Hyper-minimization for deterministic tree automata

Artur Jeż[1] and Andreas Maletti[2]

[1] University of Wrocław, Poland
[2] University of Stuttgart, Germany

Porto, Portugal — July 17, 2012

Contents

Hyper-Minimization

Intuition

Minimize automaton allowing a finite number of errors

Hyper-Minimization

Intuition

Minimize automaton allowing a finite number of errors

model/process	hyper-minimization	hyper-optimization
DFA	$\mathcal{O}(m \log n)$	$\mathcal{O}(mn)$
DBA	$\mathcal{O}(mn)$	???
DCA	$\mathcal{O}(mn)$	???
DTA	$\mathcal{O}(\ell mn)$	???

DTA = deterministic tree automaton

DBA / DCA = deterministic BÜCHI / Co-BÜCHI automaton

Why Tree Automata?

Applications

- Parsing (tree substitution grammars with latent variables)
- XML processing
- . . .

Parsing

- requires huge tree automata
- typically non-deterministic (but can be determinized)
- obtained from corpora (linguistic resources)

Why Tree Automata?

Applications

- Parsing (tree substitution grammars with latent variables)
- XML processing
- ...

Parsing

- requires huge tree automata
- typically non-deterministic (but can be determinized)
- obtained from corpora (linguistic resources)

Parsing of Natural Languages

Standard approach

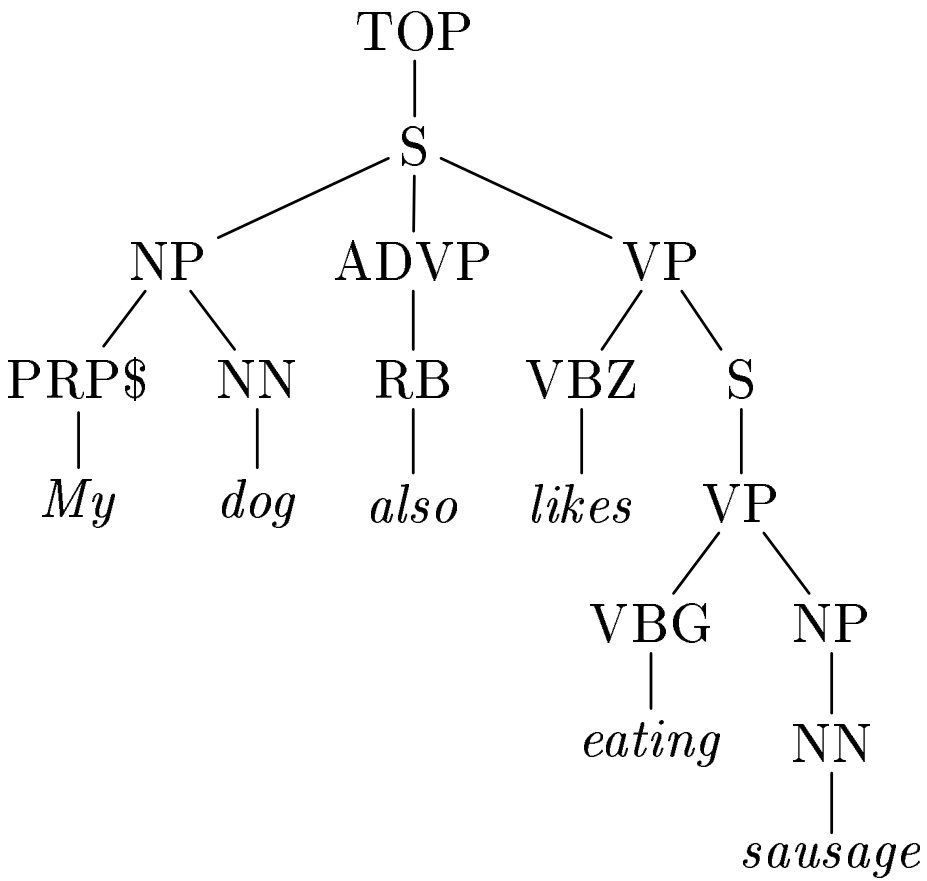
- supervised training (i.e., from annotated examples)
- but annotated samples are small, expensive
- can contain errors or inconsistencies

Intuition

- annotated sample not complete (positives only)
- annotation errors and inconsistencies (errors in annotation)

$\rightarrow$ reasonable training must generalize from finite observations to infinite language

Parse Tree


TOP
S
NP
ADVP
VP
PRP$
NN
RB
VBZ
S
My
dog
also
likes
VP
VBG
NP
eating
NN
sausage

Why Hyper-Minimization?

Advantages

- makes the DTA smaller $\rightarrow$ efficiency gain
- reduces spurious, artificial effects
- allows inspection of the “core” (the recursive structure)

Disadvantages

- gains sometimes rather small (in particular for DTA)
- no discrimination between errors
- no non-trivial limit on the number of errors

Why Hyper-Minimization?

Advantages

- makes the DTA smaller $\rightarrow$ efficiency gain
- reduces spurious, artificial effects
- allows inspection of the “core” (the recursive structure)

Disadvantages

- gains sometimes rather small (in particular for DTA)
- no discrimination between errors
- no non-trivial limit on the number of errors

Contents

Deterministic Tree Automaton

Definition (GÉCSEG, STEINBY 1984)

(Q, Σ, δ, F) deterministic tree automaton (DTA)

- Q finite set — *states*
- Σ ranked alphabet — *input symbols*
- $\delta\colon \Sigma(Q) \to Q$ — *transitions*
- $F \subseteq Q$ — *final states*

Definition

transition function extends to $\delta\colon T_\Sigma(Q) \to Q$ by

$$\delta(q) = q$$
$$\delta(\sigma(t_1, \ldots, t_k)) = \delta(\sigma(\delta(t_1), \ldots, \delta(t_k)))$$

Deterministic Tree Automaton

Definition (GÉCSEG, STEINBY 1984)

(Q, Σ, δ, F) deterministic tree automaton (DTA)

- Q finite set — *states*
- Σ ranked alphabet — *input symbols*
- $\delta\colon \Sigma(Q) \to Q$ — *transitions*
- $F \subseteq Q$ — *final states*

Definition

transition function extends to $\delta\colon T_\Sigma(Q) \to Q$ by

$$\delta(q) = q$$
$$\delta(\sigma(t_1, \ldots, t_k)) = \delta(\sigma(\delta(t_1), \ldots, \delta(t_k)))$$

