Characterizing the Rational Functions by Restarting Transducers

N. Hundeshagen and F. Otto

Fachbereich Elektrotechnik/Informatik, Universität Kassel,
34109 Kassel, Germany

LATA 2012, A Coruña

Analysis by Reduction

- linguistic technique
- application on sentence of natural language with free word order (Czech, Russian, ...)
- task: verify correctness of a given sentence

- mode of operation: replace a small part of the sentence and start anew on the simplified sentence

  => restarting automata (Jančar, Mráz, Plátek, Vogel (1995 – 1999))

Analysis by Reduction

- linguistic technique
- application on sentence of natural language with free word order (Czech, Russian, ...)
- task: verify correctness of a given sentence

- mode of operation: replace a small part of the sentence and start anew on the simplified sentence

  => restarting automata (Jančar, Mráz, Plátek, Vogel (1995 – 1999))

Analysis by Reduction

- analysis of natural languages: several different representations of the analyzed sentence
- Question: How are the different representations related?

↷ analysis by reduction: correctness verification and information extracting

=> restarting transducer

Analysis by Reduction

- analysis of natural languages: several different representations of the analyzed sentence
- Question: How are the different representations related?

↷ analysis by reduction: correctness verification and information extracting

=> restarting transducer

Restarting Transducer

A restarting transducer (RR-Td for short) is formally described as a 7-tuple

$$T = (Q, \Sigma, \Delta, ¢, \$, q_0, \delta) :$$

- Q is the set of states
- Σ is the input-, Δ is the output alphabet
- $¢, \$$ are the endmarkers of the tape
- q_0 is the initial/restart state

- $\delta : Q \times (\Sigma \cup \{¢, \$\}) \to \mathcal{P}\big((Q \times \{\mathrm{MVR}, \varepsilon\}) \cup (\{\mathrm{Restart}, \mathrm{Accept}\} \times \Delta^*)\big)$

Relation computed by T:

$$\mathrm{Rel}(T) = \{(w, z) \in \Sigma^* \times \Delta^* \mid (q_0 ¢ w \$, \varepsilon) \vdash_T^* (\mathrm{Accept}, z)\}$$

Restarting Transducer

A restarting transducer (RR-Td for short) is formally described as a 7-tuple

$$T = (Q, \Sigma, \Delta, ¢, \$, q_0, \delta) :$$

- Q is the set of states
- Σ is the input-, Δ is the output alphabet
- $¢, \$$ are the endmarkers of the tape
- q_0 is the initial/restart state
- $\delta : Q \times (\Sigma \cup \{¢, \$\}) \to \mathcal{P}\big((Q \times \{\mathrm{MVR}, \varepsilon\}) \cup (\{\mathrm{Restart}, \mathrm{Accept}\} \times \Delta^*)\big)$

Relation computed by T:

$$\mathrm{Rel}(T) = \{(w, z) \in \Sigma^* \times \Delta^* \mid (q_0 ¢ w \$, \varepsilon) \vdash_T^* (\mathrm{Accept}, z)\}$$

Restarting Transducer

A restarting transducer (RR-Td for short) is formally described as a 7-tuple

$$T = (Q, \Sigma, \Delta, ¢, \$, q_0, \delta) :$$

- Q is the set of states
- Σ is the input-, Δ is the output alphabet
- $¢, \$$ are the endmarkers of the tape
- q_0 is the initial/restart state

- $\delta : Q \times (\Sigma \cup \{¢, \$\}) \to \mathcal{P}\big((Q \times \{\mathrm{MVR}, \varepsilon\}) \cup (\{\mathrm{Restart}, \mathrm{Accept}\} \times \Delta^*)\big)$

Relation computed by T:

$$\mathrm{Rel}(T) = \{(w, z) \in \Sigma^* \times \Delta^* \mid (q_0 ¢ w \$, \varepsilon) \vdash_T^* (\mathrm{Accept}, z)\}$$

Restarting Transducer

A restarting transducer (RR-Td for short) is formally described as a 7-tuple

$$T = (Q, \Sigma, \Delta, ¢, \$, q_0, \delta) :$$

- Q is the set of states
- Σ is the input-, Δ is the output alphabet
- $¢, \$$ are the endmarkers of the tape
- q_0 is the initial/restart state

- $\delta : Q \times (\Sigma \cup \{¢, \$\}) \to$
  $\mathcal{P}\big((Q \times \{\mathrm{MVR}, \varepsilon\}) \cup (\{\mathrm{Restart}, \mathrm{Accept}\} \times \Delta^*)\big)$

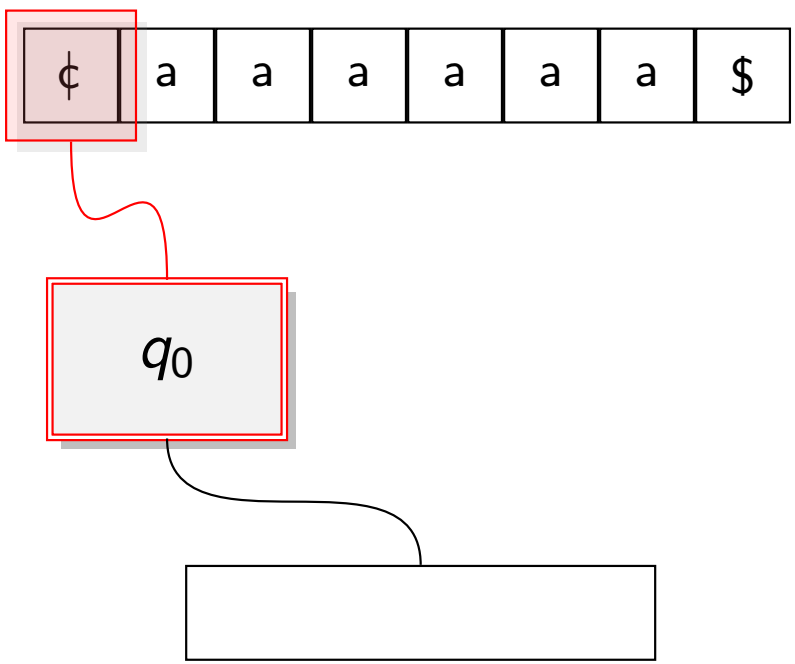
Relation computed by T:

$$\mathrm{Rel}(T) = \{(w, z) \in \Sigma^* \times \Delta^* \mid (q_0 ¢ w \$, \varepsilon) \vdash_T^* (\mathrm{Accept}, z)\}$$

Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

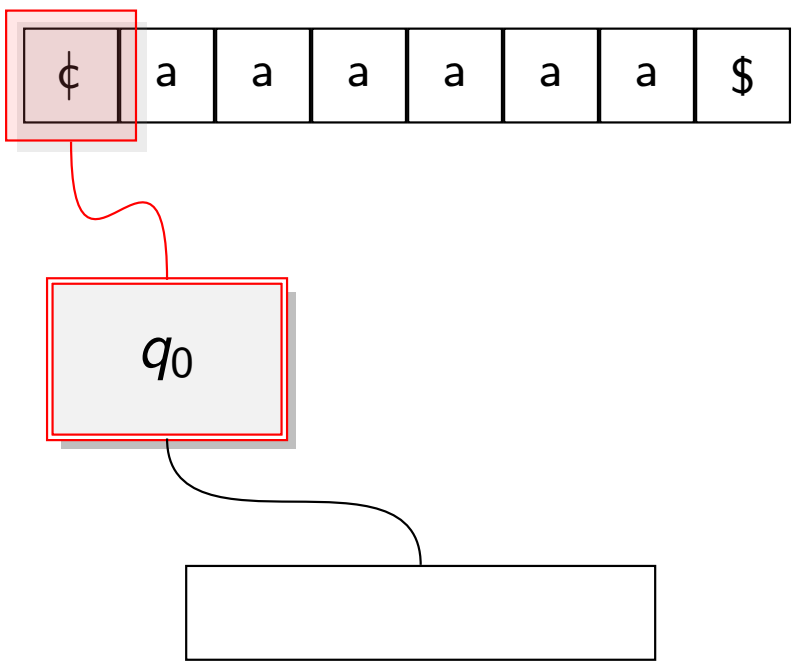
$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$



Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$

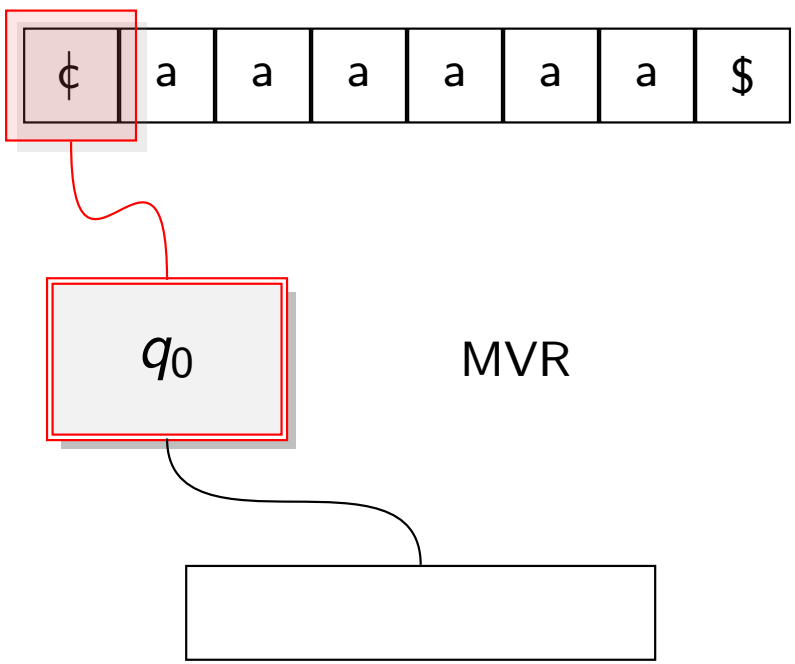


- the prefix det- denotes deterministic restarting transducer

Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

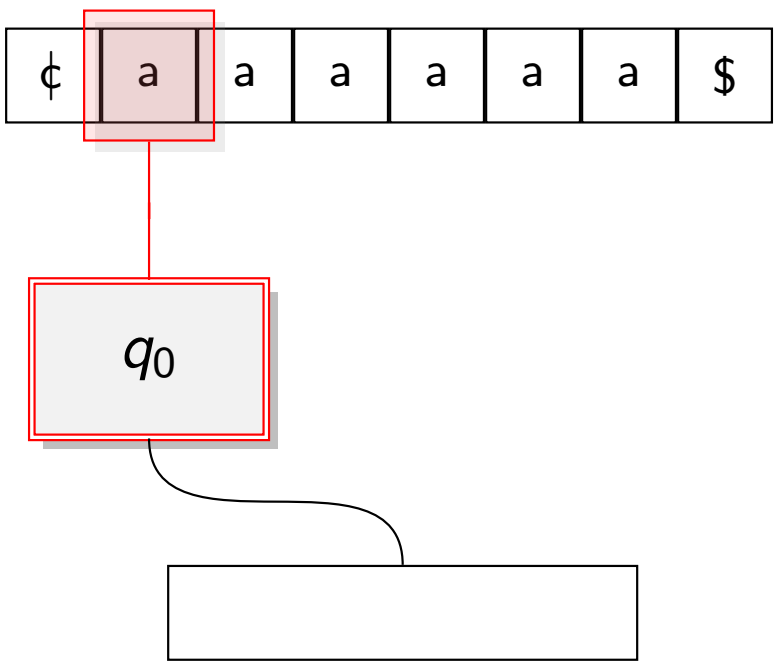
$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$



Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

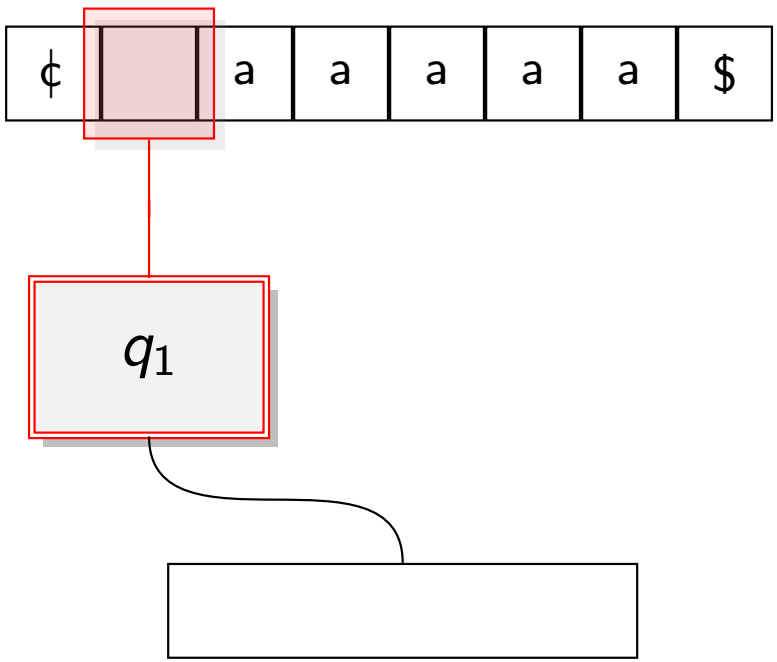
$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$



Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

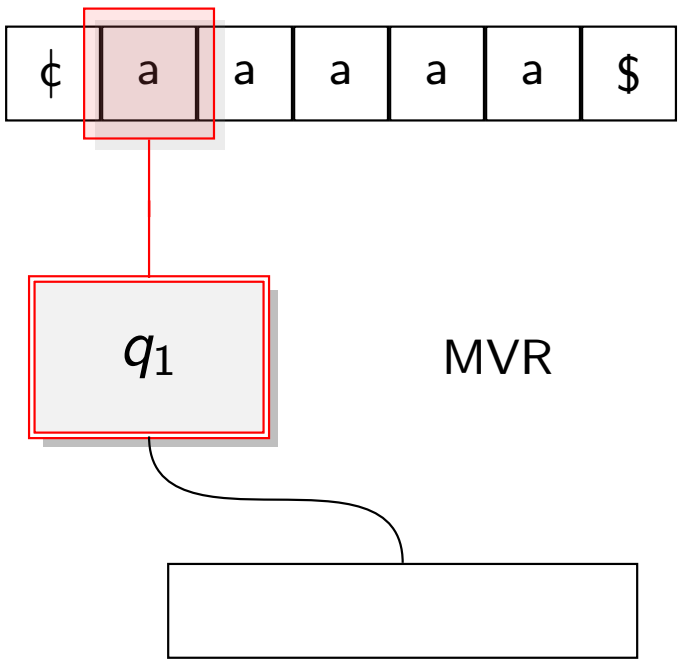
$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$



Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

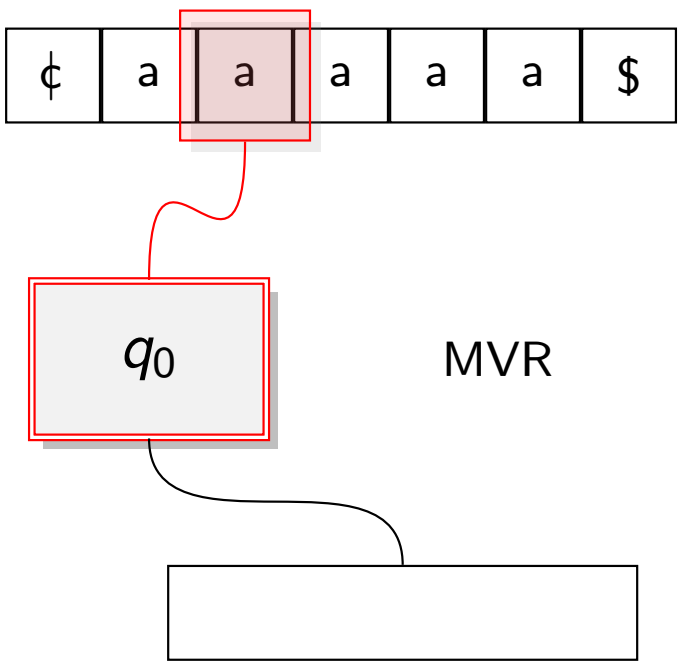
$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$



Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

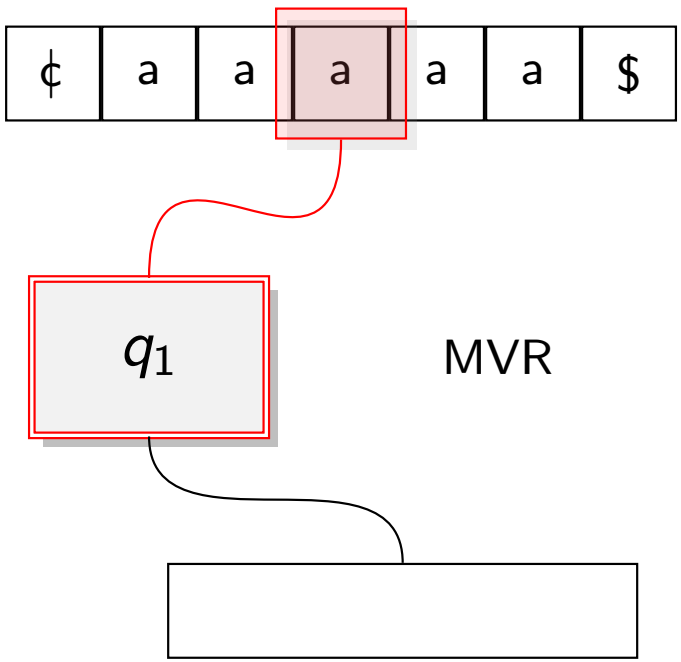
$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$



Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

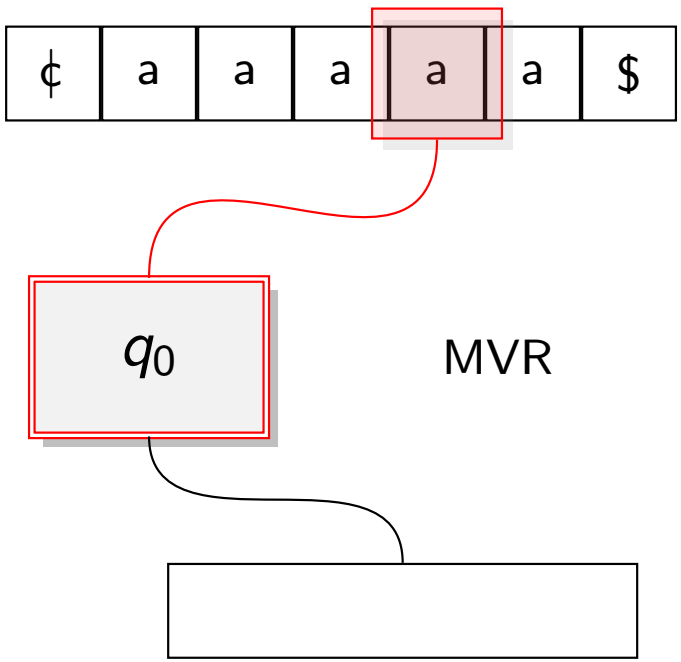
$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$



Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

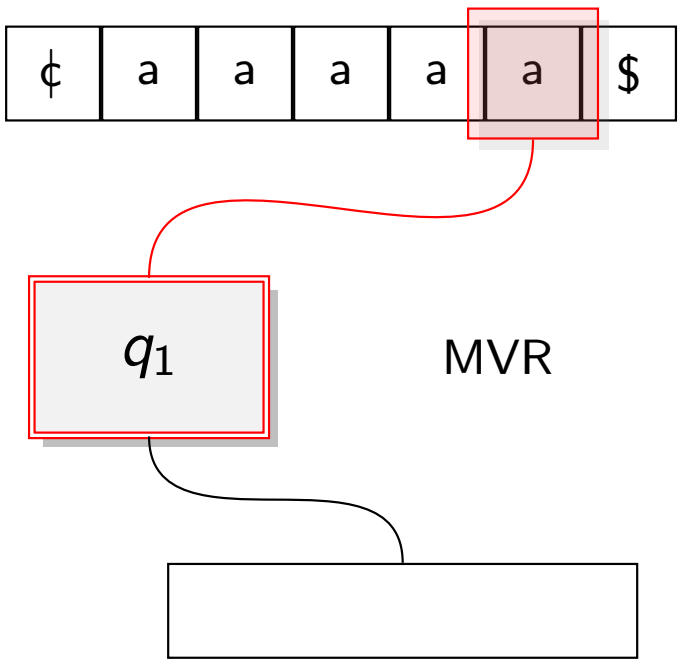
$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$



Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

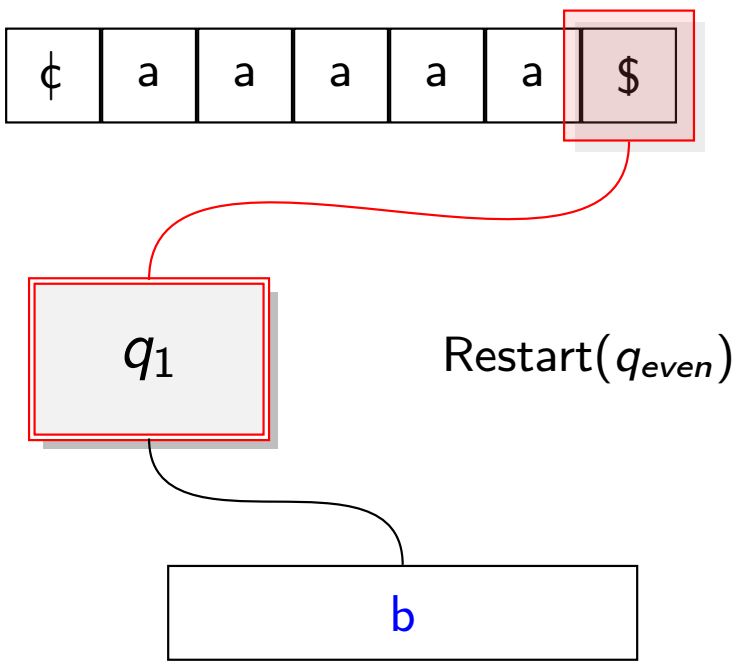
$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$



Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$

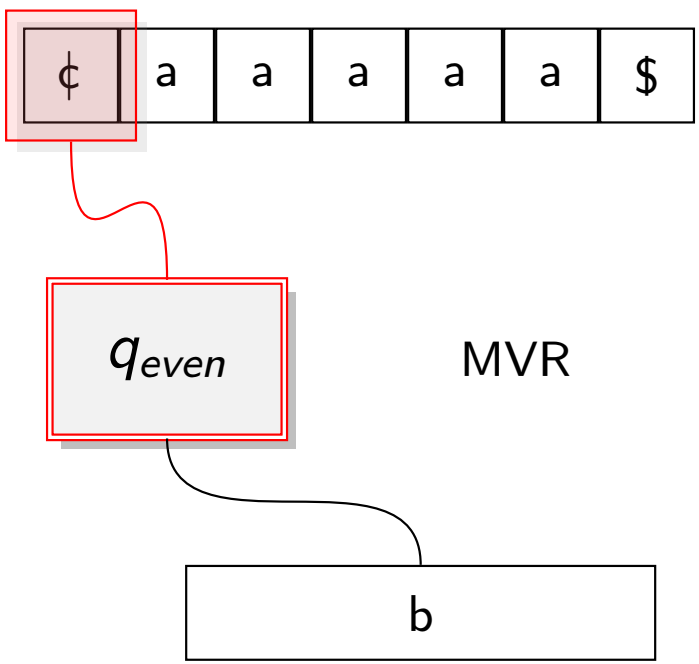


- nf- (nonforgetting): a restart operation is combined with a change of state as any other operation

Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

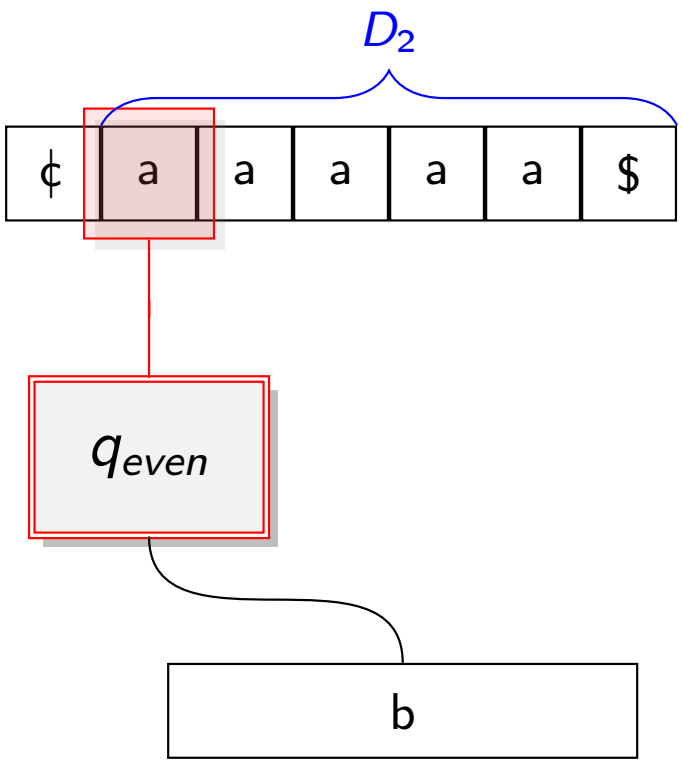
$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$



Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$

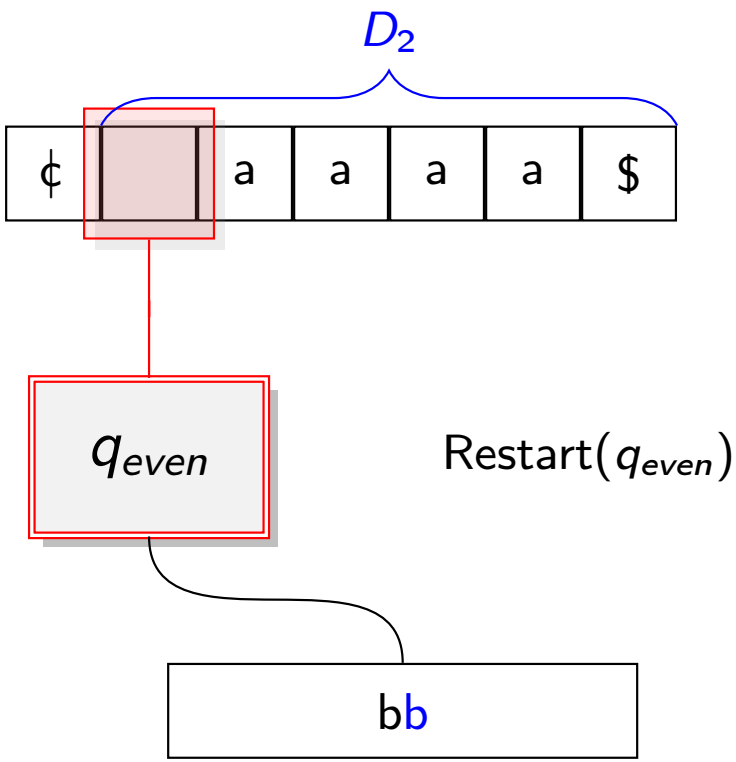


- mon- (monotone): in each computation $D_1 \geq D_2 \geq \ldots \geq D_k$

Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$

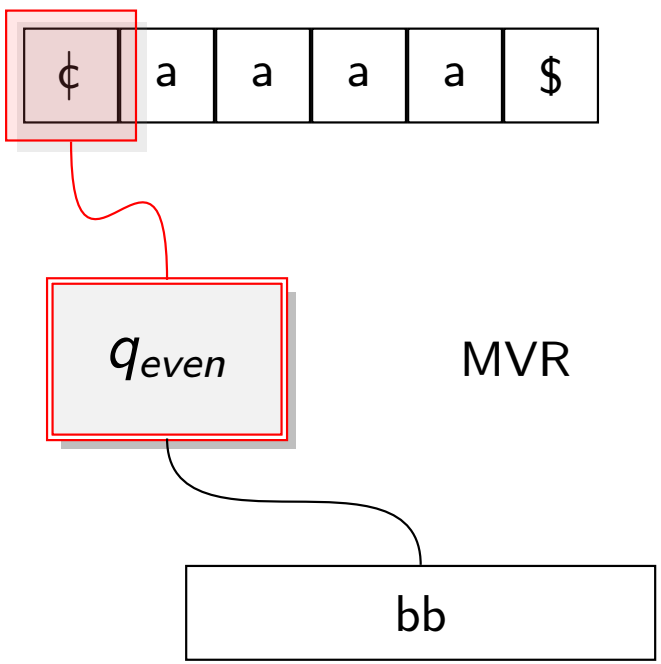


- mon- (monotone): in each computation $D_1 \geq D_2 \geq \ldots \geq D_k$

Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

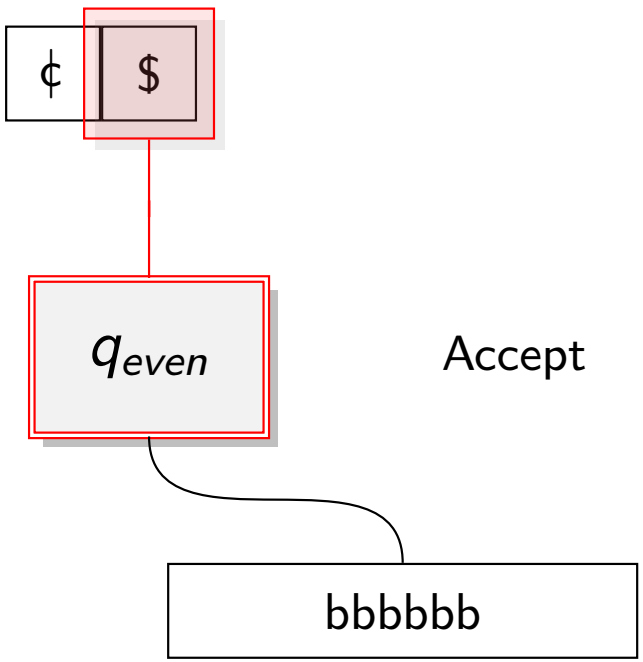
$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$



Example

A det-mon-nf-RR-transducer $M = (Q, \{a\}, \{b, c\}, ¢, \$, q_0, \delta)$ that computes the transduction

$$\tau(a^n) = \begin{cases} b^n & , n \text{ even}, \\ c^n & , n \text{ odd}. \end{cases}$$



Restarting Automata with Window Size One and Regular Languages

Known Results

Proposition (Mráz 2001,Reimann 2007)

$\mathcal{L}(\text{R}) = \mathcal{L}(\text{mon-R}) = \mathcal{L}(\text{det-R}) = \mathcal{L}(\text{det-RR}) = \text{REG}$

Proposition (2011)

$\mathcal{L}(\text{det-mon-nf-R}) = \mathcal{L}(\text{mon-nf-R}) = \mathcal{L}(\text{det-mon-nf-RR}) = \text{REG}$

- an R-automaton is an RR-automaton that has to restart immediately after deleting a symbol

Restarting Automata with Window Size One and Regular Languages

Known Results

Proposition (Mráz 2001,Reimann 2007)

$\mathcal{L}(\text{R}) = \mathcal{L}(\text{mon-R}) = \mathcal{L}(\text{det-R}) = \mathcal{L}(\text{det-RR}) = \text{REG}$

Proposition (2011)

$\mathcal{L}(\text{det-mon-nf-R}) = \mathcal{L}(\text{mon-nf-R}) = \mathcal{L}(\text{det-mon-nf-RR}) = \text{REG}$

- an R-automaton is an RR-automaton that has to restart immediately after deleting a symbol

Upper Bounds

Theorem (Nivat 1968)

- Pr^{Σ}, Pr^{Δ} *are the projections onto* Σ^* *and* Δ^*

  $R \in \mathrm{RatRel}(\Sigma^* \times \Delta^*)$ *iff*
  $\exists L \in \mathrm{REG}(\Sigma \cup \Delta) : \; R = \{(\mathrm{Pr}^{\Sigma}(w), \mathrm{Pr}^{\Delta}(w)) \mid w \in L\}$

Lemma

$\mathcal{R}el(\text{det-mon-nf-RR-Td}) \subseteq \mathrm{RatF} \quad | \quad \mathcal{R}el(\text{mon-nf-R-Td}) \subsetneq \mathrm{RatRel}$

Upper Bounds

Theorem (Nivat 1968)

- Pr^{Σ}, Pr^{Δ} *are the projections onto* Σ^* *and* Δ^*

  $R \in \mathrm{RatRel}(\Sigma^* \times \Delta^*)$ *iff*
  $\exists L \in \mathrm{REG}(\Sigma \cup \Delta) : \; R = \{(\mathrm{Pr}^{\Sigma}(w), \mathrm{Pr}^{\Delta}(w)) \mid w \in L\}$

Lemma

$\mathcal{R}el(\text{det-mon-nf-RR-Td}) \subseteq \mathrm{RatF} \quad | \quad \mathcal{R}el(\text{mon-nf-R-Td}) \subsetneq \mathrm{RatRel}$

First Results

DgsmF : deterministic sequential transducers
SubSeqF : subsequential transducers (ability to guess the end of the input word)

Corollary (Choffrut 1978)

A function $f : \Sigma^*\$ \mapsto \Delta^*$ ($\$ \notin \Sigma$) is a subsequential function iff f is a dgsm mapping.

Theorem

SubSeqF $= \mathcal{R}el$(det-mon-nf-R-Td)

Proof.

$\Rightarrow$ computing the subsequential function $f' : \Sigma^* \mapsto \Delta^*$ by simulating a dgsm for $f : \Sigma^*\$ \mapsto \Delta^*$

$\Leftarrow$ constructing a dgsm that computes the subsequential function $f : \Sigma^*\$ \mapsto \Delta^*$

First Results

DgsmF : deterministic sequential transducers
SubSeqF : subsequential transducers (ability to guess the end of the input word)

Corollary (Choffrut 1978)

A function $f : \Sigma^*\$ \mapsto \Delta^*$ ($\$ \notin \Sigma$) is a subsequential function iff f is a dgsm mapping.

Theorem

SubSeqF $= \mathcal{R}el$(det-mon-nf-R-Td)

Proof.

$\Rightarrow$ computing the subsequential function $f' : \Sigma^* \mapsto \Delta^*$ by simulating a dgsm for $f : \Sigma^*\$ \mapsto \Delta^*$

$\Leftarrow$ constructing a dgsm that computes the subsequential function $f : \Sigma^*\$ \mapsto \Delta^*$

First Results

DgsmF : deterministic sequential transducers
SubSeqF : subsequential transducers (ability to guess the end of the input word)

Corollary (Choffrut 1978)

A function $f : \Sigma^*\$ \mapsto \Delta^*$ ($\$ \notin \Sigma$) is a subsequential function iff f is a dgsm mapping.

