Multiset Languages and Minimization Problem for Multiset Finite Automata

Pavel Martinek

Department of Mathematics
Faculty of Applied Informatics
Tomas Bata University in Zlín
Czech Republic

e-mail: pmartinek@utb.cz

LTA 2020, Brno

Outline of the talk

1. Introduction
2. Multiset grammars and multiset automata
3. Similarities and dissimilarities of multiset languages with
   - string languages of Chomsky hierarchy
   - languages accepted by jumping finite automata
4. Minimization problem for multiset finite automata
   - in classical form
   - in a generalized form
5. Conclusion

Introduction

The concept of multiset processing is present in various domains, e.g. in

- DNA computing
- membrane computing
- Petri nets
- chemical abstract machines
- etc.

Introduction

History of grammars which generate multisets starts with:

- Crespi-Reghizzi S., Mandrioli D., *Commutative grammars*, Calcolo, vol. XIII, fasc. II, 1976

Solid fundamentals were put (on the basis of Formal languages methodology) in the beginning of 21st century, namely by:

- Manfred Kudlek and his collaborators (long-term project *Multiset languages* at University of Hamburg, 29 papers in 9 years),
- Csuhaj-Varjú E., Martín-Vide C., and Mitrana V., *Multiset automata*, in Multiset processing — mathematical, computer science, and molecular computing points of view, LNCS 2235, Springer, 2001

Multiset grammars and multiset automata

Grammars ... generate strings of elements
(whose order is strict)

Multiset grammars ... generate multisets of elements
(no order of the elements
in the multiset is given)

Multiset grammars and multiset automata

Grammars ... generate strings of elements
(whose order is strict)

Multiset grammars ... generate multisets of elements
(no order of the elements
in the multiset is given)

Automata ... accept strings of elements

Multiset automata ... accept multisets of elements

Multisets

	Occurrence of its elements	Example
Classical set	single	$\{a, c, d\}$
Multiset	single or multiplied	$\{a, a, c, d, d, d\}$

Multisets

	Occurrence of its elements	Example
Classical set	single	$\{a, c, d\}$
Multiset	single or multiplied	$\{a, a, c, d, d, d\}$

We will write the multiset $\{a, a, c, d, d, d\}$ as

$$\langle a\rangle^2 \oplus \langle c\rangle \oplus \langle d\rangle^3$$

where

$\langle a\rangle$ denotes singleton multiset (with the only element a),

$\oplus$ denotes the operation of addition of multisets.

Multisets

	Occurrence of its elements	Example
Classical set	single	$\{a, c, d\}$
Multiset	single or multiplied	$\{a, a, c, d, d, d\}$

We will write the multiset $\{a, a, c, d, d, d\}$ as

$$\langle a\rangle^2 \oplus \langle c\rangle \oplus \langle d\rangle^3 \quad \text{or} \;\; (2, 0, 1, 3) \;\; \text{w.r.t.} \;\; \Sigma = \{a, b, c, d\}$$

where

$\langle a\rangle$ denotes singleton multiset (with the only element a),

$\oplus$ denotes the operation of addition of multisets.

Multisets

	Occurrence of its elements	Example
Classical set	single	$\{a, c, d\}$
Multiset	single or multiplied	$\{a, a, c, d, d, d\}$

We will write the multiset $\{a, a, c, d, d, d\}$ as

$$\langle a\rangle^2 \oplus \langle c\rangle \oplus \langle d\rangle^3$$

where

$\langle a\rangle$ denotes singleton multiset (with the only element a),

$\oplus$ denotes the operation of addition of multisets.

Further denotation:

$\mathbf{0}_\Sigma$ ... the empty multiset,

$\Sigma^\oplus$ ... the set of all multisets over alphabet Σ,

$\Sigma^{\langle\rangle}$ ... the set of all singleton multisets over alphabet Σ.

Multiset grammars

Multiset grammar: $G = (N, \Sigma, P, S)$ where

- N is an alphabet of nonterminals,
- Σ is an alphabet of terminals $(N \cap \Sigma = \emptyset)$,
- $P \subseteq \left[N^{\langle\rangle} \oplus (N \cup \Sigma)^{\oplus}\right] \times (N \cup \Sigma)^{\oplus}$ is a finite set of productions,
- $S \in N$ is the initial nonterminal.

Multiset grammars

Multiset grammar: $G = (N, \Sigma, P, S)$ where

- N is an alphabet of nonterminals,
- Σ is an alphabet of terminals $(N \cap \Sigma = \emptyset)$,
- $P \subseteq \left[N^{\langle\rangle} \oplus (N \cup \Sigma)^{\oplus}\right] \times (N \cup \Sigma)^{\oplus}$ is a finite set of productions,
- $S \in N$ is the initial nonterminal.

For $\mu_1, \mu_2 \in (N \cup \Sigma)^{\oplus}$, we define $\mu_1 \Rightarrow \mu_2$ if there are $(\alpha, \beta) \in P$, $\gamma \in (N \cup \Sigma)^{\oplus}$ such that $\mu_1 = \gamma \oplus \alpha$ and $\mu_2 = \gamma \oplus \beta$.

$\Rightarrow^*$ ... reflexive and transitive closure of the relation $\Rightarrow$

$M(G) = \{\omega \in \Sigma^{\oplus} | \langle S \rangle \Rightarrow^* \omega\}$ ... *the multiset language generated by G*

Chomsky-like classification of multiset grammars

1. Grammars G as above are called *arbitrary* (or *unrestricted*).
2. Grammars G with all productions $(\alpha, \beta) \in P$ restricted by the condition $|\alpha| \leq |\beta|$ (where $|\alpha|$ denotes cardinality of the multiset α) are called *monotone*.
3. Grammars G with all productions $(\alpha, \beta) \in P$ restricted by the condition $\alpha \in N^{\langle\rangle}$ are called *context-free*.
4. Grammars G with all productions $(\alpha, \beta) \in P$ restricted by the conditions $\alpha \in N^{\langle\rangle}$ and $\beta \in \left[N^{\langle\rangle} \oplus \Sigma^{\langle\rangle} \cup \Sigma^{\langle\rangle}\right]$ are called *regular*.

Context-free multiset grammar

Example: Let $G = (\{S, A, B\}, \{a, b\}, P, S)$ where

$P = \{(\langle S\rangle, \langle S\rangle \oplus \langle S\rangle)\,,\ (\langle S\rangle, \langle A\rangle \oplus \langle B\rangle)\,,\ (\langle A\rangle, \langle a\rangle)\,,\ (\langle B\rangle, \langle b\rangle)\}$.

Then: $\langle S\rangle \Rightarrow \langle A\rangle \oplus \langle B\rangle \Rightarrow \langle a\rangle \oplus \langle B\rangle \Rightarrow \langle a\rangle \oplus \langle b\rangle$,

$\langle S\rangle \Rightarrow \langle S\rangle \oplus \langle S\rangle \Rightarrow^* \langle a\rangle \oplus \langle b\rangle \oplus \langle a\rangle \oplus \langle b\rangle = \langle a\rangle^2 \oplus \langle b\rangle^2$,

$\langle S\rangle \Rightarrow \langle S\rangle \oplus \langle S\rangle \Rightarrow \langle S\rangle \oplus \langle S\rangle \oplus \langle S\rangle \Rightarrow^* \langle a\rangle^3 \oplus \langle b\rangle^3$,

etc.

Context-free multiset grammar

Example: Let $G = (\{S, A, B\}, \{a, b\}, P, S)$ where
$P = \{(\langle S\rangle, \langle S\rangle \oplus \langle S\rangle)\,,\ (\langle S\rangle, \langle A\rangle \oplus \langle B\rangle)\,,\ (\langle A\rangle, \langle a\rangle)\,,\ (\langle B\rangle, \langle b\rangle)\}$.

Then: $\langle S\rangle \Rightarrow \langle A\rangle \oplus \langle B\rangle \Rightarrow \langle a\rangle \oplus \langle B\rangle \Rightarrow \langle a\rangle \oplus \langle b\rangle$,

$\langle S\rangle \Rightarrow \langle S\rangle \oplus \langle S\rangle \Rightarrow^* \langle a\rangle \oplus \langle b\rangle \oplus \langle a\rangle \oplus \langle b\rangle = \langle a\rangle^2 \oplus \langle b\rangle^2$,

$\langle S\rangle \Rightarrow \langle S\rangle \oplus \langle S\rangle \Rightarrow \langle S\rangle \oplus \langle S\rangle \oplus \langle S\rangle \Rightarrow^* \langle a\rangle^3 \oplus \langle b\rangle^3$,

etc.

Hence $\langle S\rangle \Rightarrow^* \langle a\rangle \oplus \langle b\rangle$,

$\langle S\rangle \Rightarrow^* \langle a\rangle^2 \oplus \langle b\rangle^2$,

$\langle S\rangle \Rightarrow^* \langle a\rangle^3 \oplus \langle b\rangle^3$,

etc.

Context-free multiset grammar

Example: Let $G = (\{S, A, B\}, \{a, b\}, P, S)$ where
$P = \{(\langle S\rangle, \langle S\rangle \oplus \langle S\rangle), \ (\langle S\rangle, \langle A\rangle \oplus \langle B\rangle), \ (\langle A\rangle, \langle a\rangle), \ (\langle B\rangle, \langle b\rangle)\}$.