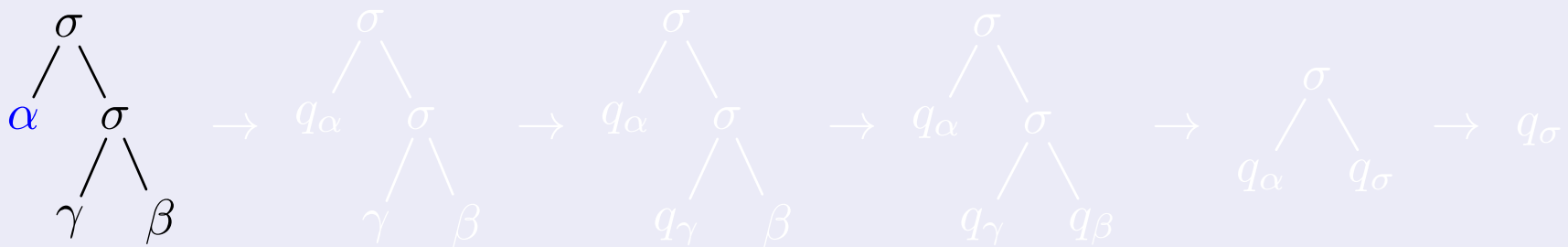
Deterministic Tree Automaton

Example

- states q_α, q_β (nonfinal) and q_γ, q_σ (final)
- nullary input symbols α, β, γ and binary σ
- for all nullary symbols π, π'

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$

Evaluating a tree



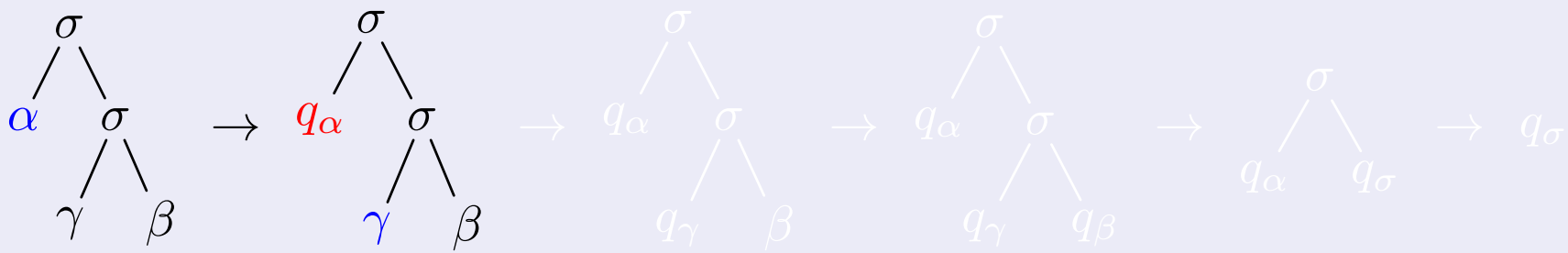
Deterministic Tree Automaton

Example

- states q_α, q_β (nonfinal) and q_γ, q_σ (final)
- nullary input symbols α, β, γ and binary σ
- for all nullary symbols π, π'

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$

Evaluating a tree



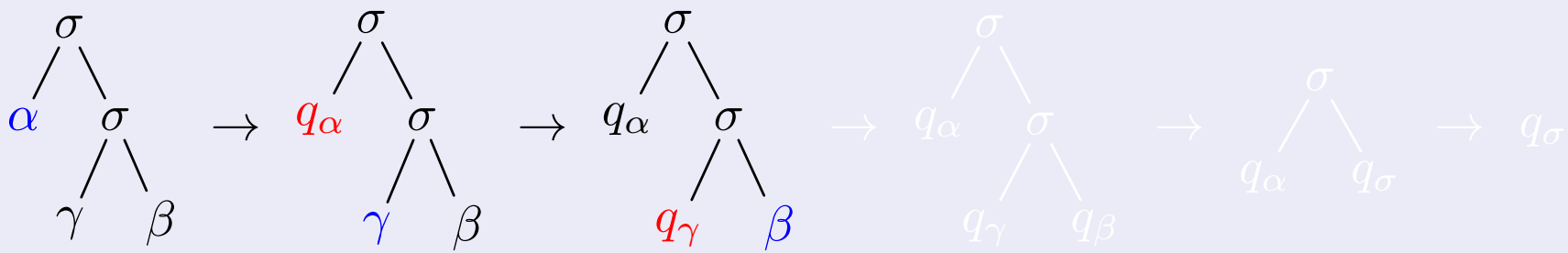
Deterministic Tree Automaton

Example

- states q_α, q_β (nonfinal) and q_γ, q_σ (final)
- nullary input symbols α, β, γ and binary σ
- for all nullary symbols π, π'

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$

Evaluating a tree



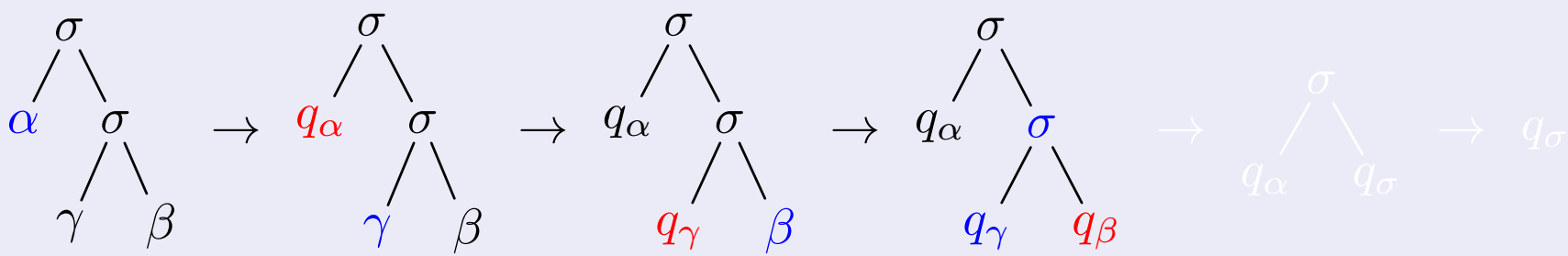
Deterministic Tree Automaton

Example

- states q_α, q_β (nonfinal) and q_γ, q_σ (final)
- nullary input symbols α, β, γ and binary σ
- for all nullary symbols π, π'

