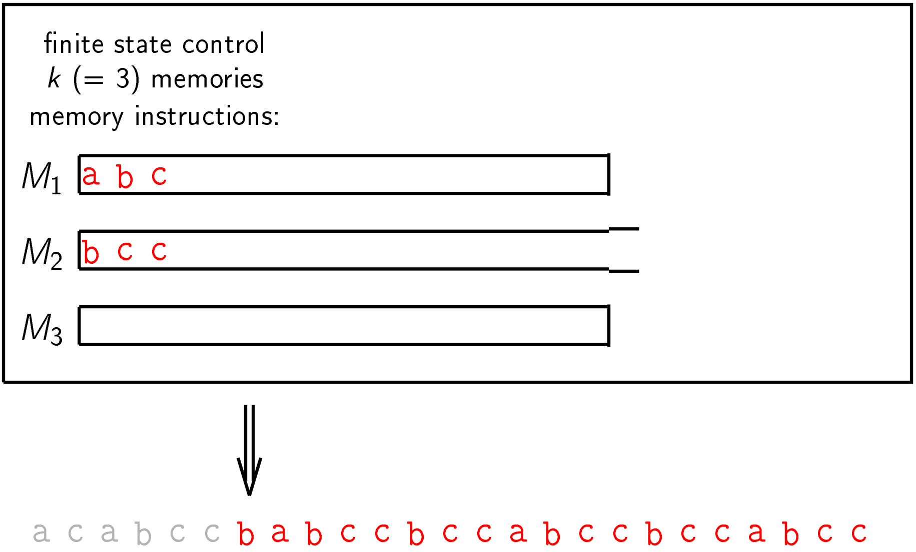
Memory Automata (MFA)



Memory Automata (MFA)



Memory Automata (MFA)



Memory Automata (MFA)

finite state control

k (= 3) memories

memory instructions: close M_2

M_1 a b c

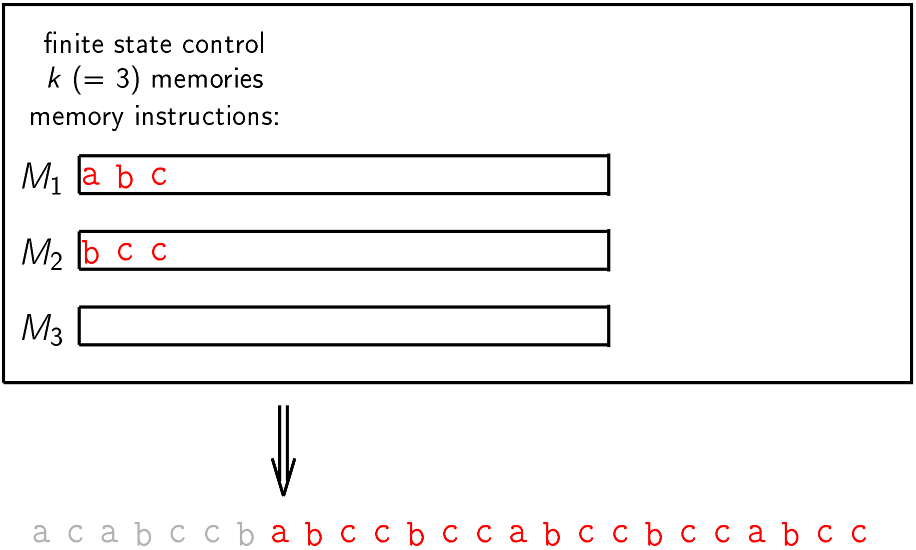
M_2 b c c

M_3

$\Downarrow$

a c a b c c b a b c c b c c a b c c b c c a b c c

Memory Automata (MFA)



Memory Automata (MFA)