Then: $\langle S\rangle \Rightarrow \langle A\rangle \oplus \langle B\rangle \Rightarrow \langle a\rangle \oplus \langle B\rangle \Rightarrow \langle a\rangle \oplus \langle b\rangle$,

$\langle S\rangle \Rightarrow \langle S\rangle \oplus \langle S\rangle \Rightarrow^* \langle a\rangle \oplus \langle b\rangle \oplus \langle a\rangle \oplus \langle b\rangle = \langle a\rangle^2 \oplus \langle b\rangle^2$,

$\langle S\rangle \Rightarrow \langle S\rangle \oplus \langle S\rangle \Rightarrow \langle S\rangle \oplus \langle S\rangle \oplus \langle S\rangle \Rightarrow^* \langle a\rangle^3 \oplus \langle b\rangle^3$,

etc.

Hence $\langle S\rangle \Rightarrow^* \langle a\rangle \oplus \langle b\rangle$,

$\langle S\rangle \Rightarrow^* \langle a\rangle^2 \oplus \langle b\rangle^2$,

$\langle S\rangle \Rightarrow^* \langle a\rangle^3 \oplus \langle b\rangle^3$,

etc.

Obviously: $M(G) = \{\alpha \in \{a, b\}^{\oplus} \mid |\alpha|_a = |\alpha|_b > 0\}$.

Chomsky-like classification of multiset languages

Definition: Multiset languages generated by arbitrary, monotone, context-free and regular grammars are called *arbitrary*, *monotone*, *context-free* and *regular*, respectively.

Chomsky-like classification of multiset languages

Definition: Multiset languages generated by arbitrary, monotone, context-free and regular grammars are called *arbitrary*, *monotone*, *context-free* and *regular*, respectively.

Assertion: The family of multiset context-free languages is equal to the family of multiset regular languages.

Proof directly follows from Parikh's theorem.

Multiset finite automata

A *multiset finite automaton* (mFA): $A = (Q, \Sigma, \delta, q_0, F)$ where

- Q is a nonempty finite set of states,
- Σ is an input alphabet,
- $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation,
- q_0 is the initial state,
- $F \subseteq Q$ is a set of final states.

Multiset finite automata

A *multiset finite automaton* (mFA): $A = (Q, \Sigma, \delta, q_0, F)$ where

- Q is a nonempty finite set of states,
- Σ is an input alphabet,
- $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation,
- q_0 is the initial state,
- $F \subseteq Q$ is a set of final states.

A *configuration:* $(q, \mu) \in Q \times \Sigma^{\oplus}$.

A *computational step* is a relation $\vdash \subseteq (Q \times \Sigma^{\oplus}) \times (Q \times \Sigma^{\oplus})$ defined by $(q, \langle a \rangle \oplus \mu) \vdash (q', \mu)$ iff $(q, a, q') \in \delta$.

$\vdash^*$ denotes the reflexive and transitive closure of $\vdash$.

Multiset finite automata

A *multiset finite automaton* (mFA): $A = (Q, \Sigma, \delta, q_0, F)$ where

- Q is a nonempty finite set of states,
- Σ is an input alphabet,
- $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation,
- q_0 is the initial state,
- $F \subseteq Q$ is a set of final states.

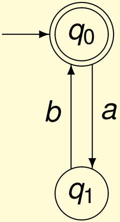
A *configuration:* $(q, \mu) \in Q \times \Sigma^{\oplus}$.
A *computational step* is a relation $\vdash \subseteq (Q \times \Sigma^{\oplus}) \times (Q \times \Sigma^{\oplus})$ defined by $(q, \langle a \rangle \oplus \mu) \vdash (q', \mu)$ iff $(q, a, q') \in \delta$.
$\vdash^*$ denotes the reflexive and transitive closure of $\vdash$.

The *multiset language* $M(A)$ accepted by A is defined by

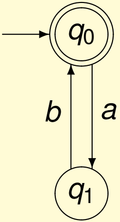
$$M(A) = \{\omega \in \Sigma^{\oplus} | (q_0, \omega) \vdash^* (q_f, \mathbf{0}_\Sigma) \text{ for some } q_f \in F\}.$$

Multiset finite automata

A comparison:



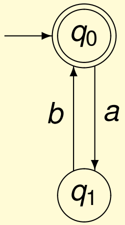
Finite automaton A



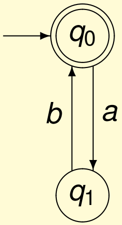
Multiset finite automaton B

Multiset finite automata

A comparison:



Finite automaton A



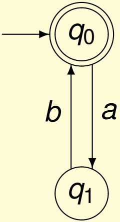
Multiset finite automaton B

For example

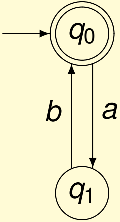
- the finite automaton A accepts the string *abab* and does not accept the string *aabb*,
- the multiset finite automaton B accepts the multiset $\{a, a, b, b\}$ (alternatively written as $\langle a \rangle^2 \oplus \langle b \rangle^2$).

Multiset finite automata

A comparison:



Finite automaton A



Multiset finite automaton B

Accepted languages

- $L(A) = \{(ab)^n \mid n \geq 0\}$,
- $M(B) = \{\langle a \rangle^n \oplus \langle b \rangle^n \mid n \geq 0\}$.

Chomsky hierarchy of string languages

Languages	Grammars	Automata
recursively enumerable	arbitrary	Turing machine
monotone	monotone	linear bounded automaton
context-free	context-free	pushdown automaton
regular	regular	finite automaton

Chomsky hierarchy of string languages

Languages	Grammars	Automata
recursively enumerable	arbitrary	Turing machine
monotone	monotone	linear bounded automaton
context-free	context-free	pushdown automaton
regular	regular	finite automaton

$$\begin{array}{c} \text{RE} \\ \upuparrows \\ \text{MON} \\ \upuparrows \\ \text{CF} \\ \upuparrows \\ \text{REG} \end{array}$$

$\upuparrows$ ... proper inclusion

Chomsky-like hierarchy of multiset languages

Multiset languages	Multiset grammars	Automata
arbitrary	arbitrary	multiset Turing machine
monotone	monotone	multiset linear bounded automaton
regular	context-free & regular	multiset finite automaton

Chomsky-like hierarchy of multiset languages

Multiset languages	Multiset grammars	Automata
arbitrary	arbitrary	multiset Turing machine
monotone	monotone	multiset linear bounded automaton
regular	context-free & regular	multiset finite automaton

$$\begin{array}{c} \text{mARB} \\ \uparrow \\ \text{mMON} \\ \uparrow\uparrow \\ \text{mREG} \end{array}$$

$\uparrow$ ... inclusion with unclear properness

$\uparrow\uparrow$ ... proper inclusion

Similarities and dissimilarities of multiset languages with string languages

- Similarities – some (we use techniques and concepts invented for exploration of string languages, for example generating and accepting devices).
- Dissimilarities – usual due to work with multisets (their elements are not ordered).

Similarities and dissimilarities of m-languages with string languages

$$\begin{array}{c} \mathrm{RE} = \mathcal{L}(\mathrm{TM}) \\ \upuparrows \\ \mathrm{MON} \\ \upuparrows \\ \mathrm{CF} \\ \upuparrows \\ \mathrm{REG} = \mathcal{L}(\mathrm{FA}) \end{array}$$

Similarities and dissimilarities of m-languages with string languages

Matrix grammar with appearance checking: $G = (N, \Sigma, M, S, F)$ where

- N, Σ and $S \in N$ are as in a context-free grammar
- $M = \{m_1, m_2, \ldots, m_n\}$ is a finite set of finite sequences of context-free productions (incl. erasing productions) using symbols from $N \cup \Sigma \cup \{\varepsilon\}$.
- F is a subset of productions contained in M.

$x \Rightarrow y$ with $x, y \in (N \cup \Sigma)^*$ ... a direct derivation which uses productions of a sequence $m_i \in M$

1. either all of them one by one
2. or productions contained in F can be omitted if they cannot be applied; the other productions must be used (respecting their order in the sequence)

$\Rightarrow^*$ ... reflexive and transitive closure of the relation $\Rightarrow$

$L(G) = \{w \in \Sigma^* | S \Rightarrow^* w\}$

Similarities and dissimilarities of m-languages with string languages

Example: Let $G = (\{S, X, Y\}, \{a, b\}, \{m_1, m_2, m_3, m_4\}, S, \emptyset)$ where

$$m_1 = (S \to XX)\,,$$
$$m_2 = (X \to aY,\ X \to aX,\ Y \to X)\,,$$
$$m_3 = (X \to bY,\ X \to bX,\ Y \to X)\,,$$
$$m_4 = (X \to \varepsilon,\ X \to \varepsilon)\,.$$

Similarities and dissimilarities of m-languages with string languages

Example: Let $G = (\{S, X, Y\}, \{a, b\}, \{m_1, m_2, m_3, m_4\}, S, \emptyset)$
where

$$m_1 = (S \to XX)\,,$$
$$m_2 = (X \to aY,\ X \to aX,\ Y \to X)\,,$$
$$m_3 = (X \to bY,\ X \to bX,\ Y \to X)\,,$$
$$m_4 = (X \to \varepsilon,\ X \to \varepsilon)\,.$$

We have for all $w \in \{a, b\}^*$:

$$(m_2)\ \ wXwX \cdots\!> waYwX \cdots\!> waYwaX \cdots\!> waXwaX,$$
$$(m_3)\ \ wXwX \cdots\!> wbYwX \cdots\!> wbYwbX \cdots\!> wbXwbX,$$
$$(m_4)\ \ wXwX \cdots\!> wwX \cdots\!> ww.$$

Similarities and dissimilarities of m-languages with string languages

Example: Let $G = (\{S, X, Y\}, \{a, b\}, \{m_1, m_2, m_3, m_4\}, S, \emptyset)$ where

$$m_1 = (S \to XX),$$
$$m_2 = (X \to aY,\ X \to aX,\ Y \to X),$$
$$m_3 = (X \to bY,\ X \to bX,\ Y \to X),$$
$$m_4 = (X \to \varepsilon,\ X \to \varepsilon).$$

We have for all $w \in \{a, b\}^*$:

$$(m_2)\ \ wXwX \cdots> waYwX \cdots> waYwaX \cdots> waXwaX,$$
$$(m_3)\ \ wXwX \cdots> wbYwX \cdots> wbYwbX \cdots> wbXwbX,$$
$$(m_4)\ \ wXwX \cdots> wwX \cdots> ww.$$

Hence $S \Rightarrow_{m_1} XX \Rightarrow^*_{m_2,m_3} wXwX \Rightarrow_{m_4} ww$.