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$

Evaluating a tree



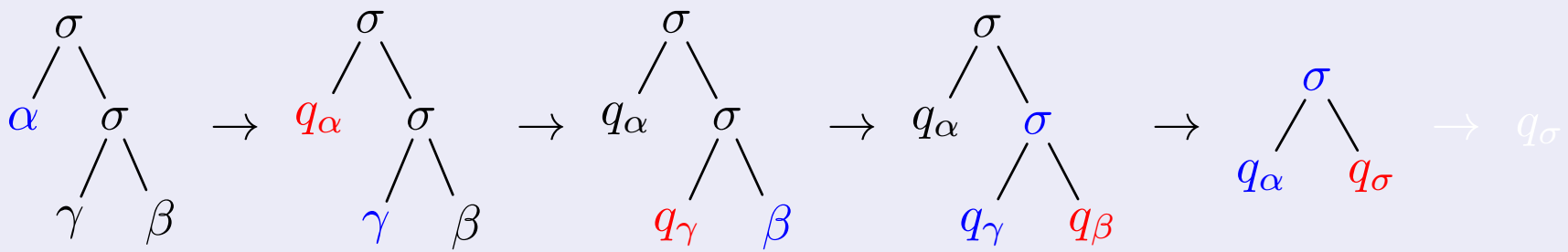
Deterministic Tree Automaton

Example

- states q_α, q_β (nonfinal) and q_γ, q_σ (final)
- nullary input symbols α, β, γ and binary σ
- for all nullary symbols π, π'

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$

Evaluating a tree



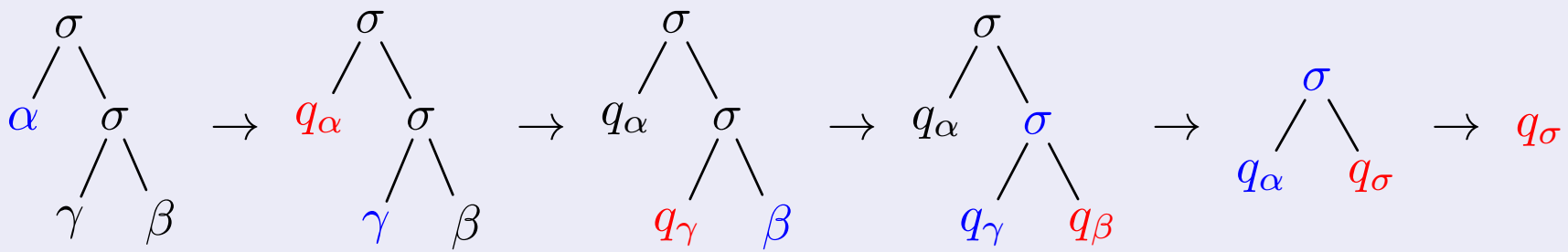
Deterministic Tree Automaton

Example

- states q_α, q_β (nonfinal) and q_γ, q_σ (final)
- nullary input symbols α, β, γ and binary σ
- for all nullary symbols π, π'

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$

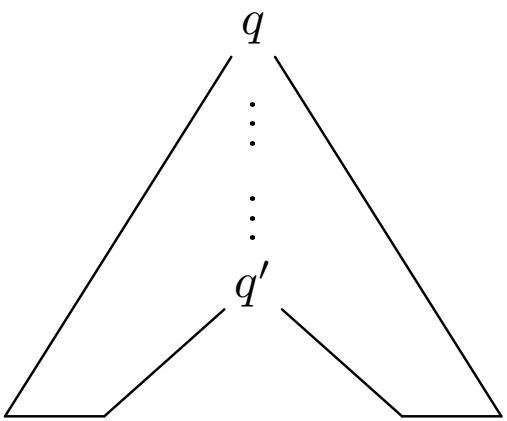
Evaluating a tree



Deterministic Tree Automaton

Shorthands

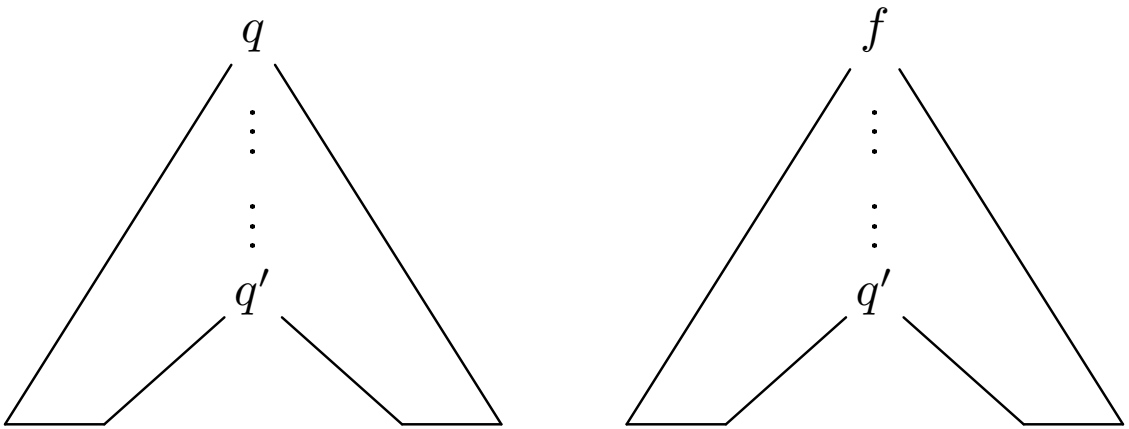
- $L(M)^q_{q'} = \{c \in C_\Sigma \mid \delta(c[q']) = q\}$
- $L(M)_{q'} = \bigcup_{f \in F} L(M)^f_{q'}$
- $L(M)^q = \delta^{-1}(q) \cap T_\Sigma$



Deterministic Tree Automaton

Shorthands

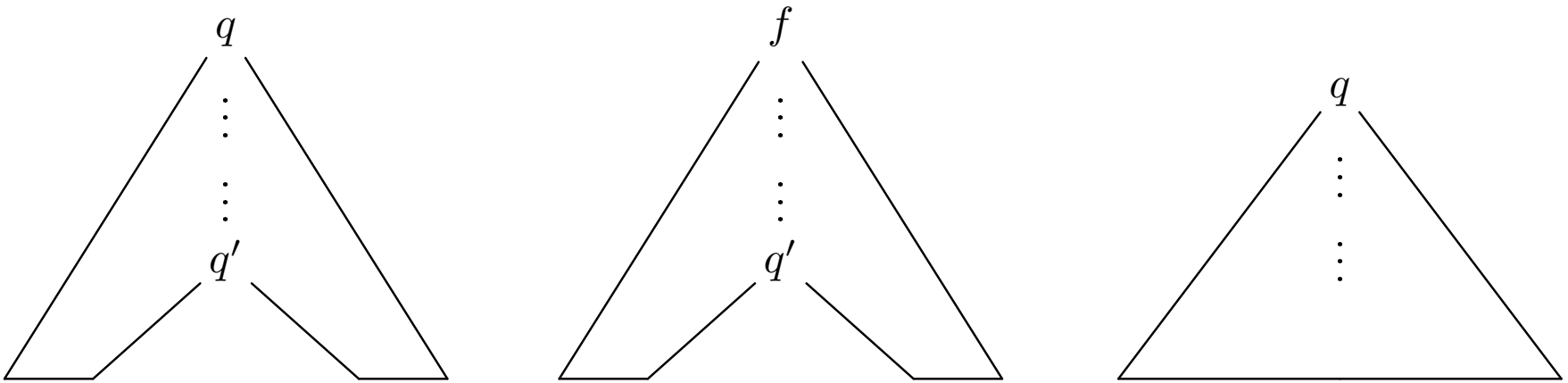
- $L(M)^q_{q'} = \{c \in C_\Sigma \mid \delta(c[q']) = q\}$
- $L(M)_{q'} = \bigcup_{f \in F} L(M)^f_{q'}$
- $L(M)^q = \delta^{-1}(q) \cap T_\Sigma$