finite state control

k $(= 3)$ memories

memory instructions: consult M_1

M_1 a b c

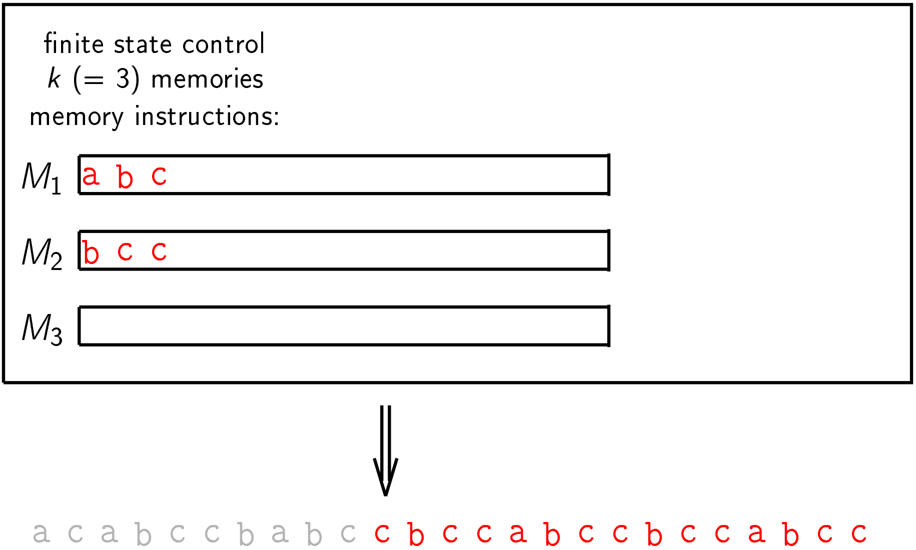
M_2 b c c

M_3

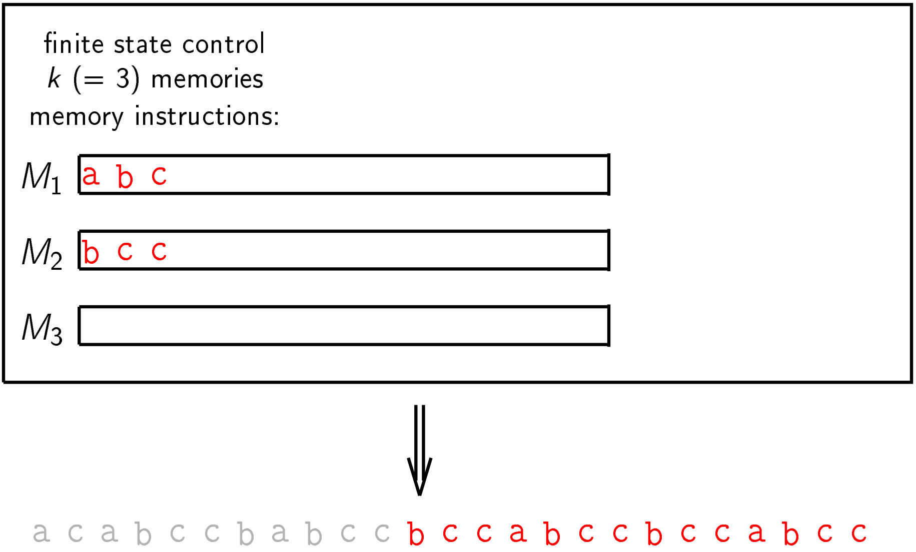
$\Downarrow$

a c a b c c b a b c c b c c a b c c b c c a b c c

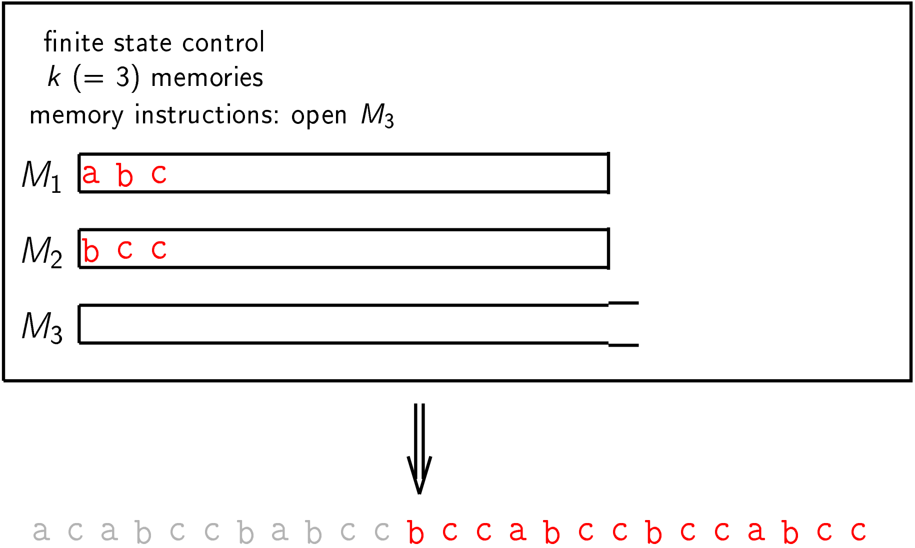
Memory Automata (MFA)



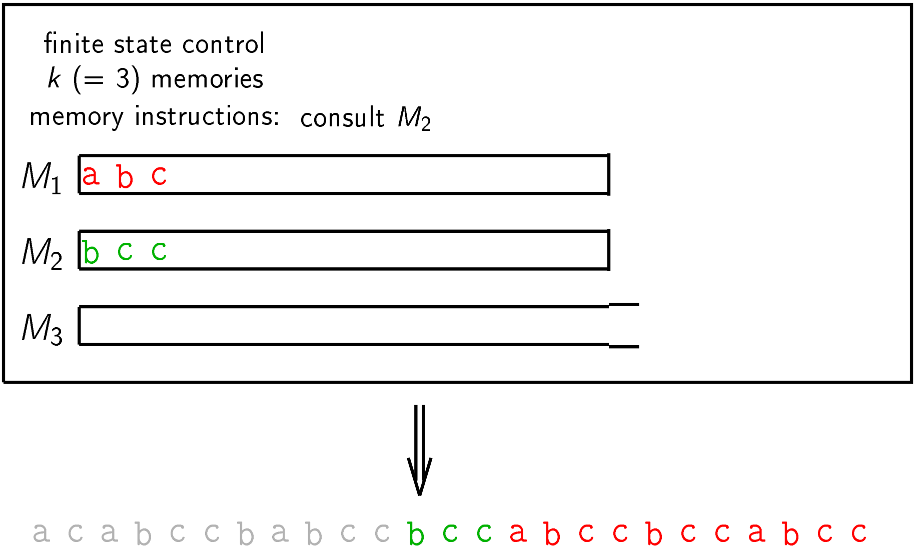
Memory Automata (MFA)



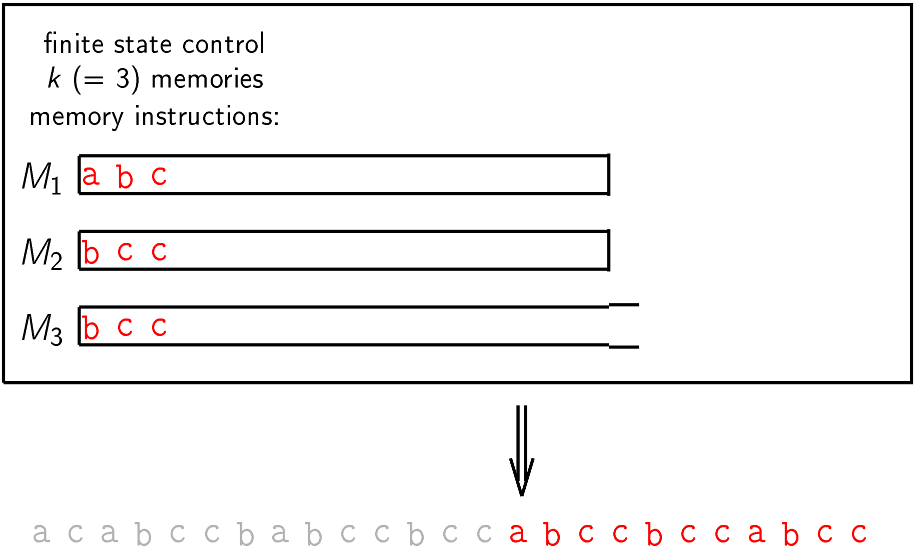
Memory Automata (MFA)