So, $L(G) = \{ww \mid w \in \{a, b\}^*\}$.

Similarities and dissimilarities of m-languages with string languages

$$\begin{array}{c} \mathrm{RE} = \mathcal{L}(\mathrm{TM}) = \mathrm{MAT}_{ac} \\ \upuparrows \\ \mathrm{MON} \\ \upuparrows \\ \mathrm{CF} \\ \upuparrows \\ \mathrm{REG} = \mathcal{L}(\mathrm{FA}) \end{array}$$

Similarities and dissimilarities of m-languages with string languages

$$\begin{array}{l} \mathrm{RE} = \mathcal{L}(\mathrm{TM}) = \mathrm{MAT}_{ac} \\ \upuparrows \\ \mathrm{MON} \\ \upuparrows \\ \mathrm{CF} \\ \upuparrows \\ \mathrm{REG} = \mathcal{L}(\mathrm{FA}) \end{array}$$

$$\begin{array}{l} \mathrm{mARB} = \mathcal{L}(\mathrm{mTM}) \\ \uparrow \\ \mathrm{mMON} \\ \upuparrows \\ \mathrm{mCF} = \mathrm{mREG} = \mathcal{L}(\mathrm{mFA}) \end{array}$$

Similarities and dissimilarities of m-languages with string languages

$$
\begin{array}{l}
\mathrm{RE} = \mathcal{L}(\mathrm{TM}) = \mathrm{MAT}_{ac} \\
\upuparrows \\
\mathrm{MON} \\
\upuparrows \\
\mathrm{CF} \\
\upuparrows \\
\mathrm{REG} = \mathcal{L}(\mathrm{FA})
\end{array}
\qquad
\begin{array}{l}
\mathrm{mMAT}_{ac} \\
\upuparrows \\
\mathrm{mARB} = \mathcal{L}(\mathrm{mTM}) \\
\uparrow \\
\mathrm{mMON} \\
\upuparrows \\
\mathrm{mCF} = \mathrm{mREG} = \mathcal{L}(\mathrm{mFA})
\end{array}
$$

Similarities and dissimilarities of m-languages with string languages

$$\mathrm{mMAT}_{ac} = \mathcal{L}(\mathrm{mTM}_{ac})$$
$$\upuparrows$$
$$\mathrm{mARB} = \mathcal{L}(\mathrm{mTM})$$
$$\uparrow$$
$$\mathrm{mMON}$$
$$\upuparrows$$
$$\mathrm{mCF} = \mathrm{mREG} = \mathcal{L}(\mathrm{mFA})$$

$$\mathrm{RE} = \mathcal{L}(\mathrm{TM}) = \mathrm{MAT}_{ac}$$
$$\upuparrows$$
$$\mathrm{MON}$$
$$\upuparrows$$
$$\mathrm{CF}$$
$$\upuparrows$$
$$\mathrm{REG} = \mathcal{L}(\mathrm{FA})$$

Similarities and dissimilarities of m-languages with string languages

$$\begin{array}{c} \mathrm{RE} = \mathcal{L}(\mathrm{TM}) = \mathrm{MAT}_{ac} \\ \upuparrows \\ \mathrm{MON} \\ \upuparrows \\ \mathrm{CF} \\ \upuparrows \\ \mathrm{REG} = \mathcal{L}(\mathrm{FA}) = \mathcal{L}(\mathrm{dFA}) \end{array}$$

$$\begin{array}{c} \mathrm{mMAT}_{ac} = \mathcal{L}(\mathrm{mTM}_{ac}) \\ \upuparrows \\ \mathrm{mARB} = \mathcal{L}(\mathrm{mTM}) \\ \uparrow \\ \mathrm{mMON} \\ \upuparrows \\ \mathrm{mCF} = \mathrm{mREG} = \mathcal{L}(\mathrm{mFA}) \end{array}$$

Similarities and dissimilarities of m-languages with string languages

$$
\begin{array}{l}
\mathrm{RE} = \mathcal{L}(\mathrm{TM}) = \mathrm{MAT}_{ac} \\
\upuparrows \\
\mathrm{MON} \\
\upuparrows \\
\mathrm{CF} \\
\upuparrows \\
\mathrm{REG} = \mathcal{L}(\mathrm{FA}) = \mathcal{L}(\mathrm{dFA})
\end{array}
$$

$$
\begin{array}{lr}
\mathrm{mMAT}_{ac} = \mathcal{L}(\mathrm{mTM}_{ac}) & \\
\upuparrows & \\
\mathrm{mARB} = \mathcal{L}(\mathrm{mTM}) & \\
\uparrow & \\
\mathrm{mMON} & \\
\upuparrows & \\
\mathrm{mCF} = \mathrm{mREG} = \mathcal{L}(\mathrm{mFA}) & \\
& \upuparrows \\
& \mathcal{L}(\mathrm{dmFA})
\end{array}
$$

Similarities and dissimilarities of m-languages with string languages

Definition: An mFA $A = (Q, \Sigma, \delta, q_0, F)$ is said to be *deterministic* (we write dmFA) if the following condition is satisfied:

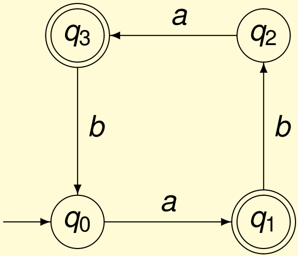
For all $q, r, r' \in Q, a, a' \in \Sigma,$ if $(q, a, r) \in \delta$ and $(q, a', r') \in \delta$, then $a = a'$ and $r = r'$.

Similarities and dissimilarities of m-languages with string languages

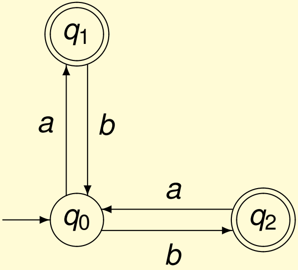
Definition: An mFA $A = (Q, \Sigma, \delta, q_0, F)$ is said to be *deterministic* (we write dmFA) if the following condition is satisfied:

For all $q, r, r' \in Q, a, a' \in \Sigma,$ if $(q, a, r) \in \delta$ and $(q, a', r') \in \delta$, then $a = a'$ and $r = r'$.

Example: dmFA A:



nondeterministic mFA B:

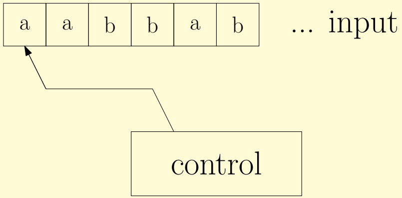


Similarities and dissimilarities of multiset languages with string languages accepted by jumping finite automata

- Similarities – wide (the concepts of multiset and jumping finite automata have a lot in common).
- Dissimilarities – rare (despite the difference between strings and multisets).

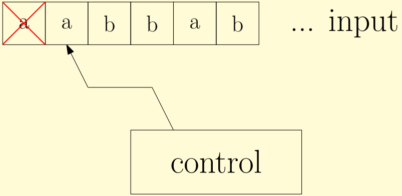
String languages accepted by jumping finite automata

Finite automaton:



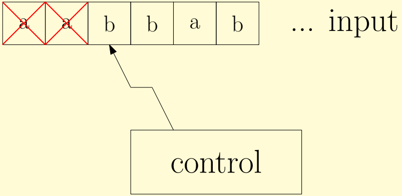
String languages accepted by jumping finite automata

Finite automaton:



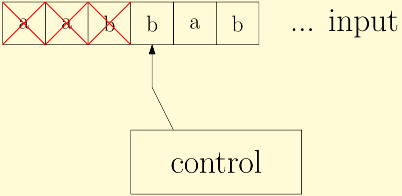
String languages accepted by jumping finite automata

Finite automaton:



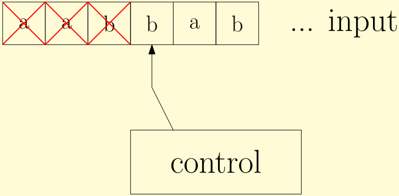
String languages accepted by jumping finite automata

Finite automaton:

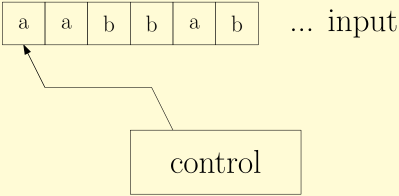


String languages accepted by jumping finite automata

Finite automaton:

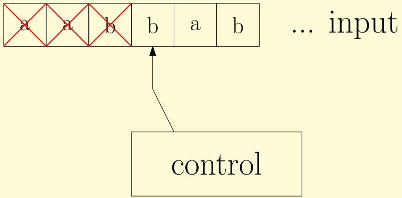


Jumping finite automaton:

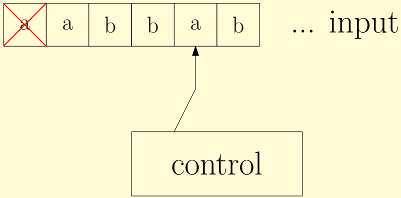


String languages accepted by jumping finite automata

Finite automaton:

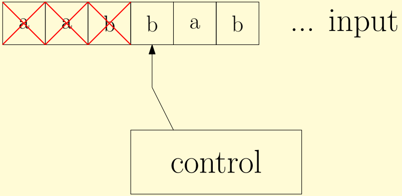


Jumping finite automaton:

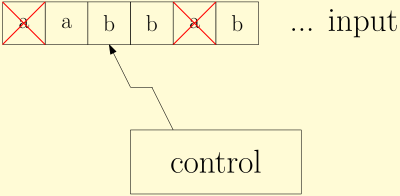


String languages accepted by jumping finite automata

Finite automaton:

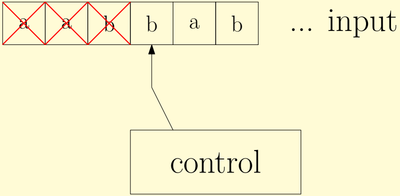


Jumping finite automaton:

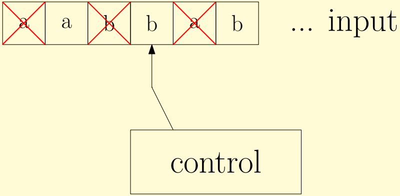


String languages accepted by jumping finite automata

Finite automaton:



Jumping finite automaton:



String languages accepted by jumping finite automata

A *jumping finite automaton* (JFA): $A = (Q, \Sigma, \delta, q_0, F)$ where

- Q is a nonempty finite set of states,
- Σ is an input alphabet, $\Sigma \cap Q = \emptyset$,
- $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation,
- q_0 is the initial state,
- $F \subseteq Q$ is a set of final states.

String languages accepted by jumping finite automata

A *jumping finite automaton* (JFA): $A = (Q, \Sigma, \delta, q_0, F)$ where

- Q is a nonempty finite set of states,
- Σ is an input alphabet, $\Sigma \cap Q = \emptyset$,
- $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation,
- q_0 is the initial state,
- $F \subseteq Q$ is a set of final states.

A *configuration:* $uqv \in \Sigma^* Q \Sigma^*$ (uv is the not yet processed content of the input string)

A *jumping relation* is a relation $\curvearrowright \subseteq \Sigma^* Q \Sigma^* \times \Sigma^* Q \Sigma^*$ defined by $(uqav, u'rv') \in \curvearrowright$ iff $(q, a, r) \in \delta$ and $uv = u'v'$.

$\curvearrowright^*$ denotes the reflexive and transitive closure of $\curvearrowright$.

String languages accepted by jumping finite automata

A *jumping finite automaton* (JFA): $A = (Q, \Sigma, \delta, q_0, F)$ where

- Q is a nonempty finite set of states,
- Σ is an input alphabet, $\Sigma \cap Q = \emptyset$,
- $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation,
- q_0 is the initial state,
- $F \subseteq Q$ is a set of final states.

A *configuration:* $uqv \in \Sigma^* Q \Sigma^*$ (uv is the not yet processed content of the input string)

A *jumping relation* is a relation $\curvearrowright \subseteq \Sigma^* Q \Sigma^* \times \Sigma^* Q \Sigma^*$ defined by $(uqav, u'rv') \in \curvearrowright$ iff $(q, a, r) \in \delta$ and $uv = u'v'$.

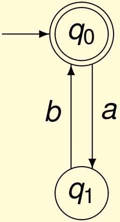
$\curvearrowright^*$ denotes the reflexive and transitive closure of $\curvearrowright$.

The *language* $L(A)$ accepted by A is defined by

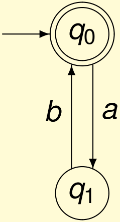
$$L(A) = \{uv \mid u, v \in \Sigma^*, (uq_0v, q_f) \in \curvearrowright^* \text{ for some } q_f \in F\}.$$

String languages accepted by jumping finite automata

A comparison:



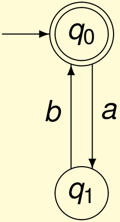
Finite automaton A



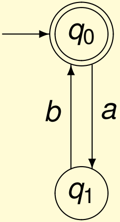
Jumping finite automaton B

String languages accepted by jumping finite automata

A comparison:



Finite automaton A



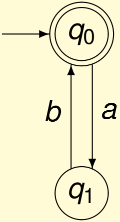
Jumping finite automaton B

For example

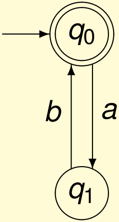
- the finite automaton A accepts the string *abab* and does not accept the string *aabb*,
- the jumping finite automaton B accepts both the string *abab* and the string *aabb*.

String languages accepted by jumping finite automata

A comparison:



Finite automaton A



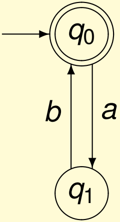
Jumping finite automaton B

Accepted languages

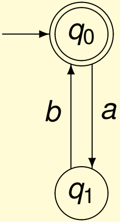
- $L(A) = \{(ab)^n \mid n \geq 0\}$,
- $L(B) = \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$.

String languages accepted by jumping finite automata

Another comparison:



Multiset finite automaton A



Jumping finite automaton B

Accepted languages

- $M(A) = \{\langle a \rangle^n \oplus \langle b \rangle^n \mid n \geq 0\}$,
- $L(B) = \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$.

Similarities and dissimilarities of multiset languages with languages accepted by jumping finite automata

Theorem: If a language $L \subseteq \Sigma^*$ is accepted by some jumping finite automaton then for every $w \in L$, any permutation of symbols in w is in L.

Similarities and dissimilarities of multiset languages with languages accepted by jumping finite automata

<u>Theorem</u>: If a language $L \subseteq \Sigma^*$ is accepted by some jumping finite automaton then for every $w \in L$, any permutation of symbols in w is in L.

<u>Definition</u>: Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be an alphabet. The mapping $\Psi : \Sigma^* \to \mathbb{N}^n$ such that

$$\Psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n}) \text{ for any } w \in \Sigma^*$$

is called *Parikh mapping* (over Σ). Here, $|w|_{a_i}$ denotes the number of occurences of a_i in w and $\mathbb{N}$ is the set of non-negative integers.

Similarities and dissimilarities of multiset languages with languages accepted by jumping finite automata

Theorem: If a language $L \subseteq \Sigma^*$ is accepted by some jumping finite automaton then for every $w \in L$, any permutation of symbols in w is in L.

Definition: Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be an alphabet. The mapping $\Psi : \Sigma^* \to \mathbb{N}^n$ such that

$$\Psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n}) \text{ for any } w \in \Sigma^*$$

is called *Parikh mapping* (over Σ). Here, $|w|_{a_i}$ denotes the number of occurences of a_i in w and $\mathbb{N}$ is the set of non-negative integers.

Example: $\Sigma = \{a, b, c\}$, $v = abaa$, $w = aaab$,

$$\Psi(v) = (3, 1, 0) = \Psi(w)$$

Similarities and dissimilarities of multiset languages with languages accepted by jumping finite automata

Theorem: If a language $L \subseteq \Sigma^*$ is accepted by some jumping finite automaton then for every $w \in L$, any permutation of symbols in w is in L.

Definition: Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be an alphabet. The mapping $\Psi : \Sigma^* \to \mathbb{N}^n$ such that

$$\Psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n}) \text{ for any } w \in \Sigma^*$$

is called *Parikh mapping* (over Σ). Here, $|w|_{a_i}$ denotes the number of occurences of a_i in w and $\mathbb{N}$ is the set of non-negative integers.

For any language $L \subseteq \Sigma^*$, we define $\Psi(L) = \{\Psi(w) \mid w \in L\}$.

Example: If $\Sigma = \{a, b\}$ and $L = \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$, then $\Psi(L) = \{(n, n) \mid n \in \mathbb{N}\}$.

Similarities and dissimilarities of multiset languages with languages accepted by jumping finite automata

Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$.

- Image of any $w \in \Sigma^*$ by Parikh mapping is the n-tuple

  $$(|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n}).$$

- A multiset $\alpha \in \Sigma^{\oplus}$ can be represented as the n-tuple

  $$(|\alpha|_{a_1}, |\alpha|_{a_2}, \ldots, |\alpha|_{a_n}).$$

So, Parikh mapping of a language $L \subseteq \Sigma^*$ represents a multiset language.

Similarities and dissimilarities of multiset languages with languages accepted by jumping finite automata

Theorem: If a language $L \subseteq \Sigma^*$ is accepted by some jumping finite automaton then $\Psi(L)$ is accepted by some multiset finite automaton.

Proof: Straightforward. (Both automata have identical state diagrams.)

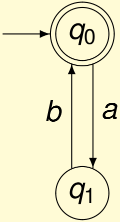
Theorem: If a multiset language $M \subseteq \Sigma^{\oplus}$ is accepted by some multiset finite automaton then there is exactly one language $L \subseteq \Sigma^*$ such that $M = \Psi(L)$ and L is accepted by some jumping finite automaton.

Proof: Straightforward. (Both automata have identical state diagrams. The uniqueness follows from the fact that every language accepted by jumping finite automaton contains with every word also any permutation of its symbols.)

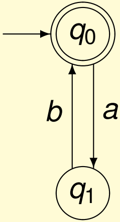
Similarities and dissimilarities of multiset languages with languages accepted by jumping finite automata

Consequence: $\mathcal{L}$(JFA) corresponds to $\mathcal{L}$(mFA) and vice versa.

Example:



Jumping finite automaton A



Multiset finite automaton B

Accepted languages

- $L(A) = \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$,
- $\Psi(L(A)) = \{(n, n) \mid n \in \mathbb{N}\} = M(B)$.