Deterministic Tree Automaton

Shorthands

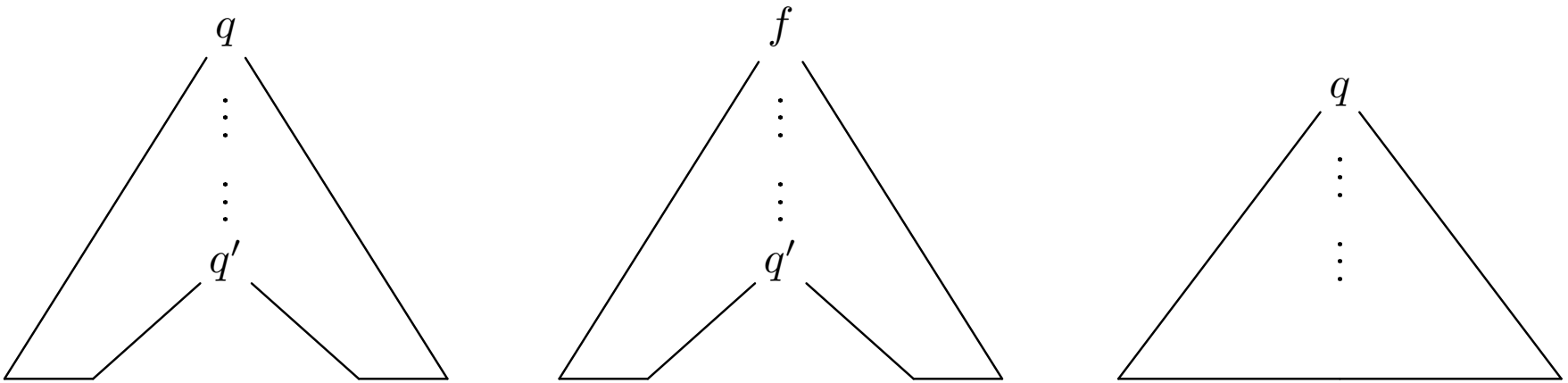
- $L(M)^q_{q'} = \{c \in C_\Sigma \mid \delta(c[q']) = q\}$
- $L(M)_{q'} = \bigcup_{f \in F} L(M)^f_{q'}$
- $L(M)^q = \delta^{-1}(q) \cap T_\Sigma$



Deterministic Tree Automaton

Shorthands

- $L(M)^q_{q'} = \{c \in C_\Sigma \mid \delta(c[q']) = q\}$
- $L(M)_{q'} = \bigcup_{f \in F} L(M)^f_{q'}$
- $L(M)^q = \delta^{-1}(q) \cap T_\Sigma$



Definition

Recognized tree language $L(M) = \bigcup_{f \in F} L(M)^f$

Deterministic Tree Automaton

Example

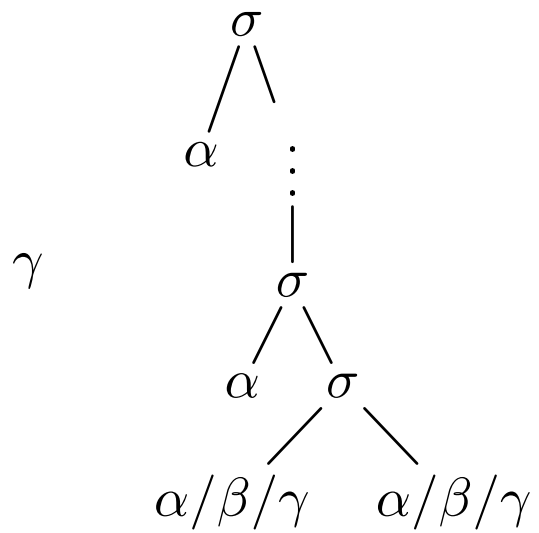
- states q_α, q_β (nonfinal) and q_γ, q_σ (final)
- nullary input symbols α, β, γ and binary σ
- for all nullary symbols π, π'

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$

Recognized tree language

With $c = \sigma(\alpha, \square)$

$$\{\gamma\} \cup \{c^n[\sigma(\pi, \pi')] \mid n \in \mathbb{N},\ \text{nullary } \pi, \pi'\}$$



Minimization

Definition

States q and q' are equivalent if $L(M)_q = L(M)_{q'}$

Theorem

DTA minimal $\iff$ no different, but equivalent states

Theorem (HÖGBERG et al. 2008)

For every DTA we can efficiently construct an equivalent minimal DTA

Minimization

Definition

States q and q' are equivalent if $L(M)_q = L(M)_{q'}$

Theorem

DTA minimal $\iff$ *no different, but equivalent states*

Theorem (HÖGBERG et al. 2008)

For every DTA we can efficiently construct an equivalent minimal DTA

Minimization

Definition

States q and q' are equivalent if $L(M)_q = L(M)_{q'}$

Theorem

DTA minimal $\iff$ *no different, but equivalent states*

Theorem (HÖGBERG et al. 2008)

For every DTA we can efficiently construct an equivalent minimal DTA

Contents

Almost Equivalence

Definition

States q and q' are almost equivalent
if $L(M)_q$ and $L(M)_{q'}$ have finite difference

Example

States q_α, q_β (nonfinal) and q_γ, q_σ (final)

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$

$\rightarrow$ minimal, but q_β and q_γ *almost equivalent*

$$L(M)_{q_\beta} \ominus L(M)_{q_\gamma} = \{\square\}$$

Almost Equivalence

Definition

States q and q' are almost equivalent
if $L(M)_q$ and $L(M)_{q'}$ have finite difference

Example

States q_α, q_β (nonfinal) and q_γ, q_σ (final)