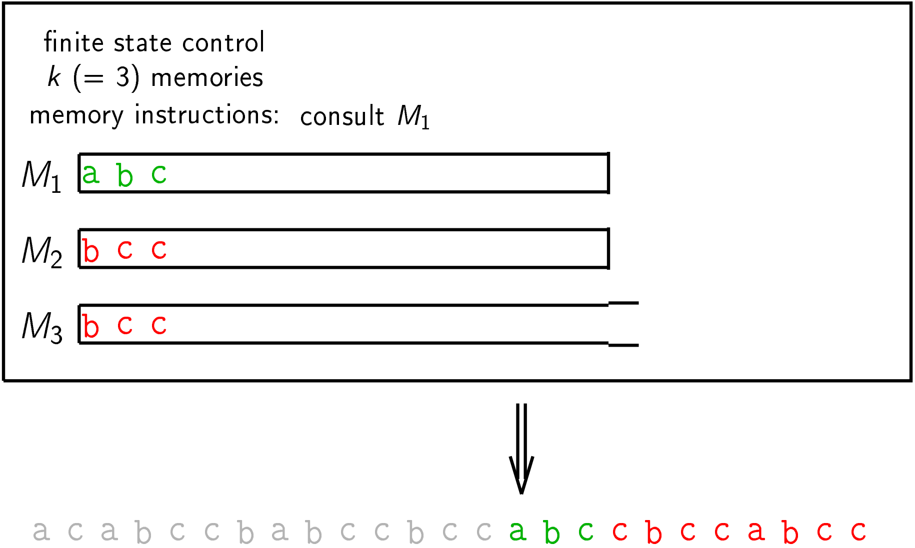
Memory Automata (MFA)



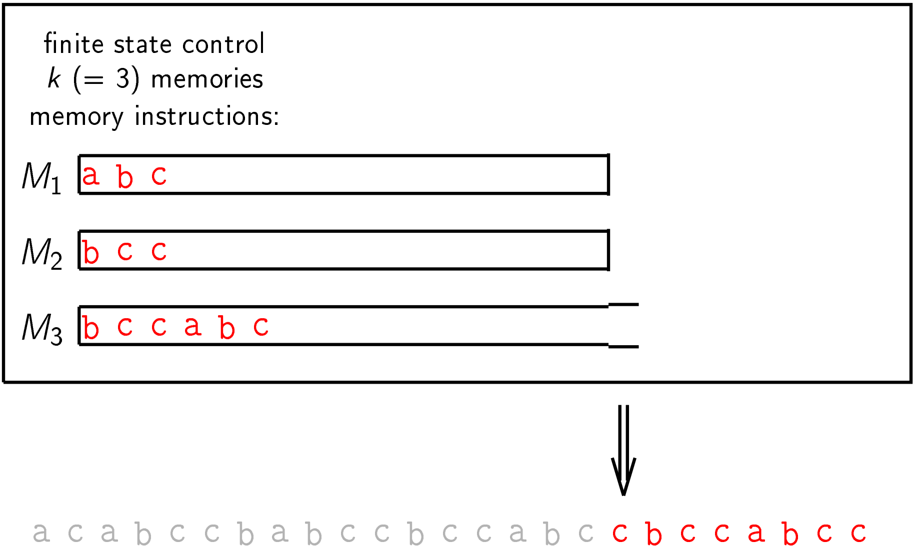
Memory Automata (MFA)



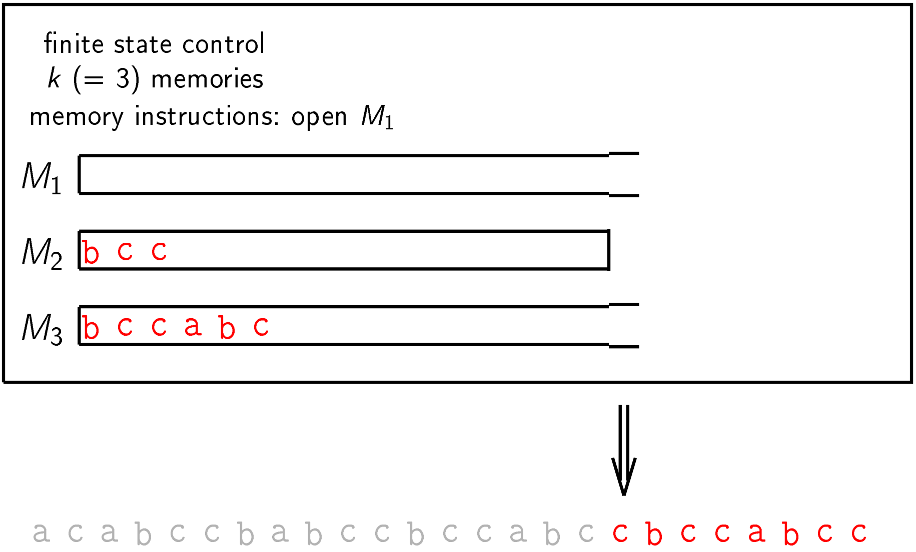
Memory Automata (MFA)