Similarities and dissimilarities of multiset languages with languages accepted by jumping finite automata

Consequence: $\mathcal{L}$(JFA) corresponds to $\mathcal{L}$(mFA) and vice versa.

Note: $\mathcal{L}$(mFA) is equal to the set of all semilinear sets.

Definition: A *semilinear set* M over $\mathbb{N}^n$ is defined by

$$M = \bigcup_{i=1}^{k} \{\mathbf{u}_{i,0} + l_{i,1}\mathbf{u}_{i,1} + \ldots + l_{i,m_i}\mathbf{u}_{i,m_i} \mid l_{i,1}, \ldots, l_{i,m_i} \in \mathbb{N}\}$$

where

$$k, m_1, \ldots, m_k \in \mathbb{N},\ \mathbf{u}_{1,0}, \ldots, \mathbf{u}_{1,m_1}, \ldots, \mathbf{u}_{k,0}, \ldots, \mathbf{u}_{k,m_k} \in \mathbb{N}^n.$$

Some closure properties

Operation	$\mathcal{L}$(JFA)	$\mathcal{L}$(mFA)
union	$+$	$+$
intersection	$+$	$+$
complement	$+$	$+$
concatenation	$-$	xxx
multiset addition	xxx	$+$
homomorphism	$-$	$+$
substitution	$-$	$+$

Similarities and dissimilarities of multiset languages with languages accepted by jumping finite automata

Definition: A *homomorphism on strings* is defined as a mapping $h: \ \Sigma^* \to \Delta^*$ such that

$h(\varepsilon) = \varepsilon$ and $h(u \cdot v) = h(u) \cdot h(v)$ for all $u, v \in \Sigma^*$.

Note that homomorphism "respects" concatenation.

Example: $h: \{0,1\}^* \to \{a,b\}^*$ where $h(0) = ab$ and $h(1) = ba$.
Then $h(011) = abbaba$.

Definition: A *homomorphism on multisets* is defined as a mapping $h: \ \Sigma^{\oplus} \to \Delta^{\oplus}$ such that

$h(\mathbf{0}_\Sigma) = \mathbf{0}_\Delta$ and $h(\alpha \oplus \beta) = h(\alpha) \oplus h(\beta)$ for all $\alpha, \beta \in \Sigma^{\oplus}$.

Similarities and dissimilarities of multiset languages with languages accepted by jumping finite automata

<u>Definition</u>: A *substitution on strings* is defined as a mapping $s: \ \Sigma^* \to 2^{\Delta^*}$ such that

$s(\varepsilon) = \{\varepsilon\}$ and $s(u \cdot v) = s(u) \cdot s(v)$ for all $u, v \in \Sigma^*$.

Note that substitution “respects” concatenation.

<u>Definition</u>: A *substitution on multisets* is defined as a mapping $s: \ \Sigma^{\oplus} \to 2^{\Delta^{\oplus}}$ such that

$s(\mathbf{0}_{\Sigma}) = \{\mathbf{0}_{\Delta}\}$ and $s(\alpha \oplus \beta) = s(\alpha) \oplus s(\beta)$ for all $\alpha, \beta \in \Sigma^{\oplus}$.

Minimization problem for multiset finite automata

Definition: A and B are equivalent iff $M(A) = M(B)$.

Minimization problem for multiset finite automata

Definition: A and B are equivalent iff $M(A) = M(B)$.

Definition: An mFA A is called *minimal* if there is no equivalent mFA B with smaller number of states.

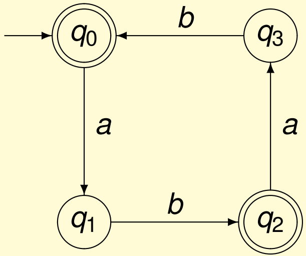
Minimization problem for multiset finite automata

Definition: A and B are equivalent iff $M(A) = M(B)$.

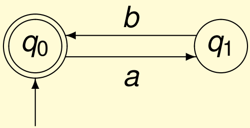
Definition: An mFA A is called *minimal* if there is no equivalent mFA B with smaller number of states.

Example:

A:



B:



The automaton B is minimal $(M(B) = \{\langle a\rangle^n \oplus \langle b\rangle^n \mid n \geq 0\})$.

Minimization problem for multiset finite automata

Problem of minimization: If an automaton of certain type is given, then we look for an equivalent minimal automaton of the same type.

Optimally, the minimal automaton is unique (up to isomorphism).

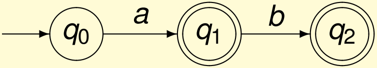
Note: the following parts are based on

- Martinek P., *Some notes to minimization of multiset finite automata*, in *International Conference of Numerical Analysis and Applied Mathematics (ICNAAM 2017)*, T. Simos and Ch. Tsitouras, Eds., AIP Conference proceedings 1978, 470019 (2018)
- Martinek P., *On a Generalized Form of Multiset Finite Automata with Suppressed Nonfinal States*, in *International Conference of Numerical Analysis and Applied Mathematics (ICNAAM 2020)*, to appear

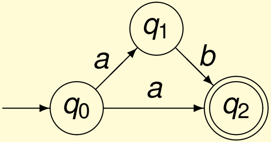
Minimization problem for multiset finite automata

An example of nonisomorphic minimal nondeterministic multiset finite automata:

A :



B :

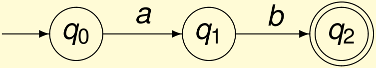


$$M(A) = M(B) = \{\langle a \rangle, \langle a \rangle \oplus \langle b \rangle\}$$

Minimization problem for multiset finite automata

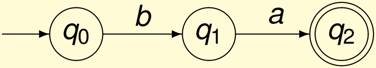
An example of nonisomorphic and isomorphic minimal deterministic multiset finite automata:

C :



$$M(C) = M(D) = \{\langle a \rangle \oplus \langle b \rangle\}$$

D :

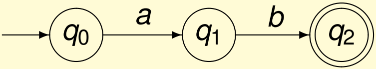


C and D are not isomorphic.

Minimization problem for multiset finite automata

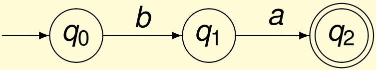
An example of nonisomorphic and isomorphic minimal deterministic multiset finite automata:

C :



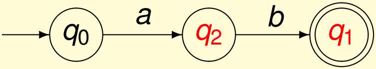
$$M(C) = M(D) = \{\langle a\rangle \oplus \langle b\rangle\}$$

D :



C and D are not isomorphic.

D' :



$$M(C) = M(D') = \{\langle a\rangle \oplus \langle b\rangle\}$$

C and D' are isomorphic.

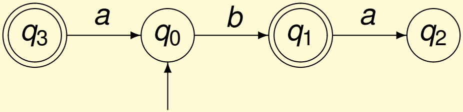
Minimization process for deterministic multiset finite automata

1. Removing unreachable states.

Minimization process for deterministic multiset finite automata

1. Removing unreachable states.

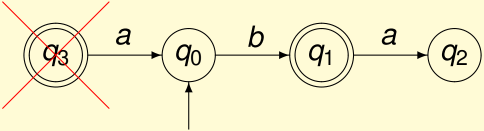
Example:



Minimization process for deterministic multiset finite automata

1. Removing unreachable states.

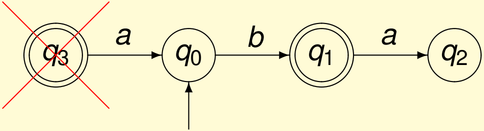
Example:



Minimization process for deterministic multiset finite automata

1. Removing unreachable states.
2. Removing nonterminating states.

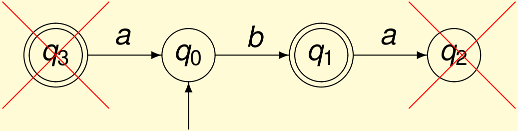
Example:



Minimization process for deterministic multiset finite automata

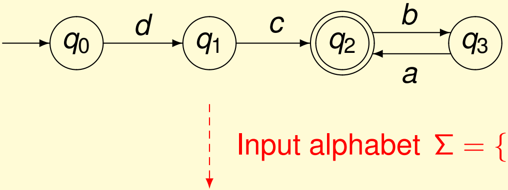
1. Removing unreachable states.
2. Removing nonterminating states.

<u>Example</u>:



Minimization process for deterministic multiset finite automata

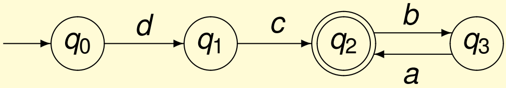
3. Lexicographic reordering of transitions:



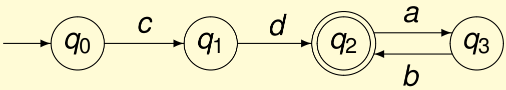
Input alphabet $\Sigma = \{a, b, c, d\}$

Minimization process for deterministic multiset finite automata

3. Lexicographic reordering of transitions:



Input alphabet $\Sigma = \{a, b, c, d\}$



Minimization process for deterministic multiset finite automata

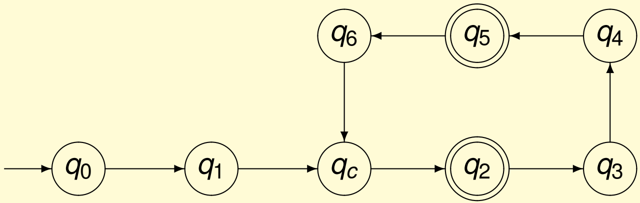
Key states for lexicographically ordered sequences of transitions at deterministic multiset finite automata:

- the initial state,

Minimization process for deterministic multiset finite automata

Key states for lexicographically ordered sequences of transitions at deterministic multiset finite automata:

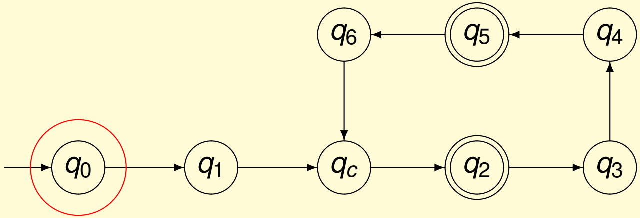
- the initial state,



Minimization process for deterministic multiset finite automata

Key states for lexicographically ordered sequences of transitions at deterministic multiset finite automata:

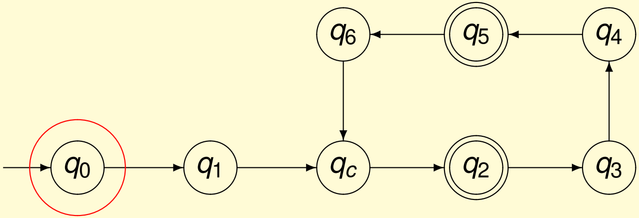
- the initial state,



Minimization process for deterministic multiset finite automata

Key states for lexicographically ordered sequences of transitions at deterministic multiset finite automata:

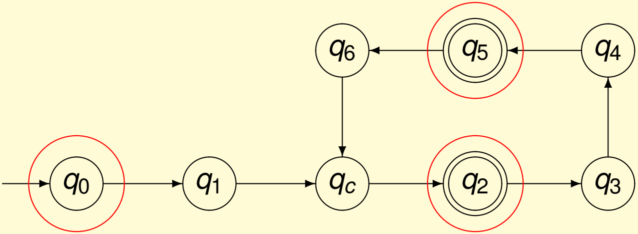
- the initial state,
- final states,



Minimization process for deterministic multiset finite automata

Key states for lexicographically ordered sequences of transitions at deterministic multiset finite automata:

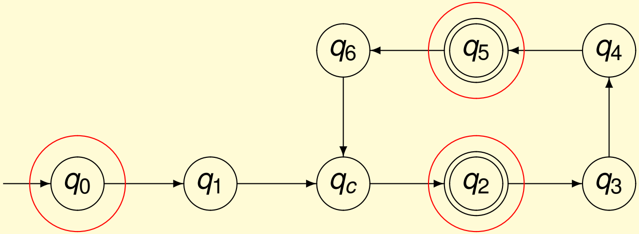
- the initial state,
- final states,



Minimization process for deterministic multiset finite automata

Key states for lexicographically ordered sequences of transitions at deterministic multiset finite automata:

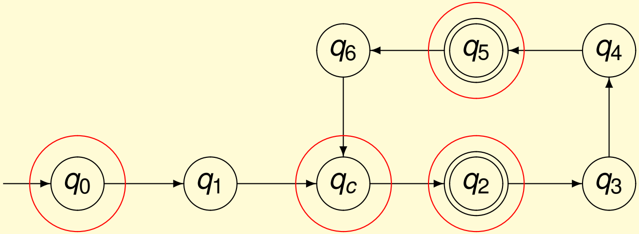
- the initial state,
- final states,
- the state q_c for which $(q, a, q_c) \in \delta$ and $(r, b, q_c) \in \delta$ with $q \neq r$.



Minimization process for deterministic multiset finite automata

Key states for lexicographically ordered sequences of transitions at deterministic multiset finite automata:

- the initial state,
- final states,
- the state q_c for which $(q, a, q_c) \in \delta$ and $(r, b, q_c) \in \delta$ with $q \neq r$.

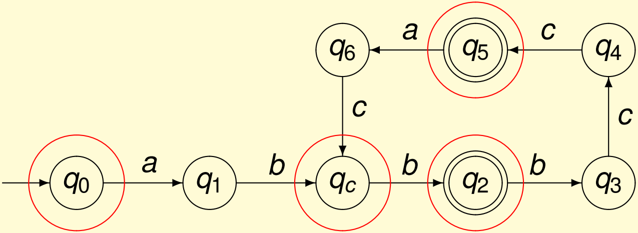


Minimization process for deterministic multiset finite automata

<u>Definition</u>: A dmFA $A = (Q, \Sigma, \delta, q_0, F)$ is said to be a *deterministic multiset finite automaton with lexicographically ordered transitions* if for any sequence of transitions $(q_i, a_i, q_{i+1})_{i=1}^{n}$ from δ with $n \geq 1$, the sequence $(a_i)_{i=1}^{n}$ is lexicographically ordered whenever $q_1, q_{n+1} \in F \cup \{q_0, q_c\}$ and $q_2, \ldots, q_n \notin F \cup \{q_0, q_c\}$.

Minimization process for deterministic multiset finite automata

<u>Definition</u>: A dmFA $A = (Q, \Sigma, \delta, q_0, F)$ is said to be a *deterministic multiset finite automaton with lexicographically ordered transitions* if for any sequence of transitions $(q_i, a_i, q_{i+1})_{i=1}^{n}$ from δ with $n \geq 1$, the sequence $(a_i)_{i=1}^{n}$ is lexicographically ordered whenever $q_1, q_{n+1} \in F \cup \{q_0, q_c\}$ and $q_2, \ldots, q_n \notin F \cup \{q_0, q_c\}$.



Minimization process for deterministic multiset finite automata

4. Merging indistinguishable states.

Minimization process for deterministic multiset finite automata

4. Merging indistinguishable states.

Definition: States $p, q \in Q$ of a dmFA $A = (Q, \Sigma, \delta, q_0, F)$ are called *distinguishable* iff there exists $\alpha \in \Sigma^{\oplus}$ satisfying either

a) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \in F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \notin F$

or

b) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \notin F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \in F$.

States, which are not distinguishable, are called *indistinguishable*.

Minimization process for deterministic multiset finite automata

4. Merging indistinguishable states.

<u>Definition</u>: States $p, q \in Q$ of a dmFA $A = (Q, \Sigma, \delta, q_0, F)$ are called *distinguishable* iff there exists $\alpha \in \Sigma^{\oplus}$ satisfying either

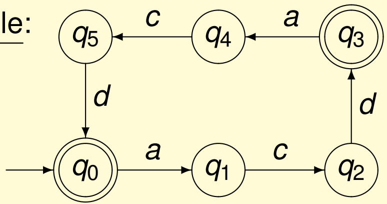
a) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \in F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \notin F$

or

b) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \notin F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \in F$.

States, which are not distinguishable, are called *indistinguishable*.

<u>Example</u>:



Minimization process for deterministic multiset finite automata

4. Merging indistinguishable states.

<u>Definition</u>: States $p, q \in Q$ of a dmFA $A = (Q, \Sigma, \delta, q_0, F)$ are called *distinguishable* iff there exists $\alpha \in \Sigma^{\oplus}$ satisfying either

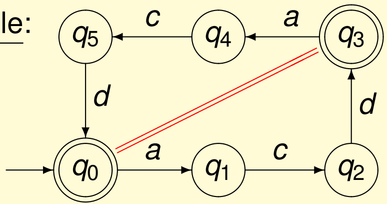
a) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \in F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \notin F$

or

b) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \notin F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \in F$.

States, which are not distinguishable, are called *indistinguishable*.

<u>Example</u>:



Minimization process for deterministic multiset finite automata

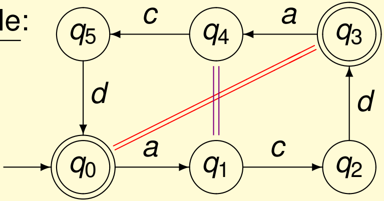
4. Merging indistinguishable states.

<u>Definition</u>: States $p, q \in Q$ of a dmFA $A = (Q, \Sigma, \delta, q_0, F)$ are called *distinguishable* iff there exists $\alpha \in \Sigma^{\oplus}$ satisfying either

a) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \in F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \notin F$
or
b) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \notin F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \in F$.

States, which are not distinguishable, are called *indistinguishable*.

<u>Example</u>:



Minimization process for deterministic multiset finite automata

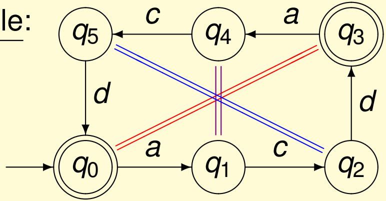
4. Merging indistinguishable states.

<u>Definition</u>: States $p, q \in Q$ of a dmFA $A = (Q, \Sigma, \delta, q_0, F)$ are called *distinguishable* iff there exists $\alpha \in \Sigma^{\oplus}$ satisfying either

a) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \in F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \notin F$
or
b) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \notin F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \in F$.

States, which are not distinguishable, are called *indistinguishable*.

<u>Example</u>:



Minimization process for deterministic multiset finite automata

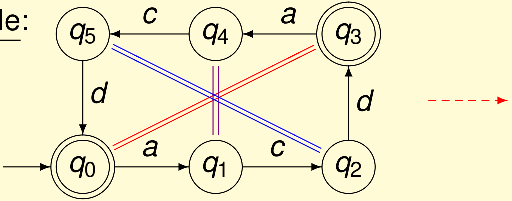
4. Merging indistinguishable states.

Definition: States $p, q \in Q$ of a dmFA $A = (Q, \Sigma, \delta, q_0, F)$ are called *distinguishable* iff there exists $\alpha \in \Sigma^{\oplus}$ satisfying either

a) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \in F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \notin F$
or
b) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \notin F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \in F$.

States, which are not distinguishable, are called *indistinguishable*.