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$

$\rightarrow$ minimal, but q_β and q_γ *almost equivalent*

$$L(M)_{q_\beta} \ominus L(M)_{q_\gamma} = \{\Box\}$$

Properties of Almost Equivalence

Lemma

Almost equivalent states agree on deep contexts

$\delta(c[q_1]) = \delta(c[q_2])$ *for suitably deep* c

Contrast

On strings: the converse also holds

On trees: the converse is not true

Properties of Almost Equivalence

Lemma

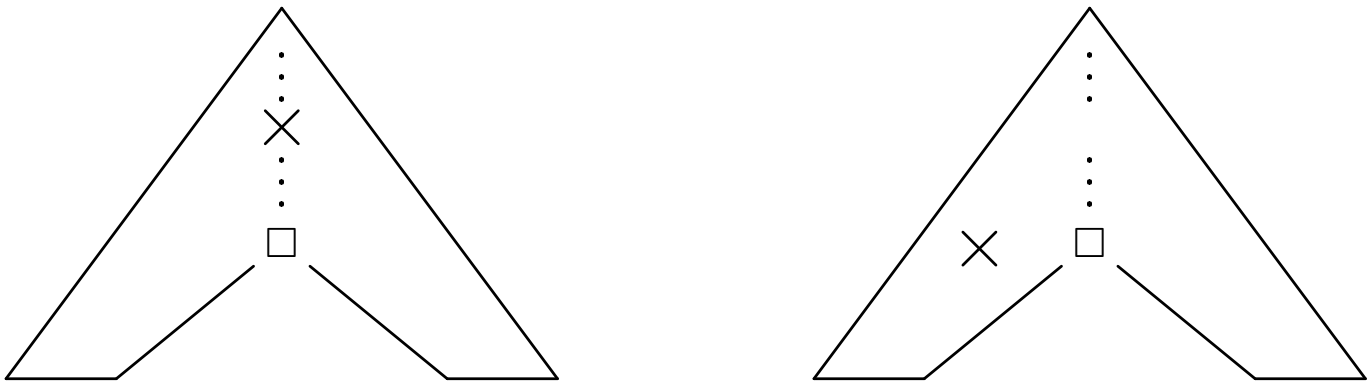
Almost equivalent states agree on deep contexts

$\delta(c[q_1]) = \delta(c[q_2])$ *for suitably deep* c

Contrast

On strings: the converse also holds

On trees: the converse is not true



Properties of Almost Equivalence

Lemma (Lemma 2.10 of BADR et al. 2007)

- $\delta(c[q])$ *and* $\delta(c[q'])$ *are almost equivalent for almost equivalent* q *and* q' *and context* c
- *Almost equivalence is a congruence*

Definition

DTA are almost equivalent if they recognize tree languages with finite difference

Lemma

Almost equivalent DTA evaluate trees to almost equivalent states

$L(M)_{\delta(t)}$ *and* $L(N)_{\mu(t)}$ *are almost equal*

Properties of Almost Equivalence

Lemma (Lemma 2.10 of BADR et al. 2007)

- $\delta(c[q])$ *and* $\delta(c[q'])$ *are almost equivalent for almost equivalent* q *and* q' *and context* c
- *Almost equivalence is a congruence*

Definition

DTA are almost equivalent if they recognize tree languages with finite difference

Lemma

Almost equivalent DTA evaluate trees to almost equivalent states

$L(M)_{\delta(t)}$ *and* $L(N)_{\mu(t)}$ *are almost equal*

Properties of Almost Equivalence

Lemma (Lemma 2.10 of BADR et al. 2007)

- $\delta(c[q])$ *and* $\delta(c[q'])$ *are almost equivalent for almost equivalent* q *and* q' *and context* c
- *Almost equivalence is a congruence*

Definition

DTA are almost equivalent if they recognize tree languages with finite difference

Lemma

Almost equivalent DTA evaluate trees to almost equivalent states

$L(M)_{\delta(t)}$ *and* $L(N)_{\mu(t)}$ *are almost equal*

Properties of Almost Equivalence

Lemma (Lemma 2.10 of BADR et al. 2007)

- $\delta(c[q])$ *and* $\delta(c[q'])$ *are almost equivalent for almost equivalent* q *and* q' *and context* c
- *Almost equivalence is a congruence*

Definition

DTA are almost equivalent if they recognize tree languages with finite difference

Lemma

Almost equivalent DTA evaluate trees to almost equivalent states

$L(M)_{\delta(t)}$ *and* $L(N)_{\mu(t)}$ *are almost equal*

State Merging

Definition

Merge of q into q': redirect all transitions leading to q into q'

Definition

State q is a preamble state if $L(M)^q$ is finite

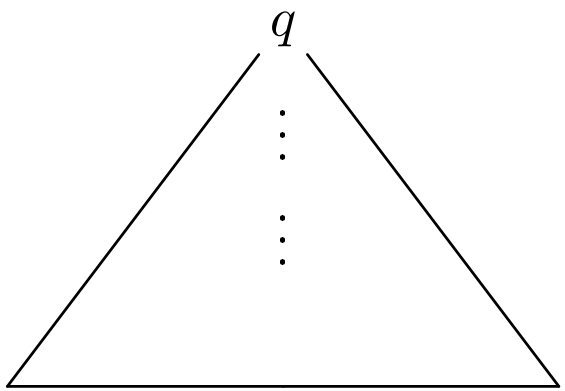
State Merging

Definition

Merge of q into q': redirect all transitions leading to q into q'