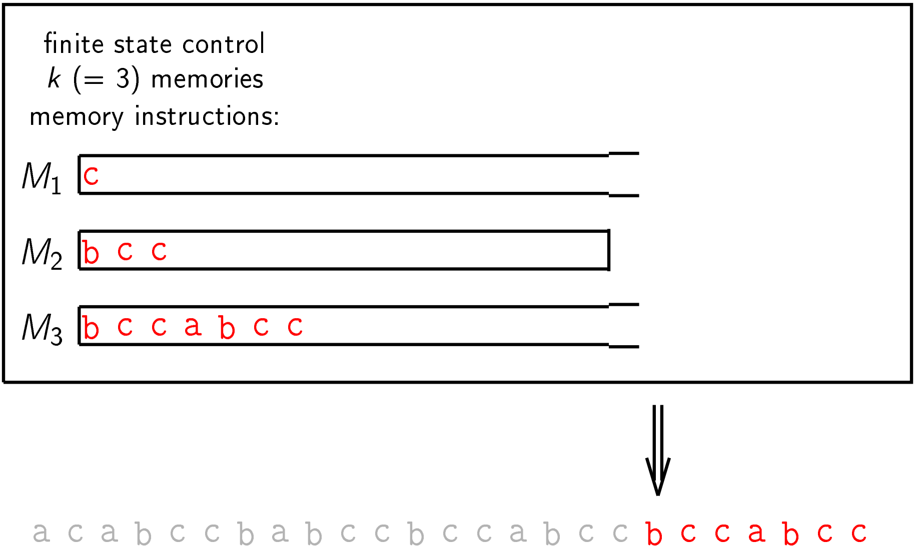
Memory Automata (MFA)



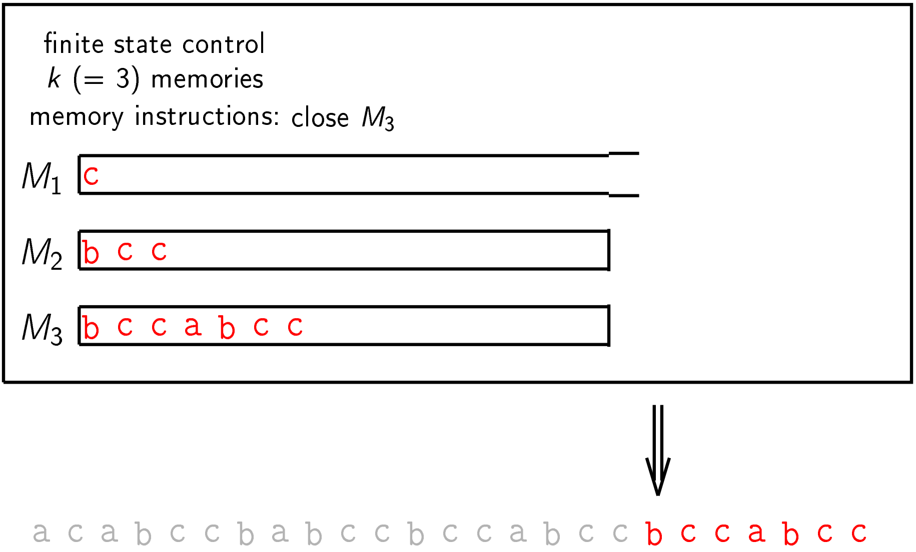
Memory Automata (MFA)



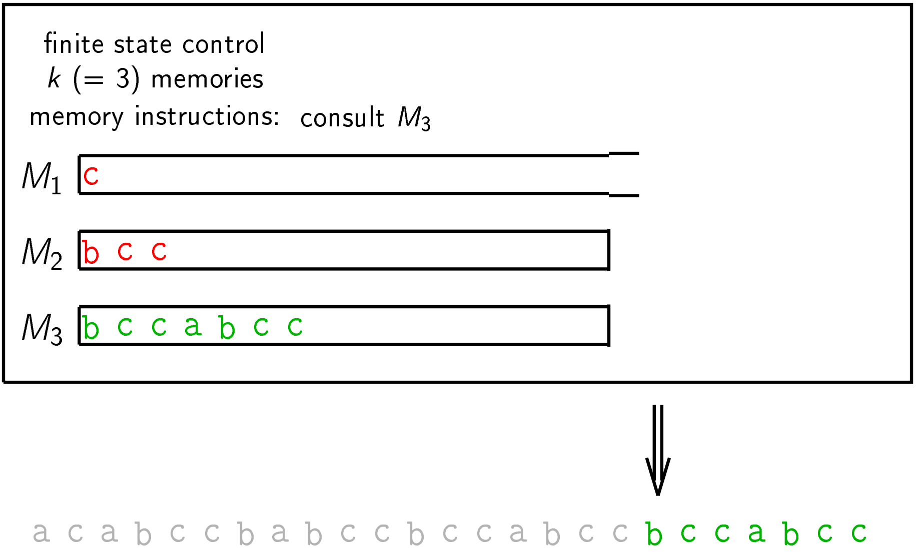
Memory Automata (MFA)



Memory Automata (MFA)



Memory Automata (MFA)



Memory Automata (MFA)

finite state control

k $(= 3)$ memories

memory instructions:

M_1 c b c c a b c c

M_2 b c c

M_3 b c c a b c c

$\Downarrow$

a c a b c c b a b c c b c c a b c c b c c a b c c

Memory Automata (MFA)

finite state control

k ($= 3$) memories

memory instructions: close M_1

M_1	c b c c a b c c
M_2	b c c
M_3	b c c a b c c

a c a b c c b a b c c b c c a b c c b c c a b c c

Determinism of MFA

- Pseudo deterministic: For every state, symbol b and memory i, at most one move that reads b and at most one move that consults memory i.

Determinism of MFA

- Pseudo deterministic: For every state, symbol b and memory i, at most one move that reads b and at most one move that consults memory i.
- Deterministic: ε-free and for every state at most one possible move.

Determinism of MFA

- Pseudo deterministic: For every state, symbol b and memory i, at most one move that reads b and at most one move that consults memory i.
- Deterministic: ε-free and for every state at most one possible move.

