Hyper-minimization for deterministic weighted tree automata

Andreas Maletti and Daniel Quernheim

Institute of Computer Science, Universität Leipzig, Germany
maletti@informatik.uni-leipzig.de

May 29, 2014

Overview

Weighted Tree Language

- Assigns weight (e.g. a probability) to each tree
- Weight drawn from commutative semiring; e.g. $(\mathbb{Q}, +, \cdot, 0, 1)$

Overview

Weighted Tree Language

- Assigns weight (e.g. a probability) to each tree
- Weight drawn from commutative semiring; e.g. $(\mathbb{Q}, +, \cdot, 0, 1)$

Weighted Tree Automaton

- Finitely represents weighted tree language
- Defines the recognizable weighted tree languages

Overview

Weighted Tree Language

- Assigns weight (e.g. a probability) to each tree
- Weight drawn from commutative semiring; e.g. $(\mathbb{Q}, +, \cdot, 0, 1)$

Weighted Tree Automaton

- Finitely represents weighted tree language
- Defines the recognizable weighted tree languages

Application

- Re-ranker for parse trees
- Representation of parses

large models

Basics

Semiring

Definition

A commutative semiring is an algebraic structure $\mathcal{A} = (A, +, \cdot, 0, 1)$

- $(A, +, 0)$ commutative monoid
- $(A, \cdot, 1)$ commutative monoid
- $\cdot$ distributes over $+$ $\quad a \cdot (a_1 + a_2) = (a \cdot a_1) + (a \cdot a_2)$
- $0 \cdot a = 0$ for all $a \in A$

Examples: $(\mathbb{N}, +, \cdot, 0, 1)$ and $(\mathbb{Q}, +, \cdot, 0, 1)$

Semiring

Definition

A commutative semiring is an algebraic structure $\mathcal{A} = (A, +, \cdot, 0, 1)$

- $(A, +, 0)$ commutative monoid
- $(A, \cdot, 1)$ commutative monoid
- $\cdot$ distributes over $+$ $\qquad a \cdot (a_1 + a_2) = (a \cdot a_1) + (a \cdot a_2)$
- $0 \cdot a = 0$ for all $a \in A$

Examples: $(\mathbb{N}, +, \cdot, 0, 1)$ and $(\mathbb{Q}, +, \cdot, 0, 1)$

Definition

A commutative semifield is a commutative semiring $\mathcal{A} = (A, +, \cdot, 0, 1)$

- for all $a \in A \setminus \{0\}$ there exists $a^{-1} \in A$ with $a \cdot a^{-1} = 1$

Example: $(\mathbb{Q}, +, \cdot, 0, 1)$

Syntax

Definition

Weighted tree automaton (WTA) is tuple $(Q, \Sigma, \mathcal{A}, F, \mu)$ where

- finite set Q *states*
- ranked alphabet Σ *input symbols*
- commutative semiring $\mathcal{A} = (A, +, \cdot, 0, 1)$ *weight structure*
- $F \subseteq Q$ *final states*

Syntax

Definition

Weighted tree automaton (WTA) is tuple $(Q, \Sigma, \mathcal{A}, F, \mu)$ where

- finite set Q *states*
- ranked alphabet Σ *input symbols*
- commutative semiring $\mathcal{A} = (A, +, \cdot, 0, 1)$ *weight structure*
- $F \subseteq Q$ *final states*
- $\mu = (\mu_k)_{k \in \mathbb{N}}$ with $\mu_k \colon \Sigma_k \to A^{Q \times Q^k}$ *weighted transitions*

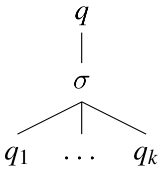
Syntax

Definition

Weighted tree automaton (WTA) is tuple $(Q, \Sigma, \mathcal{A}, F, \mu)$ where

- finite set Q — *states*
- ranked alphabet Σ — *input symbols*
- commutative semiring $\mathcal{A} = (A, +, \cdot, 0, 1)$ — *weight structure*
- $F \subseteq Q$ — *final states*
- $\mu = (\mu_k)_{k \in \mathbb{N}}$ with $\mu_k \colon \Sigma_k \to A^{Q \times Q^k}$ — *weighted transitions*

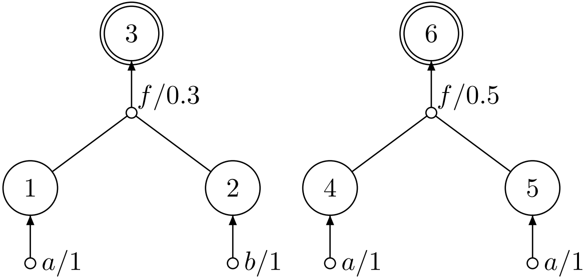
Sample Transition



with weight $\mu_k(\sigma)_{q, q_1 \cdots q_k}$

Syntax — Illustration

Sample Automaton



Semantics

Definition

Let $t \in T_\Sigma(Q)$ and $W = \mathrm{pos}(t)$.