Definition

State q is a preamble state if $L(M)^q$ is finite



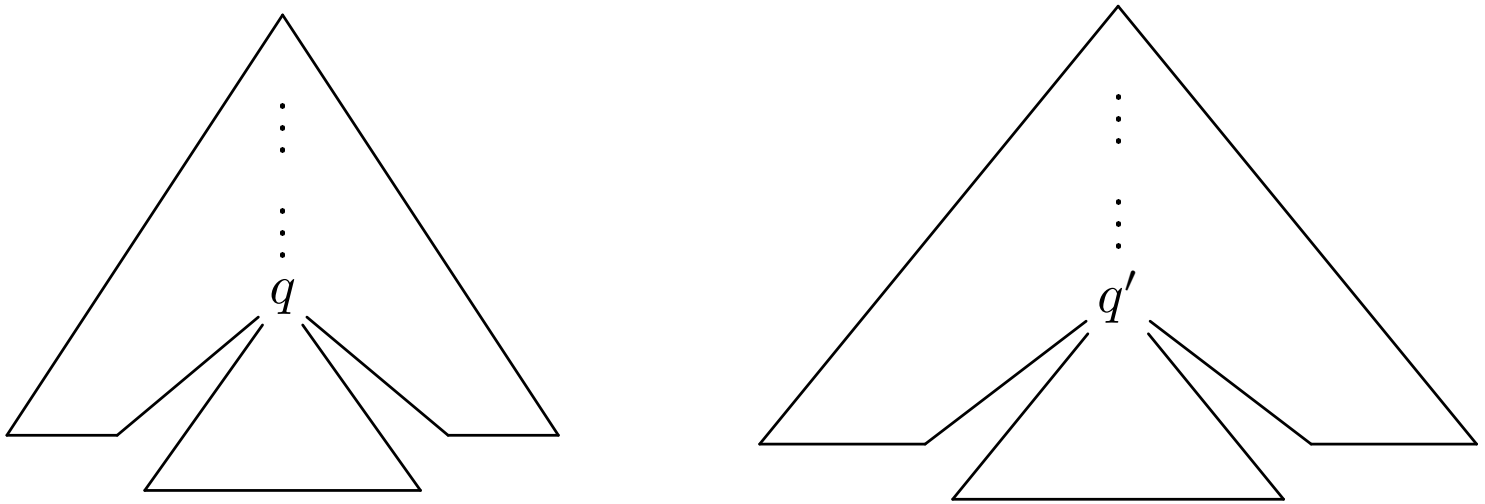
State Merging

Lemma

If

- *q and q' are almost equivalent*
- *q is a preamble state*

then merging q into q' yields an almost equivalent DTA



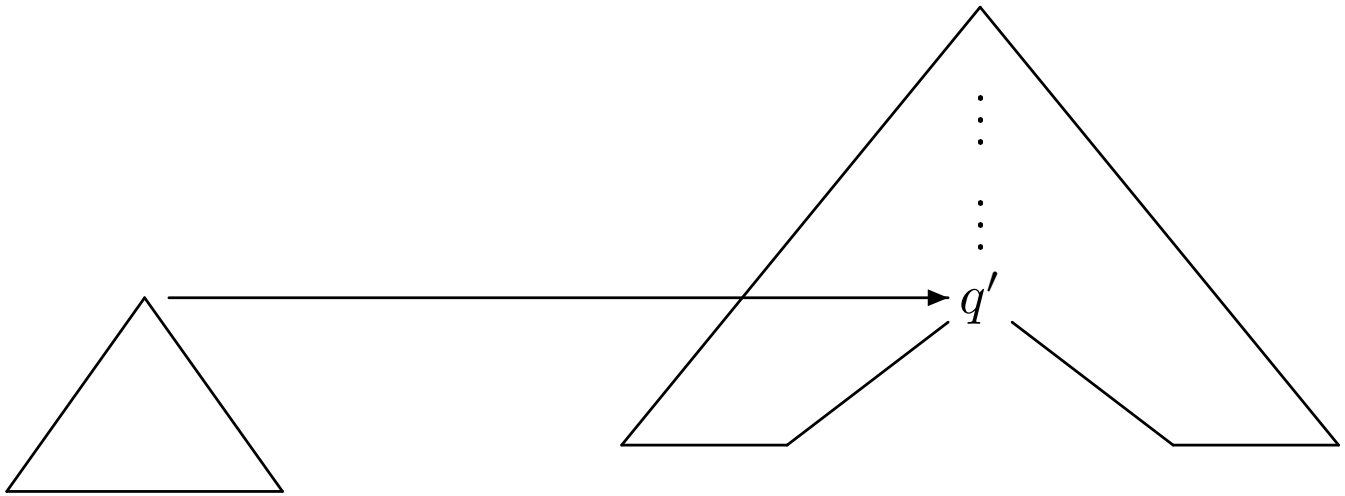
State Merging

Lemma

If

- *q and q' are almost equivalent*
- *q is a preamble state*

then merging q into q' yields an almost equivalent DTA



State Merging

Example (Original)

States q_α, q_β (nonfinal) and q_γ, q_σ (final)

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$

Example (Merged)

Merging q_β into q_γ yields

$$\alpha \mapsto q_\alpha \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$
$$\beta \mapsto q_\gamma$$
$$\gamma \mapsto q_\gamma$$

State Merging

Example (Original)

States q_α, q_β (nonfinal) and q_γ, q_σ (final)

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$

Example (Merged)

Merging q_β into q_γ yields

$$\alpha \mapsto q_\alpha \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$
$$\beta \mapsto q_\gamma$$
$$\gamma \mapsto q_\gamma$$

Hyper-minimal DTA

Definition

DTA hyper-minimal if there is no smaller DTA that is almost equivalent

Theorem

DTA hyper-minimal $\iff$ *no different, but almost equivalent states involving a preamble state*

Hyper-minimal DTA

Definition

DTA hyper-minimal if there is no smaller DTA that is almost equivalent

Theorem

DTA hyper-minimal $\iff$ *no different, but almost equivalent states involving a preamble state*

Hyper-minimal DTA

Example (Original)

States q_α, q_β (nonfinal) and q_γ, q_σ (final)

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$

$\to$ not hyper-minimal (q_β and q_γ almost equivalent)

Example (Merged)

Merging q_β into q_γ yields

$$\alpha \mapsto q_\alpha \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$
$$\beta \mapsto q_\gamma$$
$$\gamma \mapsto q_\gamma$$

$\to$ hyper-minimal

Hyper-minimal DTA

Example (Original)

States q_α, q_β (nonfinal) and q_γ, q_σ (final)

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$

$\rightarrow$ not hyper-minimal (q_β and q_γ almost equivalent)

Example (Merged)

Merging q_β into q_γ yields

$$\alpha \mapsto q_\alpha \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$
$$\beta \mapsto q_\gamma$$
$$\gamma \mapsto q_\gamma$$

$\rightarrow$ hyper-minimal

Contents

Overview

Require: DTA M
Return: almost equivalent hyper-minimal DTA

$M \leftarrow \textsc{Minimize}(M)$ // complexity: $\mathcal{O}(\ell m \log n)$
$P \leftarrow \textsc{ComputePreamble}(M)$ // complexity: $\mathcal{O}(\ell m)$
$\sim \leftarrow \{\langle q, q' \rangle \mid L(M_\otimes)_{\langle q,q' \rangle}$ is finite$\}$
for all $B \in (Q/\sim)$ **do**
  select $q_B \in B$ such that $q_B \notin P$ if possible
  merge each $q \in B \cap P$ into q_B // complexity: $\mathcal{O}(|B|)$
end for
return M

Overview

Require: DTA M
Return: almost equivalent hyper-minimal DTA

$M \leftarrow \textsc{Minimize}(M)$ // complexity: $\mathcal{O}(\ell m \log n)$
$P \leftarrow \textsc{ComputePreamble}(M)$ // complexity: $\mathcal{O}(\ell m)$
$\sim \leftarrow \{\langle q, q' \rangle \mid L(M_\otimes)_{\langle q,q' \rangle}$ is finite$\}$
for all $B \in (Q/\sim)$ **do**
  select $q_B \in B$ such that $q_B \notin P$ if possible
  merge each $q \in B \cap P$ into q_B // complexity: $\mathcal{O}(|B|)$
end for
return M