- $\mathcal{L}(\text{MFA}) = \mathcal{L}(\text{pseudo-det- MFA})$ (extended subset construction).

Determinism of MFA

- Pseudo deterministic: For every state, symbol b and memory i, at most one move that reads b and at most one move that consults memory i.
- Deterministic: ε-free and for every state at most one possible move.

- $\mathcal{L}(\text{MFA}) = \mathcal{L}(\text{pseudo-det- MFA})$ (extended subset construction).
- $\mathcal{L}(\text{DMFA}) \subset \mathcal{L}(\text{MFA})$ $(\{ww \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\} \notin \mathcal{L}(\text{DMFA}))$.

Nested MFA

An MFA is nested, if no two memories record factors that are overlapping, i. e.,

Nested MFA

An MFA is nested, if no two memories record factors that are overlapping, i. e.,

$$\mathtt{a}\,\rlap{$\underbrace{\phantom{\mathtt{b\,a\,c\,b\,a\,c}}}_{\text{memory 1}}$}\mathtt{b\,a}\,\overbrace{\mathtt{c\,b\,a\,c\,b\,a\,b}}^{\text{memory 2}}\,\mathtt{c\,c\,b}$$

is not possible.

Nested MFA

An MFA is nested, if no two memories record factors that are overlapping, i. e.,

$$\mathtt{a}\,\underbrace{\mathtt{b\,a\,}\rlap{$\overbrace{\mathtt{c\,b\,a\,c\,b\,a\,b}}^{\text{memory 2}}$}\phantom{\mathtt{c\,b\,a\,c}}}_{\text{memory 1}}\phantom{\mathtt{\,b\,a\,b}}\,\mathtt{c\,c\,b}$$

is not possible.

Theorem

Every MFA(k) *can be transformed into a equivalent* MFA(k^2) *that is pseudo-deterministic and nested.*

Equivalence of $\mathcal{L}$(MFA) and ref-REG

Theorem

ref-REG $= \mathcal{L}(\mathsf{MFA})$.

Equivalence of $\mathcal{L}$(MFA) and ref-REG

Theorem

ref-REG $= \mathcal{L}$(MFA).

Define $\psi_{\mathcal{D}} : \{M \in \text{NFA} \mid L(M) \subseteq \Sigma^{[*]}\} \to \text{MFA}$ by

Equivalence of $\mathcal{L}$(MFA) and ref-REG

Theorem

ref-REG $= \mathcal{L}$(MFA).

Define $\psi_{\mathcal{D}} : \{M \in \text{NFA} \mid L(M) \subseteq \Sigma^{[*]}\} \rightarrow$ MFA by

NFA reads $a \in \Sigma$	$\Rightarrow$	MFA reads $a \in \Sigma$
NFA reads $[_{x_i}$	$\Rightarrow$	MFA opens memory i
NFA reads $]_{x_i}$	$\Rightarrow$	MFA closes memory i
NFA reads x_i	$\Rightarrow$	MFA consults memory i

Equivalence of $\mathcal{L}(\mathrm{MFA})$ and ref-REG

Theorem

ref-REG $= \mathcal{L}(\mathrm{MFA})$.

Define $\psi_{\mathcal{D}} : \{M \in \mathrm{NFA} \mid L(M) \subseteq \Sigma^{[*]}\} \to \mathrm{MFA}$ by

NFA reads $a \in \Sigma$	$\Rightarrow$	MFA reads $a \in \Sigma$
NFA reads $[_{x_i}$	$\Rightarrow$	MFA opens memory i
NFA reads $]_{x_i}$	$\Rightarrow$	MFA closes memory i
NFA reads x_i	$\Rightarrow$	MFA consults memory i

Lemma

Let $M \in \mathrm{NFA}$ with $L(M) \subseteq \Sigma^{[*]}$. Then $\mathcal{D}(L(M)) = L(\psi_{\mathcal{D}}(M))$.

Equivalence of $\mathcal{L}(\text{MFA})$ and ref-REG

Theorem

ref-REG $= \mathcal{L}(\text{MFA})$.

Define $\psi_{\mathcal{D}} : \{M \in \text{NFA} \mid L(M) \subseteq \Sigma^{[*]}\} \to \text{MFA}$ by

NFA reads $a \in \Sigma$	$\Rightarrow$	MFA reads $a \in \Sigma$
NFA reads $[_{x_i}$	$\Rightarrow$	MFA opens memory i
NFA reads $]_{x_i}$	$\Rightarrow$	MFA closes memory i
NFA reads x_i	$\Rightarrow$	MFA consults memory i

Lemma

Let $M \in \text{NFA}$ *with* $L(M) \subseteq \Sigma^{[*]}$. *Then* $\mathcal{D}(L(M)) = L(\psi_{\mathcal{D}}(M))$.

Lemma

Let $M \in \text{MFA}$. *Then* $L(M) = \mathcal{D}(L(\psi_{\mathcal{D}}^{-1}(M)))$.

Extended Regular Expressions with Backreferences (REGEX)

REGEX = regular expressions with references to subexpressions.

Extended Regular Expressions with Backreferences (REGEX)

REGEX = regular expressions with references to subexpressions.

$r := (\,(\,\mathtt{a} \mid \mathtt{b}\,)^*\,)\,(\,\mathtt{c}^* \mid (\,\mathtt{a}^*\mathtt{b}\,)\,)$

Extended Regular Expressions with Backreferences (REGEX)

REGEX = regular expressions with references to subexpressions.

$r := (_1\,(\,\mathtt{a} \mid \mathtt{b}\,)^*\,)_1\,(\,\mathtt{c}^* \mid (_2\,\mathtt{a}^*\mathtt{b}\,)_2\,)$

Extended Regular Expressions with Backreferences (REGEX)

REGEX = regular expressions with references to subexpressions.