Theorem

SubSeqF $= \mathcal{R}el$(det-mon-nf-R-Td)

Proof.

$\Rightarrow$ computing the subsequential function $f' : \Sigma^* \mapsto \Delta^*$ by simulating a dgsm for $f : \Sigma^*\$ \mapsto \Delta^*$

$\Leftarrow$ constructing a dgsm that computes the subsequential function $f : \Sigma^*\$ \mapsto \Delta^*$

First Results

DgsmF : deterministic sequential transducers
SubSeqF : subsequential transducers (ability to guess the end of the input word)

Corollary (Choffrut 1978)

A function $f : \Sigma^*\$ \mapsto \Delta^*$ ($\$ \notin \Sigma$) is a subsequential function iff f is a dgsm mapping.

Theorem

SubSeqF $= \mathcal{R}el$(det-mon-nf-R-Td)

Proof.

- $\Rightarrow$ computing the subsequential function $f' : \Sigma^* \mapsto \Delta^*$ by simulating a dgsm for $f : \Sigma^*\$ \mapsto \Delta^*$
- $\Leftarrow$ constructing a dgsm that computes the subsequential function $f : \Sigma^*\$ \mapsto \Delta^*$

First Results

Definition

A restarting transducer is called proper if no output is produced during a accept operation.

Corollary

DgsmF $= \mathcal{R}el$(prop-det-mon-nf-R-Td)

Characterizing the Rational Functions

RatF :
- unambiguous finite state transducer
- bimachine

Theorem
RatF $= \mathcal{R}el$(det-mon-nf-RR-Td).

Characterizing the Rational Functions

Proof.

- [Santean 2004] If $f : \Sigma^* \to \Delta^*$ is a rational function such that $f(\varepsilon) = \varepsilon$, then there exists a sequential function $f_L : (\Sigma \cup \{\$\})^* \to \Delta^*$ such that

$$f = f_L \circ \mu_\$.$$

- Here $\mu_\$: \Sigma^* \to (\Sigma \cup \{\$\})^*$ is a function defined by $\mu_\$(\varepsilon) = \varepsilon$ and

$$\mu_\$(a_1 \dots a_k) = a_1 \dots a_k \$ a_2 \dots a_k \$ \dots \$ a_{k-1} a_k \$ a_k.$$

$(k \geq 1,\ a_1, \dots, a_k \in \Sigma)$

Characterizing the Rational Functions

Proof.

- [Santean 2004] If $f : \Sigma^* \to \Delta^*$ is a rational function such that $f(\varepsilon) = \varepsilon$, then there exists a sequential function $f_L : (\Sigma \cup \{\$\})^* \to \Delta^*$ such that

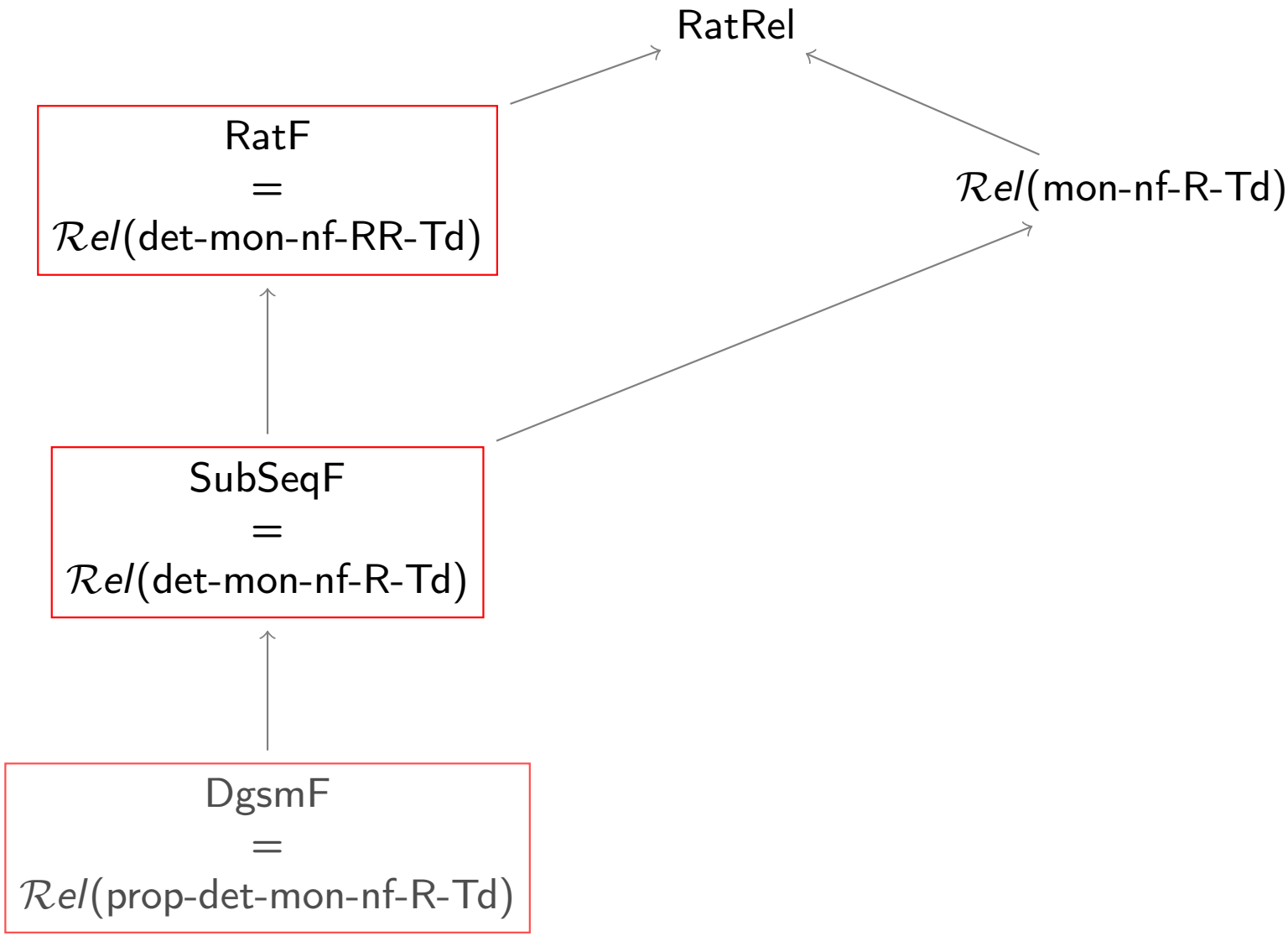
$$f = f_L \circ \mu_\$.$$

- Here $\mu_\$: \Sigma^* \to (\Sigma \cup \{\$\})^*$ is a function defined by $\mu_\$(\varepsilon) = \varepsilon$ and