Example:



Minimization process for deterministic multiset finite automata

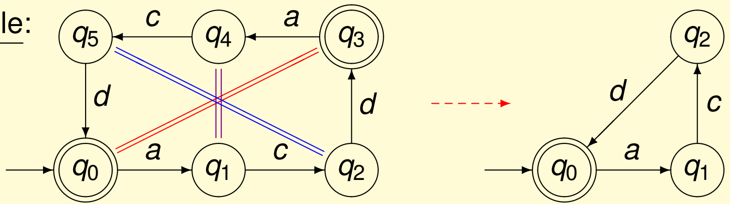
4. Merging indistinguishable states.

<u>Definition</u>: States $p, q \in Q$ of a dmFA $A = (Q, \Sigma, \delta, q_0, F)$ are called *distinguishable* iff there exists $\alpha \in \Sigma^{\oplus}$ satisfying either

a) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \in F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \notin F$
or
b) $(p, \alpha) \vdash^* (p', \mathbf{0}_\Sigma)$ with $p' \notin F$ and
$(q, \alpha) \vdash^* (q', \mathbf{0}_\Sigma)$ with $q' \in F$.

States, which are not distinguishable, are called *indistinguishable*.

<u>Example</u>:

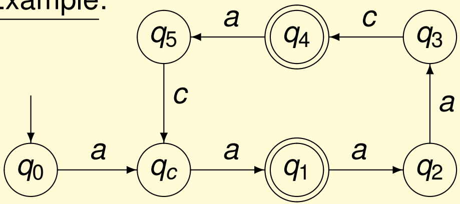


Minimization process for deterministic multiset finite automata

5. Solving situation around state q_c:
   a) shuffling transitions,
   b) merging indistinguishable states,
   c) final lexicographic reordering of transitions.

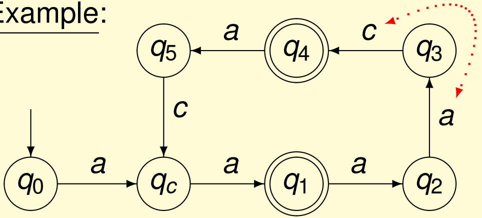
Minimization process for deterministic multiset finite automata

Example:



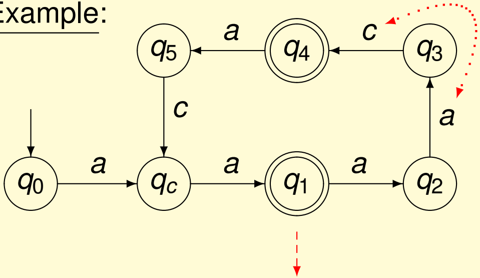
Minimization process for deterministic multiset finite automata

Example:



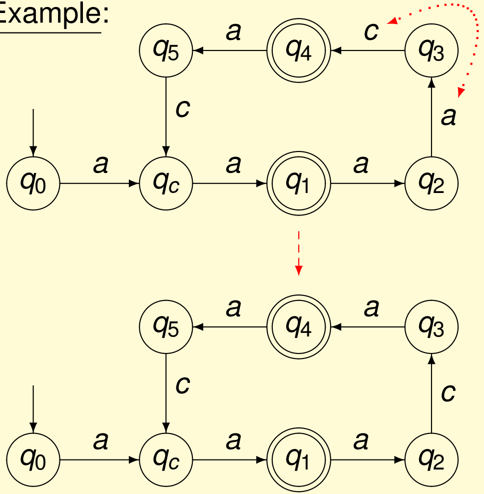
Minimization process for deterministic multiset finite automata

Example:



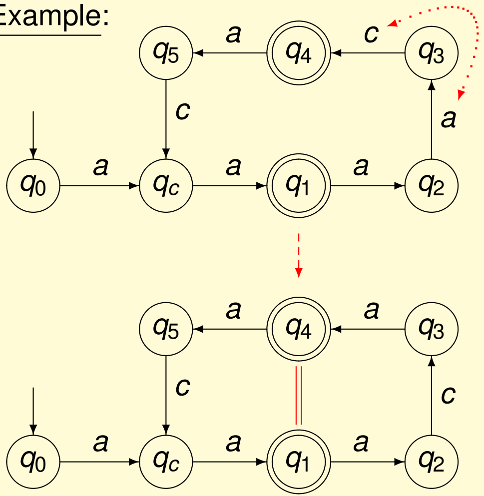
Minimization process for deterministic multiset finite automata

Example:



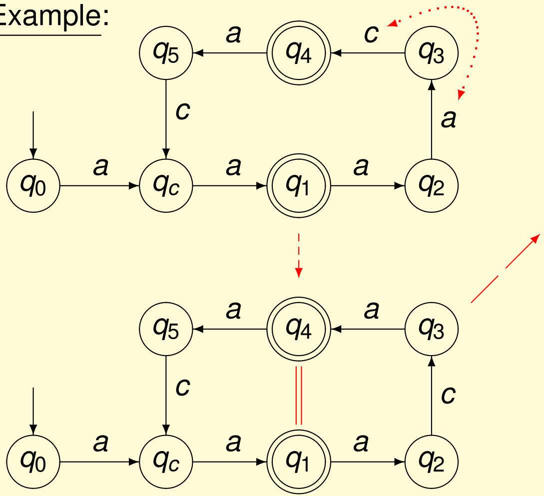
Minimization process for deterministic multiset finite automata

Example:



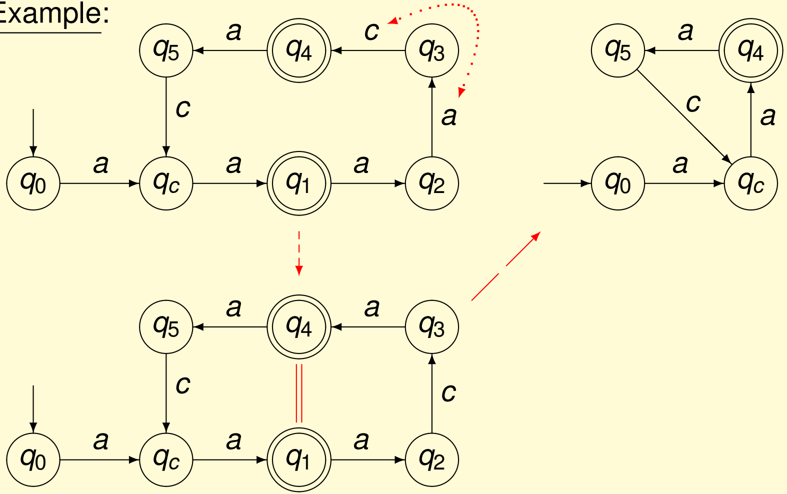
Minimization process for deterministic multiset finite automata

Example:



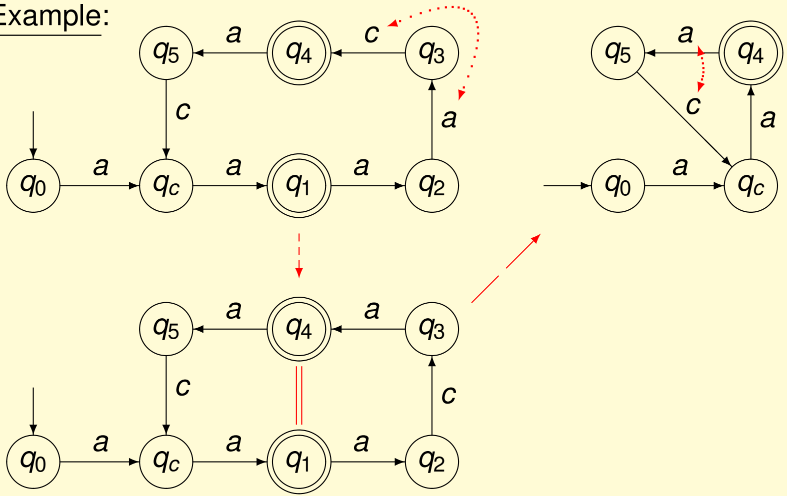
Minimization process for deterministic multiset finite automata

Example:



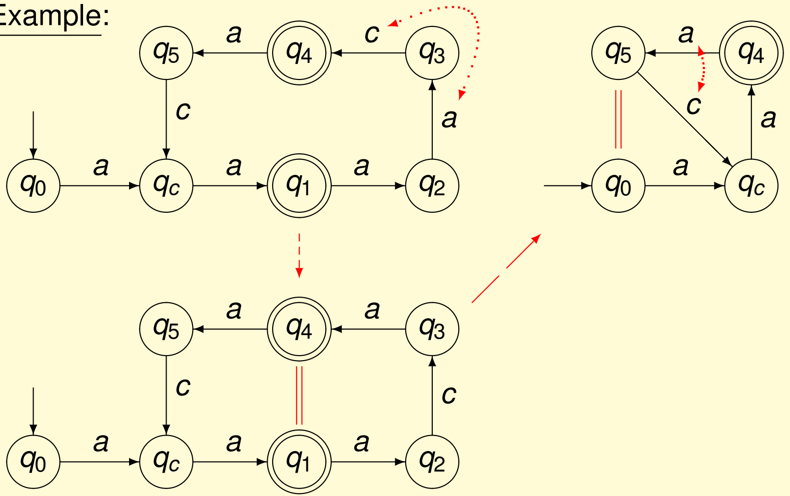
Minimization process for deterministic multiset finite automata

Example:



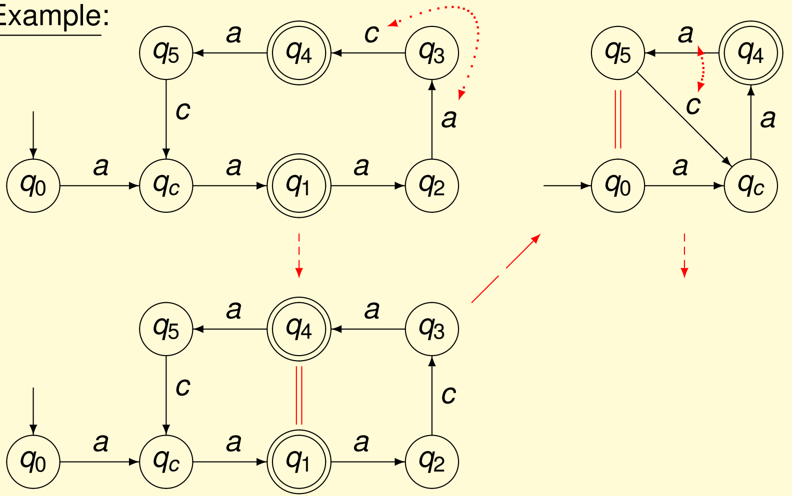
Minimization process for deterministic multiset finite automata

Example:



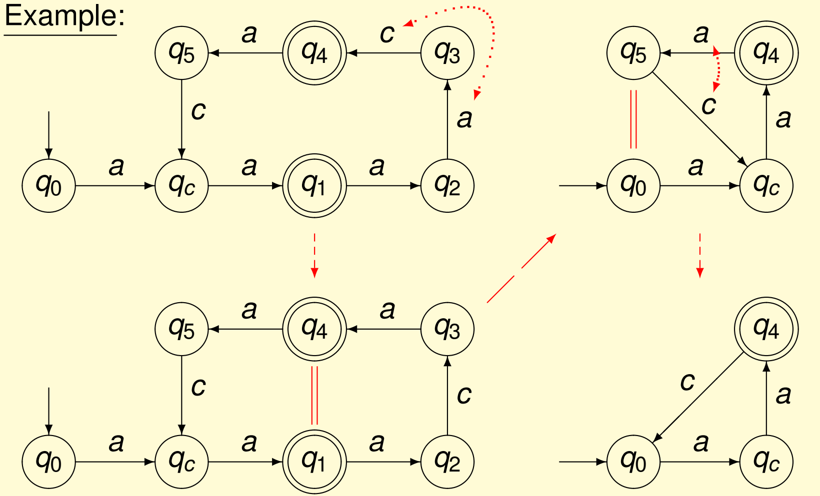
Minimization process for deterministic multiset finite automata

Example:



Minimization process for deterministic multiset finite automata

Example:



Minimization problem for multiset finite automata

Theorem: For every dmFA A, there is an equivalent minimal deterministic multiset finite automaton with lexicographically ordered transitions which is unique up to isomorphism.

Minimization problem for multiset finite automata

Theorem: For every dmFA A, there is an equivalent minimal deterministic multiset finite automaton with lexicographically ordered transitions which is unique up to isomorphism.

Remark: For some nondeterministic multiset finite automata, no unique equivalent minimal multiset finite automata exist.

Minimization problem for multiset finite automata in a generalized form

An unusual concept of a generalized multiset finite automaton with suppressed nonfinal states allows to grasp the minimization somewhat differently.

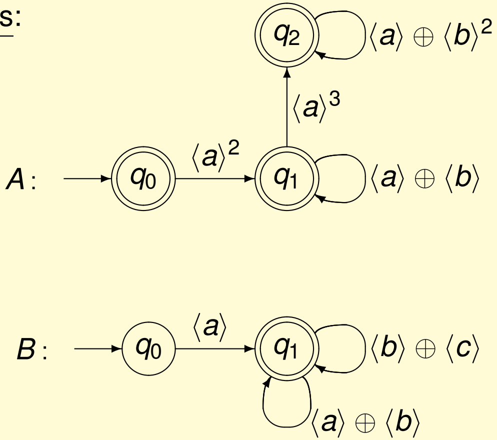
A *generalized multiset finite automaton with suppressed nonfinal states*: $A = (Q, \Sigma, \delta, q_0, F)$ where

- Q, Σ and q_0 are as in mFA,
- $F \subseteq Q$ is a set of final states such that $F \cup \{q_0\} = Q$,
- $\delta \subseteq Q \times \Sigma^{\oplus} \times Q$ is a finite transition relation satisfying the following condition.

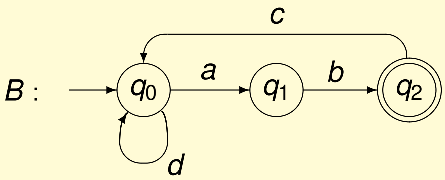
  If there are $q_1, \ldots, q_k \in Q$ such that $(q_{i-1}, \mathbf{0}_\Sigma, q_i) \in \delta$ for all $i \in \{1, \ldots, k\}$ and $q_k \in F$, then $q_0 \in F$.

Generalized multiset finite automata with suppressed nonfinal states

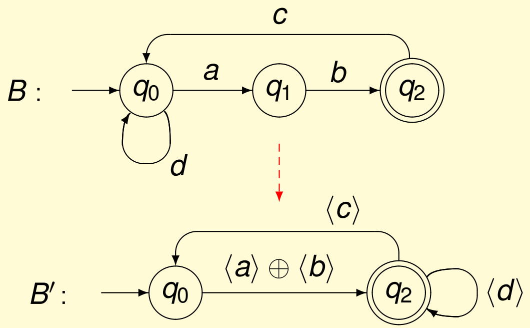
Examples:



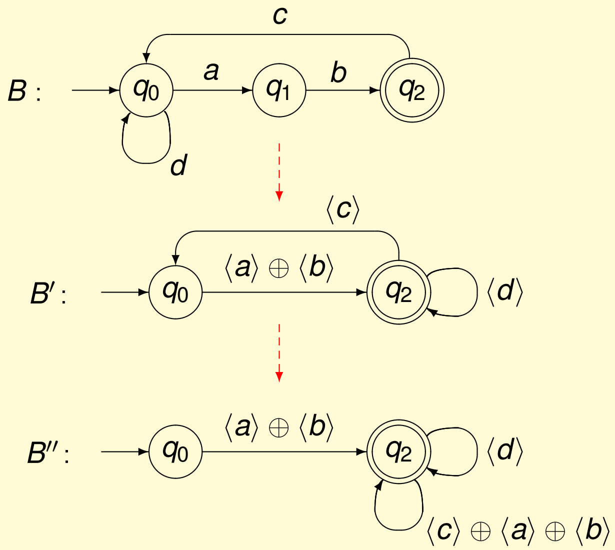
Idea of a transformation to a generalized multiset automaton with suppressed nonfinal states



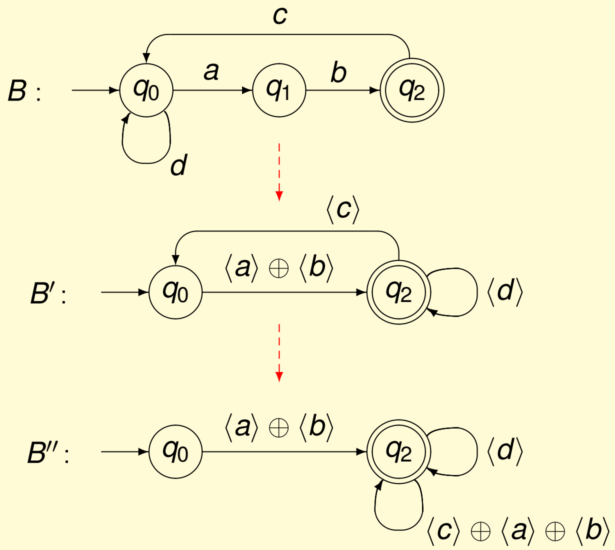
Idea of a transformation to a generalized multiset automaton with suppressed nonfinal states



Idea of a transformation to a generalized multiset automaton with suppressed nonfinal states



Idea of a transformation to a generalized multiset automaton with suppressed nonfinal states



$M(B) = M(B'')$

Generalized multiset finite automata with suppressed nonfinal states and their minimization

Theorem: Generalized multiset finite automata with suppressed nonfinal states accept the family of multiset languages accepted by multiset finite automata.

Generalized multiset finite automata with suppressed nonfinal states and their minimization

Theorem: Generalized multiset finite automata with suppressed nonfinal states accept the family of multiset languages accepted by multiset finite automata.

Theorem: For every deterministic multiset finite automaton, there is an equivalent minimal deterministic generalized multiset finite automaton with suppressed nonfinal states which is unique up to isomorphism.

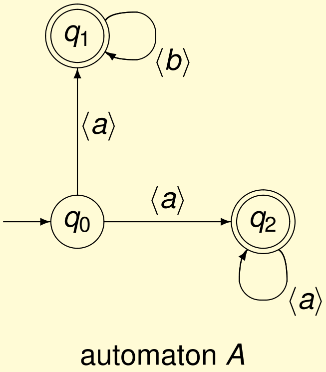
Generalized multiset finite automata with suppressed nonfinal states and their minimization

<u>Theorem</u>: Generalized multiset finite automata with suppressed nonfinal states accept the family of multiset languages accepted by multiset finite automata.

<u>Theorem</u>: For every deterministic multiset finite automaton, there is an equivalent minimal deterministic generalized multiset finite automaton with suppressed nonfinal states which is unique up to isomorphism.

<u>Note</u>: Generally, the previous theorem does not hold true for nondeterministic generalized multiset finite automata with suppressed nonfinal states.

An example of nonisomorphic minimal nondeterministic generalized multiset finite automata with suppressed nonfinal states



$$M(A) = M(B) = \{\langle a \rangle \oplus \langle b \rangle^m \mid m \geq 0\} \cup \{\langle a \rangle^n \mid n \geq 1\}$$

Conclusion

Multiset languages theory:

- Large field for further research
- Many unsolved problems

Thank you for your attention