$r := (_1\,(\,\mathtt{a} \mid \mathtt{b}\,)^*\,)_1\,(\,\mathtt{c}^* \mid (_2\,\mathtt{a}^*\mathtt{b}\,)_2\,)\,(\,\backslash 2 \mid \mathtt{b}^*\,)\,(\,\backslash 1\,)^*$

Extended Regular Expressions with Backreferences (REGEX)

REGEX = regular expressions with references to subexpressions.

$r := (_1\, (\,\mathtt{a} \mid \mathtt{b}\,)^*\,)_1\, (\, \mathtt{c}^* \mid (_2\, \mathtt{a}^*\mathtt{b}\,)_2\,)\, (\, \backslash 2 \mid \mathtt{b}^*\,)\, (\, \backslash 1\,)^*$

Some background information about REGEX:

Extended Regular Expressions with Backreferences (REGEX)

REGEX = regular expressions with references to subexpressions.

$r := (_1\, (\,\mathtt{a} \mid \mathtt{b}\,)^*\,)_1\, (\,\mathtt{c}^* \mid (_2\, \mathtt{a}^*\mathtt{b}\,)_2\,)\, (\, \backslash 2 \mid \mathtt{b}^*\,)\, (\, \backslash 1\,)^*$

Some background information about REGEX:

- invented entirely on the level of software implementation,

Extended Regular Expressions with Backreferences (REGEX)

REGEX = regular expressions with references to subexpressions.

$r := (_1\,(\,\mathtt{a} \mid \mathtt{b}\,)^*\,)_1\,(\,\mathtt{c}^* \mid (_2\,\mathtt{a}^*\mathtt{b}\,)_2\,)\,(\,\backslash 2 \mid \mathtt{b}^*\,)\,(\,\backslash 1\,)^*$

Some background information about REGEX:

- invented entirely on the level of software implementation,
- applied in practice: Traditional and Modern `grep`, `vi`, Modern `sed`, GNU `Emacs`, `Perl`, `Python`, `Java`, `.Net`,

Extended Regular Expressions with Backreferences (REGEX)

REGEX = regular expressions with references to subexpressions.

$r := (_1\, (\mathtt{a} \mid \mathtt{b})^*\,)_1\, (\mathtt{c}^* \mid (_2\, \mathtt{a}^*\mathtt{b}\,)_2\,)\, (\, \backslash 2 \mid \mathtt{b}^*\,)\, (\, \backslash 1\,)^*$

Some background information about REGEX:

- invented entirely on the level of software implementation,
- applied in practice: Traditional and Modern `grep`, `vi`, Modern `sed`, GNU `Emacs`, `Perl`, `Python`, `Java`, .Net,
- *NP*-complete membership problem, undecidable inclusion problem,

Extended Regular Expressions with Backreferences (REGEX)

REGEX = regular expressions with references to subexpressions.

$r := (_1 (\mathtt{a} \mid \mathtt{b})^*)_1 (\mathtt{c}^* \mid (_2 \mathtt{a}^*\mathtt{b})_2) (\backslash 2 \mid \mathtt{b}^*) (\backslash 1)^*$

Some background information about REGEX:

- invented entirely on the level of software implementation,
- applied in practice: Traditional and Modern `grep`, `vi`, Modern `sed`, GNU `Emacs`, `Perl`, `Python`, `Java`, .Net,
- *NP*-complete membership problem, undecidable inclusion problem,
- language theoretical investigation started 10 years ago (Câmpeanu, K. Salomaa, Yu).

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

$\mathcal{L}(\text{REGEX}) \subseteq$ ref-REG.

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

$\mathcal{L}$(REGEX) $\subseteq$ ref-REG.

Proof sketch:

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

$\mathcal{L}(\text{REGEX}) \subseteq \text{ref-REG}$.

Proof sketch:

$$r := (_1\, (\,\mathtt{a} \mid \mathtt{b}\,)^*\,)_1\, (\,\mathtt{c}^* \mid (_2\, \mathtt{a}^*\mathtt{b}\,)_2\,)\,(\,\backslash 2 \mid \mathtt{b}^*\,)\,(\,\backslash 1\,)^*$$

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

$\mathcal{L}(\text{REGEX}) \subseteq$ ref-REG.

Proof sketch:

$$r := (_1\, (\,\mathtt{a} \mid \mathtt{b}\,)^*\,)_1\, (\,\mathtt{c}^* \mid (_2\, \mathtt{a}^*\mathtt{b}\,)_2\,)\, (\, \backslash 2 \mid \mathtt{b}^*\,)\, (\, \backslash 1\,)^*$$

$$r' := [_{x_1}\, (\,\mathtt{a} \mid \mathtt{b}\,)^*\,]_{x_1}\, (\,\mathtt{c}^* \mid [_{x_2}\, \mathtt{a}^*\mathtt{b}\,]_{x_2}\,)\, (\, x_2 \mid \mathtt{b}^*\,)\, (\, x_1\,)^*$$

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

$\mathcal{L}(\text{REGEX}) \subseteq \text{ref-REG}$.

Proof sketch:

$$r := (_1\,(\,\mathtt{a} \mid \mathtt{b}\,)^*\,)_1\,(\,\mathtt{c}^* \mid (_2\,\mathtt{a}^*\mathtt{b}\,)_2\,)\,(\,\backslash 2 \mid \mathtt{b}^*\,)\,(\,\backslash 1\,)^*$$

$$r' := [_{x_1}\,(\,\mathtt{a} \mid \mathtt{b}\,)^*\,]_{x_1}\,(\,\mathtt{c}^* \mid [_{x_2}\,\mathtt{a}^*\mathtt{b}\,]_{x_2}\,)\,(\,x_2 \mid \mathtt{b}^*\,)\,(\,x_1\,)^*$$

$$L(r) = \mathcal{D}(L(r'))$$

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

ref-REG $\subseteq \mathcal{L}$(REGEX)?

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

ref-REG $\subseteq \mathcal{L}$(REGEX)?