Overview

Require: DTA M
Return: almost equivalent hyper-minimal DTA

$M \leftarrow \textsc{Minimize}(M)$ // complexity: $\mathcal{O}(\ell m \log n)$
$P \leftarrow \textsc{ComputePreamble}(M)$ // complexity: $\mathcal{O}(\ell m)$
$\sim \leftarrow \{\langle q, q' \rangle \mid L(M_\otimes)_{\langle q,q' \rangle}$ is finite$\}$
for all $B \in (Q/\sim)$ **do**
  select $q_B \in B$ such that $q_B \notin P$ if possible
  merge each $q \in B \cap P$ into q_B // complexity: $\mathcal{O}(|B|)$
end for
return M

Computing Preamble States

Lemma

The preamble states can be computed in time $O(\ell m)$

Computing Preamble States

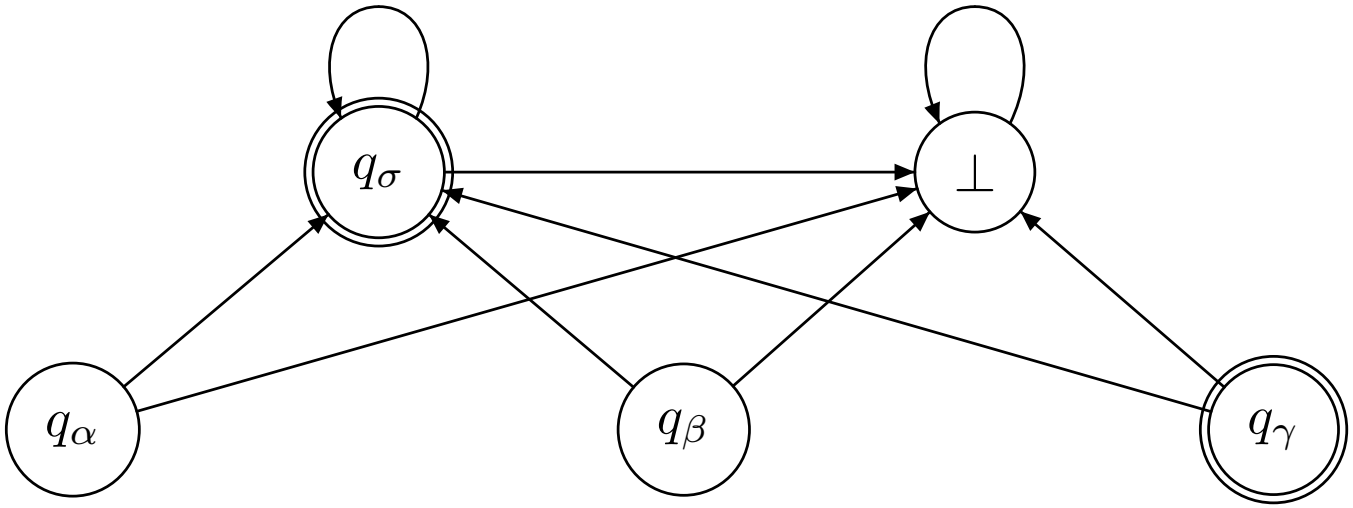
Lemma

The preamble states can be computed in time $O(\ell m)$

Example

States q_α, q_β (nonfinal) and q_γ, q_σ (final)

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$



Computing Preamble States

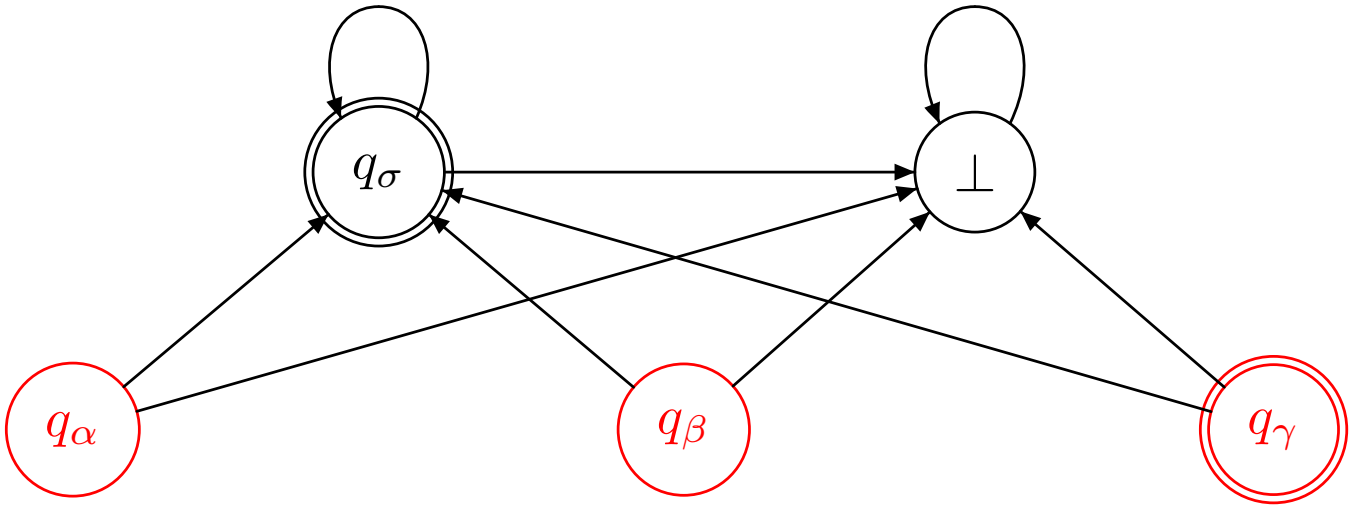
Lemma

The preamble states can be computed in time $O(\ell m)$

Example

States q_α, q_β (nonfinal) and q_γ, q_σ (final)

$$\pi \mapsto q_\pi \qquad \sigma(q_\pi, q_{\pi'}) \mapsto q_\sigma \qquad \sigma(q_\alpha, q_\sigma) \mapsto q_\sigma$$



Computing Almost Equivalence

Definition

Exclusive-or single-point self-product

$$M_{\otimes} = (Q \cup Q^2, \Sigma, \delta \cup \delta', F')$$

- $F' = \{\langle q, q' \rangle \mid \text{ either } q \in F \text{ or } q' \in F\}$
- $\delta'(\sigma(c[\langle q, q' \rangle])) = \langle \delta(c[q]), \delta(c[q']) \rangle$

Lemma

We can construct $M_{\otimes}$ in time $\mathcal{O}(\ell mn)$.

Theorem

$L(M_{\otimes})_{\langle q,q' \rangle}$ is finite $\iff$ q and q' are almost equivalent

Computing Almost Equivalence

Definition

Exclusive-or single-point self-product

$$M_{\otimes} = (Q \cup Q^2, \Sigma, \delta \cup \delta', F')$$

- $F' = \{\langle q, q' \rangle \mid \text{ either } q \in F \text{ or } q' \in F\}$
- $\delta'(\sigma(c[\langle q, q' \rangle])) = \langle \delta(c[q]), \delta(c[q']) \rangle$

Lemma

We can construct $M_{\otimes}$ in time $\mathcal{O}(\ell mn)$.