- Run on t: map $r\colon W \to Q$ with $r(w) = t(w)$ if $t(w) \in Q$

Semantics

Definition

Let $t \in T_\Sigma(Q)$ and $W = \mathrm{pos}(t)$.

- **Run** on t: map $r\colon W \to Q$ with $r(w) = t(w)$ if $t(w) \in Q$
- **Weight** of r

$$\mathrm{wt}(r) = \prod_{\substack{w \in W \\ t(w) \in \Sigma}} \mu_k(t(w))_{r(w), r(w1) \cdots r(wk)}$$

Semantics

Definition

Let $t \in T_\Sigma(Q)$ and $W = \mathrm{pos}(t)$.

- Run on t: map $r\colon W \to Q$ with $r(w) = t(w)$ if $t(w) \in Q$
- Weight of r

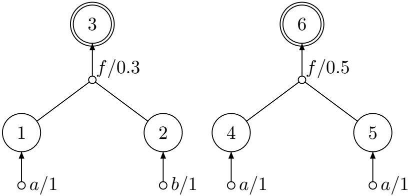
$$\mathrm{wt}(r) = \prod_{\substack{w \in W \\ t(w) \in \Sigma}} \mu_k(t(w))_{r(w), r(w1) \cdots r(wk)}$$

- Recognized weighted tree language

$$\|M\|(t) = \sum_{\substack{r \text{ run on } t \\ r(\varepsilon) \in F}} \mathrm{wt}(r)$$

Semantics — Illustration

Sample Automaton

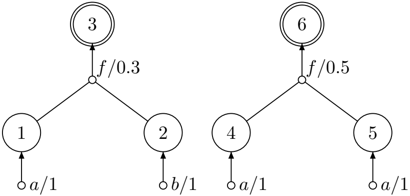


Sample Runs

Input tree: $f(a, b)$ Runs: $6(1, 2)$ with weight 0

Semantics — Illustration

Sample Automaton



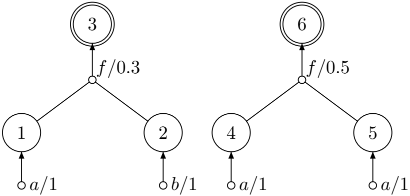
Sample Runs

Input tree: f Runs: f with weight

a b a b

Semantics — Illustration

Sample Automaton

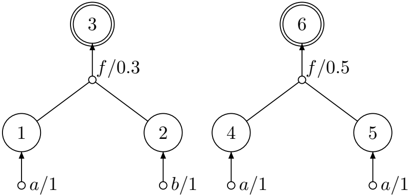


Sample Runs

Input tree: $f(a, b)$ Runs: $f(1, b)$ with weight 1

Semantics — Illustration

Sample Automaton

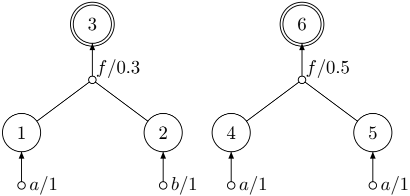


Sample Runs

Input tree: $f(a, b)$ Runs: $f(1, 2)$ with weight 1

Semantics — Illustration

Sample Automaton



Sample Runs

Input tree: $f(a, b)$ Runs: $3(1, 2)$ with weight 0.3

Determinism

Definition

Deterministic WTA: for every $\sigma \in \Sigma_k$ and $w \in Q^k$ there exists exactly one $q \in Q$ such that $\mu_k(\sigma)_{q,w} \neq 0$

Determinism

Definition

Deterministic WTA: for every $\sigma \in \Sigma_k$ and $w \in Q^k$ there exists exactly one $q \in Q$ such that $\mu_k(\sigma)_{q,w} \neq 0$

Notes

- Deterministic WTA does not use addition

Determinism

Definition

Deterministic WTA: for every $\sigma \in \Sigma_k$ and $w \in Q^k$ there exists exactly one $q \in Q$ such that $\mu_k(\sigma)_{q,w} \neq 0$

Notes

- Deterministic WTA does not use addition
- Recognizable $\neq$ deterministically recognizable

Determinism

Definition

Deterministic WTA: for every $\sigma \in \Sigma_k$ and $w \in Q^k$ there exists exactly one $q \in Q$ such that $\mu_k(\sigma)_{q,w} \neq 0$

Notes

- Deterministic WTA does not use addition
- Recognizable $\neq$ deterministically recognizable
- Determinization possible in locally-finite semirings

[Borchardt, Vogler 2003]

Determinism

Definition

Deterministic WTA: for every $\sigma \in \Sigma_k$ and $w \in Q^k$ there exists exactly one $q \in Q$ such that $\mu_k(\sigma)_{q,w} \neq 0$