A regular expression r with $L(r) \in \Sigma^{[*]}$ has the REGEX property if all $[_x \ldots]_x$ enclose a subexpression of r.

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

ref-REG $\subseteq \mathcal{L}$(REGEX)?

A regular expression r with $L(r) \in \Sigma^{[*]}$ has the REGEX property if all $[_x \ldots]_x$ enclose a subexpression of r.

Let $L \in$ ref-REG and let r be a regular expression with $\mathcal{D}(L(r)) = L$. If r has the REGEX property, then $L \in \mathcal{L}$(REGEX).

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

ref-REG $\subseteq \mathcal{L}$(REGEX)?

A regular expression r with $L(r) \in \Sigma^{[*]}$ has the REGEX property if all $[_x \ldots]_x$ enclose a subexpression of r.

Let $L \in$ ref-REG and let r be a regular expression with $\mathcal{D}(L(r)) = L$. If r has the REGEX property, then $L \in \mathcal{L}$(REGEX).

$[_{x_2} [_{x_1} (\mathtt{a} \mid \mathtt{b})^*]_{x_1} \mathtt{c}^* x_1]_{x_2} \mathtt{a} x_2 x_1$

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

ref-REG $\subseteq \mathcal{L}$(REGEX)?

A regular expression r with $L(r) \in \Sigma^{[*]}$ has the REGEX property if all $[_x \ldots]_x$ enclose a subexpression of r.

Let $L \in$ ref-REG and let r be a regular expression with $\mathcal{D}(L(r)) = L$. If r has the REGEX property, then $L \in \mathcal{L}$(REGEX).

$[_{x_2} [_{x_1} (\mathtt{a} \mid \mathtt{b})^*]_{x_1} \mathtt{c}^* x_1]_{x_2} \mathtt{a} x_2 x_1 \Rightarrow (_2 (_1 (\mathtt{a} \mid \mathtt{b})^*)_1 \mathtt{c}^* \backslash 1)_2 \mathtt{a} \backslash 2 \backslash 1$

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

ref-REG $\subseteq \mathcal{L}$(REGEX)?

A regular expression r with $L(r) \in \Sigma^{[*]}$ has the REGEX property if all $[_x \ldots]_x$ enclose a subexpression of r.

Let $L \in$ ref-REG and let r be a regular expression with $\mathcal{D}(L(r)) = L$. If r has the REGEX property, then $L \in \mathcal{L}$(REGEX).

$[_{x_2}[_{x_1}(\mathtt{a} \mid \mathtt{b})^*]_{x_1}\mathtt{c}^* x_1]_{x_2}\mathtt{a} x_2 x_1 \Rightarrow (_2(_1(\mathtt{a} \mid \mathtt{b})^*)_1\mathtt{c}^*\backslash 1)_2\mathtt{a}\backslash 2\backslash 1$

$(([_{x_1}\mathtt{a}^*) \mid ([_{x_1}(\mathtt{a} \mid \mathtt{b})^*))((\mathtt{ca}]_{x_1} x_1) \mid]_{x_1}) x_1$

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

ref-REG $\subseteq \mathcal{L}$(REGEX)?

A regular expression r with $L(r) \in \Sigma^{[*]}$ has the REGEX property if all $[_x \ldots]_x$ enclose a subexpression of r.

Let $L \in$ ref-REG and let r be a regular expression with $\mathcal{D}(L(r)) = L$. If r has the REGEX property, then $L \in \mathcal{L}$(REGEX).

$[_{x_2} [_{x_1} (\mathtt{a} \mid \mathtt{b})^*]_{x_1} \mathtt{c}^* x_1]_{x_2} \mathtt{a} x_2 x_1 \Rightarrow (_2 (_1 (\mathtt{a} \mid \mathtt{b})^*)_1 \mathtt{c}^* \backslash 1)_2 \mathtt{a} \backslash 2 \backslash 1$

$(([_{x_1} \mathtt{a}^*) \mid ([_{x_1} (\mathtt{a} \mid \mathtt{b})^*))((\mathtt{ca}]_{x_1} x_1) \mid]_{x_1}) x_1 \Rightarrow ???$

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

ref-REG $\subseteq \mathcal{L}$(REGEX)?

A regular expression r with $L(r) \in \Sigma^{[*]}$ has the REGEX property if all $[_x \ldots]_x$ enclose a subexpression of r.

Let $L \in$ ref-REG and let r be a regular expression with $\mathcal{D}(L(r)) = L$. If r has the REGEX property, then $L \in \mathcal{L}$(REGEX).

$$[_{x_2} [_{x_1} (\mathtt{a} \mid \mathtt{b})^*]_{x_1} \mathtt{c}^* x_1]_{x_2} \mathtt{a} x_2 x_1 \Rightarrow (_2 (_1 (\mathtt{a} \mid \mathtt{b})^*)_1 \mathtt{c}^* \backslash 1)_2 \mathtt{a} \backslash 2 \backslash 1$$

$$(([_{x_1} \mathtt{a}^*) \mid ([_{x_1} (\mathtt{a} \mid \mathtt{b})^*))((\mathtt{ca}]_{x_1} x_1) \mid]_{x_1}) x_1 \Rightarrow ???$$

Question

Given a regular expression r with $L(r) \in \Sigma^{[*]}$. Is it possible to transform r into r' with the REGEX property and $\mathcal{D}(L(r)) = \mathcal{D}(L(r'))$.

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

ref-REG $\subseteq \mathcal{L}$(REGEX).

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

ref-REG $\subseteq \mathcal{L}$(REGEX).

Proof sketch:

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

ref-REG $\subseteq \mathcal{L}$(REGEX).

Proof sketch:

$L \in$ ref-REG.

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

ref-REG $\subseteq \mathcal{L}$(REGEX).

Proof sketch:

$L \in$ ref-REG.

$\exists$ nested MFA M with $L(M) = L$.