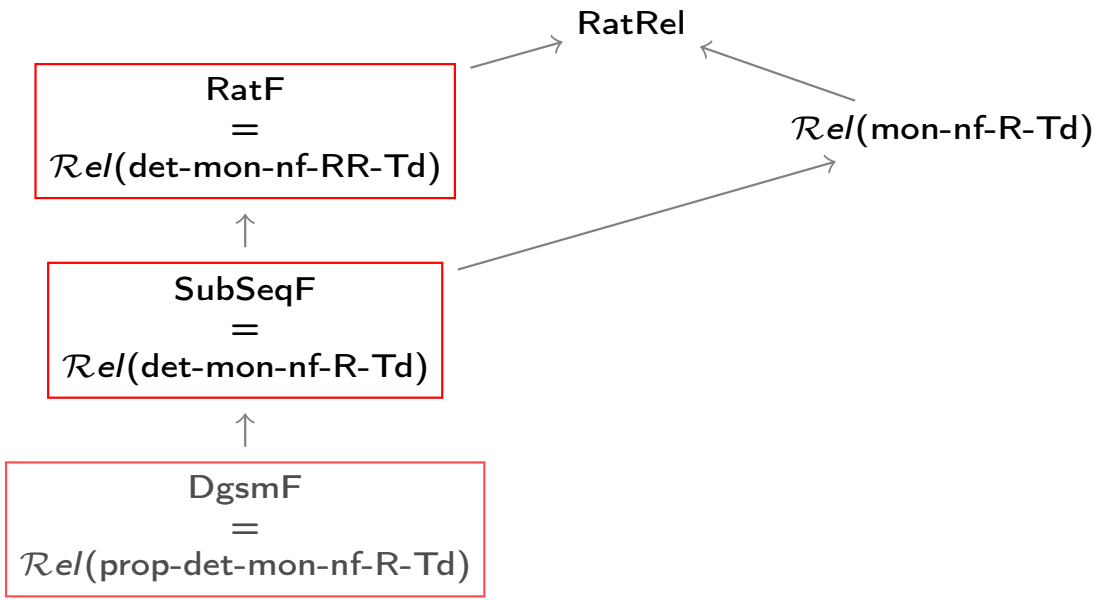
$$\mu_\$(a_1 \dots a_k) = a_1 \dots a_k \$ a_2 \dots a_k \$ \dots \$ a_{k-1} a_k \$ a_k.$$

$(k \geq 1,\ a_1, \dots, a_k \in \Sigma)$

Additional Observations, Summary and Outlook

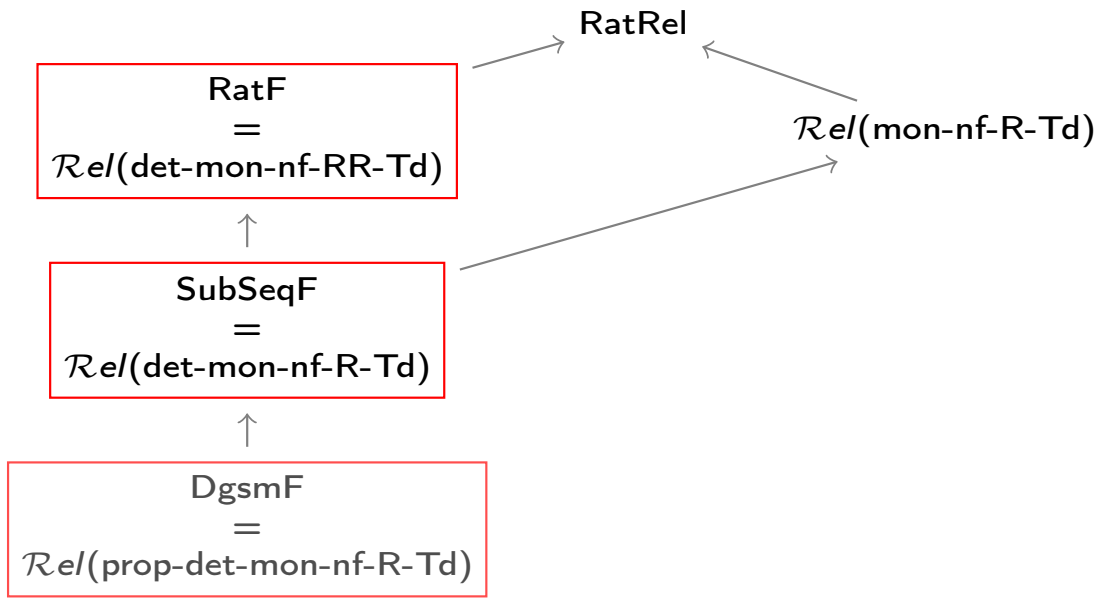


Additional Observations, Summary and Outlook



- numerous other transducer-classes within RatRel
- $\mathcal{R}el$(det-R-Td), $\mathcal{R}el$(mon-R-Td), $\mathcal{R}el$(det-RR-Td) can be separated

Additional Observations, Summary and Outlook



Open Questions:

- benefit of using restarting transducer instead of FST
- minimization algorithm
- general restarting transducer

Thank you for your attention!