Theorem

$L(M_{\otimes})_{\langle q,q' \rangle}$ is finite $\iff$ q and q' are almost equivalent

Computing Almost Equivalence

Definition

Exclusive-or single-point self-product

$$M_{\otimes} = (Q \cup Q^2, \Sigma, \delta \cup \delta', F')$$

- $F' = \{\langle q, q' \rangle \mid \text{ either } q \in F \text{ or } q' \in F\}$
- $\delta'(\sigma(c[\langle q, q' \rangle])) = \langle \delta(c[q]), \delta(c[q']) \rangle$

Lemma

We can construct $M_{\otimes}$ in time $\mathcal{O}(\ell mn)$.

Theorem

$L(M_{\otimes})_{\langle q,q' \rangle}$ is finite $\iff$ q and q' are almost equivalent

Main Result

Theorem

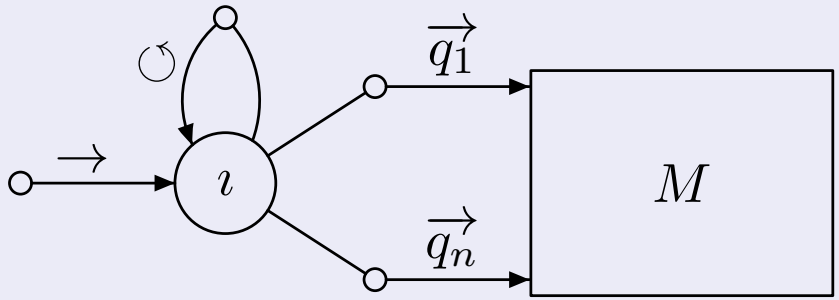
Hyper-minimization can be performed in time $\mathcal{O}(\ell mn)$

Contents

Remaining Issues

Reduction

Minimization linearly reduces to hyper-minimization



Efficiency Improvement

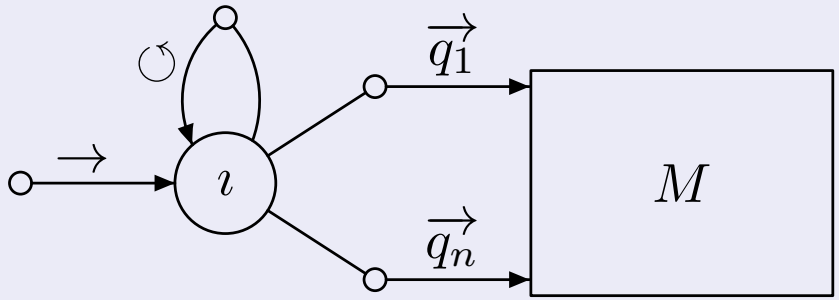
Using complex hashes we can improve hyper-minimization to

$$\mathcal{O}(\ell m \log n)$$

Remaining Issues

Reduction

Minimization linearly reduces to hyper-minimization



Efficiency Improvement

Using complex hashes we can improve hyper-minimization to

$$\mathcal{O}(\ell m \log n)$$

Conclusion

Definition

States q and q' are **equivalent**
if $L(M)_q$ and $L(M)_{q'}$ are equal

Theorem

DTA minimal $\iff$ no different, but equivalent states

Theorem ()

We can efficiently construct an equivalent minimal DTA

Conclusion

Definition

States q and q' are almost equivalent
if $L(M)_q$ and $L(M)_{q'}$ are almost equal

Theorem

DTA minimal $\iff$ no different, but equivalent states

Theorem ()

We can efficiently construct an equivalent minimal DTA

Conclusion

Definition

States q and q' are almost equivalent
if $L(M)_q$ and $L(M)_{q'}$ are almost equal

Theorem

DTA *minimal* $\iff$ *no different, but* *equivalent states*

Theorem ()

We can efficiently construct an equivalent minimal DTA

Conclusion

Definition

States q and q' are almost equivalent
if $L(M)_q$ and $L(M)_{q'}$ are almost equal

Theorem

DTA hyper-minimal $\iff$ *no different, but almost equivalent states involving a preamble state*

Theorem ()

We can efficiently construct an equivalent minimal DTA

Conclusion

Definition

States q and q' are almost equivalent
if $L(M)_q$ and $L(M)_{q'}$ are almost equal

Theorem

DTA hyper-minimal $\iff$ *no different, but almost equivalent states involving a preamble state*

Theorem (HÖGBERG et al. 2008)

We can efficiently construct an equivalent minimal DTA

Conclusion

Definition

States q and q' are almost equivalent
if $L(M)_q$ and $L(M)_{q'}$ are almost equal

Theorem

DTA hyper-minimal $\iff$ *no different, but almost equivalent states involving a preamble state*

Theorem (here)

We can efficiently construct an equivalent hyper-minimal DTA

Conclusion

Solved problems

- Characterization of hyper-minimality
- Hyper-minimization algorithm $\mathcal{O}(\ell mn)$
- Minimization reduces to hyper-minimization

Open problems

- Can hashes be avoided in optimized version?
- Error optimization efficiently possible?
- Sub-quadratic error optimization?

Conclusion

Solved problems

- Characterization of hyper-minimality
- Hyper-minimization algorithm $\mathcal{O}(\ell mn)$
- Minimization reduces to hyper-minimization

Open problems

- Can hashes be avoided in optimized version?
- Error optimization efficiently possible?
- Sub-quadratic error optimization?

References

- Badr, Geffert, Shipman: *Hyper-minimizing minimized deterministic finite state automata*. ITA 43, 2009
- Badr: *Hyper-minimization in $O(n^2)$*. IJFCS 20, 2009
- Gawrychowski, Jeż: *Hyper-minimisation made efficient*. MFCS 2009
- Gawrychowski, Jeż, Maletti: *On minimising automata with errors*. MFCS 2011
- Gécseg, Steinby: *Tree automata*. Akadémiai Kiadó, 1984
- Högberg, May, Maletti: *Backward and forward bisimulation minimization of tree automata*. TCS 410, 2009
- Holzer, Jakobi *From qquivalence to almost-equivalence, and beyond — minimizing automata with errors*. DLT 2012
- Holzer, Maletti: *An $n \log n$ algorithm for hyper-minimizing states in a (minimized) deterministic automaton*. TCS 411, 2010
- Maletti, Quernheim: *Optimal hyper-minimization*. IJFCS 22, 2011
- Schewe: *Beyond hyper-minimisation — minimising DBAs and DPAs is NP-complete*. FSTTCS 2010