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

ref-REG $\subseteq \mathcal{L}$(REGEX).

Proof sketch:

$L \in$ ref-REG.

$\exists$ nested MFA M with $L(M) = L$.

$\exists$ NFA N with $L(N) = L' \in \Sigma^{[*]}$ and $\mathcal{D}(L') = L$.

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

ref-REG $\subseteq \mathcal{L}$(REGEX).

Proof sketch:

$L \in$ ref-REG.

$\exists$ nested MFA M with $L(M) = L$.

$\exists$ NFA N with $L(N) = L' \in \Sigma^{[*]}$ and $\mathcal{D}(L') = L$.

Transform N into a regular expression r with $L(r) = L(N)$ that has the REGEX property.

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

ref-REG $= \mathcal{L}(\text{MFA}) = \mathcal{L}(\text{REGEX})$.

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

ref-REG $= \mathcal{L}(\text{MFA}) = \mathcal{L}(\text{REGEX})$.

ref-regular languages are characterised by

- regular ref-languages,
  - finite automata accepting ref-languages,
  - regular expressions generating ref-languages,
  - ...

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

ref-REG $= \mathcal{L}(\text{MFA}) = \mathcal{L}(\text{REGEX})$.

ref-regular languages are characterised by

- regular ref-languages,
  - finite automata accepting ref-languages,
  - regular expressions generating ref-languages,
  - ...
- MFA

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

ref-REG $= \mathcal{L}(\text{MFA}) = \mathcal{L}(\text{REGEX})$.

ref-regular languages are characterised by

- regular ref-languages,
    - finite automata accepting ref-languages,
    - regular expressions generating ref-languages,
    - ...
- MFA
- REGEX (which can be considered a normal form of regular expressions generating ref-languages)

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

ref-REG $= \mathcal{L}(\text{MFA}) = \mathcal{L}(\text{REGEX})$.

ref-regular languages are characterised by

- regular ref-languages,
  - finite automata accepting ref-languages,
  - regular expressions generating ref-languages,
  - ...
- MFA
- REGEX (which can be considered a normal form of regular expressions generating ref-languages)

We have characterisations of REGEX-languages independent of REGEX.

Equivalence of ref-REG and $\mathcal{L}$(REGEX)

Theorem

ref-REG $= \mathcal{L}(\text{MFA}) = \mathcal{L}(\text{REGEX})$.

ref-regular languages are characterised by

- regular ref-languages,
  - finite automata accepting ref-languages,
  - regular expressions generating ref-languages,
  - ...
- MFA
- REGEX (which can be considered a normal form of regular expressions generating ref-languages)

We have characterisations of REGEX-languages independent of REGEX.

We have separated the "regular"-part from the "reduplication"-part.

Deterministic MFA Languages

$\mathcal{L}(\text{DMFA}) \subset \mathcal{L}(\text{MFA})\,(= \mathcal{L}(\text{REGEX}) = \text{ref-REG}).$

Deterministic MFA Languages

$\mathcal{L}(\mathrm{DMFA}) \subset \mathcal{L}(\mathrm{MFA})\,(= \mathcal{L}(\mathrm{REGEX}) = \text{ref-REG})$.

Theorem

The membership problem for DMFA-*languages:* $O(|w|)$.

Deterministic MFA Languages

$\mathcal{L}(\text{DMFA}) \subset \mathcal{L}(\text{MFA})\,(= \mathcal{L}(\text{REGEX}) = \text{ref-REG})$.

Theorem

The membership problem for DMFA-*languages:* $O(|w|)$.

Theorem

$\mathcal{L}(\text{DMFA})$ *is closed under*

- *complementation and*
- *intersection with regular languages,*

but it is not closed under

- *union or*
- *intersection.*

Deterministic MFA Languages

$\mathcal{L}(\text{DMFA}) \subset \mathcal{L}(\text{MFA})\,(= \mathcal{L}(\text{REGEX}) = \text{ref-REG})$.

Theorem

The membership problem for DMFA-*languages:* $O(|w|)$.

Theorem

The membership problem for ref-REG-*languages: NP-complete.*

Theorem

$\mathcal{L}(\text{DMFA})$ *is closed under*

- *complementation and*
- *intersection with regular languages,*

but it is not closed under

- *union or*
- *intersection.*

Deterministic MFA Languages

$\mathcal{L}(\text{DMFA}) \subset \mathcal{L}(\text{MFA})\,(= \mathcal{L}(\text{REGEX}) = \text{ref-REG})$.

Theorem

The membership problem for DMFA-*languages:* $O(|w|)$.

Theorem

The membership problem for ref-REG-*languages: NP-complete.*

Theorem

$\mathcal{L}(\text{DMFA})$ *is closed under*

- *complementation and*
- *intersection with regular languages,*

but it is not closed under

- *union or*
- *intersection.*

Theorem (Câmpeanu et al., Carle et al.)

ref-REG *is closed under*

- *union*
- *intersection with regular languages,*

but it is not closed under

- *complementation or*
- *intersection.*

Further Research Ideas

- Implementations of REGEX-engines based on MFA (or DMFA).

Further Research Ideas

- Implementations of REGEX-engines based on MFA (or DMFA).
- Descriptional complexity with respect to number of references in regular expression describing ref-languages, number of references in REGEX, number of memories of MFA.

Further Research Ideas

- Implementations of REGEX-engines based on MFA (or DMFA).
- Descriptional complexity with respect to number of references in regular expression describing ref-languages, number of references in REGEX, number of memories of MFA.
- Decision problems for ref-REG.

Further Research Ideas

- Implementations of REGEX-engines based on MFA (or DMFA).
- Descriptional complexity with respect to number of references in regular expression describing ref-languages, number of references in REGEX, number of memories of MFA.
- Decision problems for ref-REG.
- Investigate ref-$\mathcal{L}$ for other language classes $\mathcal{L}$, e. g., ref-CF.

Thank you very much for your attention.