Notes

- Deterministic WTA does not use addition
- Recognizable $\neq$ deterministically recognizable
- Determinization possible in locally-finite semirings [Borchardt, Vogler 2003]
- Partial determinization for probabilities [May, Knight 2006]
- Systematic presentation [Büchse, Vogler 2009]

Hyper-minimization

Assumption

We assume a commutative semifield $\mathcal{A} = (A, +, \cdot, 0, 1)$

Minimization

equivalent = same recognized weighted tree language

Problem

Given deterministic WTA, return

- equivalent deterministic WTA such that
- no equivalent deterministic WTA is smaller

Minimization

equivalent = same recognized weighted tree language

Problem

Given deterministic WTA, return

- equivalent deterministic WTA
- minimal

Minimization

equivalent = same recognized weighted tree language

Problem

Given deterministic WTA, return

- equivalent deterministic WTA
- minimal

Theorem (M., Q. 2011)

Minimization of deterministic WTA can be done in time $\mathcal{O}(m \log n)$

- m = *size of automaton*
- n = *number of states*

Minimization

context = tree with exactly one occurrence of special symbol $\Box$
$c[t]$ = tree obtained from context c by replacing $\Box$ by t

Definition

States p and q are equivalent if there exists $a \in A \setminus \{0\}$ such that

$$\|M\|(c[p]) = a \cdot \|M\|(c[q])$$

for all contexts $c \in C_\Sigma$

Minimization

context = tree with exactly one occurrence of special symbol $\square$
$c[t]$ = tree obtained from context c by replacing $\square$ by t

Definition

States p and q are equivalent if there exists $a \in A \setminus \{0\}$ such that

$$\|M\|(c[p]) = a \cdot \|M\|(c[q])$$

for all contexts $c \in C_\Sigma$

Theorem (Borchardt 2003)

A trim deterministic WTA is minimal
$\iff$ *no pair of different, but equivalent states*

Hyper-minimization

Definition

Languages L and L' almost equal if L and L' have finite difference

$$(L \setminus L') \cup (L' \setminus L)$$

Problem [Badr et al. 2009]

Given DFA, return

- DFA recognizing almost equal language such that
- no smaller DFA recogizes an almost equal language

Hyper-minimization

Definition

Languages L and L' almost equal if L and L' have finite difference

$$(L \setminus L') \cup (L' \setminus L)$$

Problem [Badr et al. 2009]

Given DFA, return

- DFA recognizing almost equal language
- hyper-minimal

Hyper-minimization

Definition

Languages L and L' almost equal if L and L' have finite difference

$$(L \setminus L') \cup (L' \setminus L)$$

Problem [Badr et al. 2009]

Given DFA, return

- DFA recognizing almost equal language
- hyper-minimal

Theorem (Holzer, M. 2009, Gawrychowsky, Jeż 2009)

DFA hyper-minimization can be done in time $\mathcal{O}(n \log n)$

Weighted hyper-minimization

$\mathrm{supp}(\tau) = \{t \in T_\Sigma \mid \tau(t) \neq 0\}$ for $\tau \colon T_\Sigma \to A$

Three variants

Two weighted tree languages $\tau_1, \tau_2 \colon T_\Sigma \to A$ are almost equal if

- $\mathrm{supp}(\tau_1)$ and $\mathrm{supp}(\tau_2)$ are almost equal

reduces to the unweighted case

Weighted hyper-minimization

$\mathrm{supp}(\tau) = \{t \in T_\Sigma \mid \tau(t) \neq 0\}$ for $\tau \colon T_\Sigma \to A$

Three variants

Two weighted tree languages $\tau_1, \tau_2 \colon T_\Sigma \to A$ are almost equal if

- $\mathrm{supp}(\tau_1)$ and $\mathrm{supp}(\tau_2)$ are almost equal

  reduces to the unweighted case

- $\{t \in T_\Sigma \mid \tau_1(t) \neq \tau_2(t)\}$ finite

Weighted hyper-minimization

$\operatorname{supp}(\tau) = \{t \in T_\Sigma \mid \tau(t) \neq 0\}$ for $\tau \colon T_\Sigma \to A$

Three variants

Two weighted tree languages $\tau_1, \tau_2 \colon T_\Sigma \to A$ are almost equal if

- $\operatorname{supp}(\tau_1)$ and $\operatorname{supp}(\tau_2)$ are almost equal

  reduces to the unweighted case

- $\{t \in T_\Sigma \mid \tau_1(t) \neq \tau_2(t)\}$ finite
- $\sum_{t \in T_\Sigma} d(\tau_1(t), \tau_2(t)) \leq n$ for some distance d and $n \in \mathbb{N}$

  difficult

Weighted hyper-minimization

$\mathrm{supp}(\tau) = \{t \in T_\Sigma \mid \tau(t) \neq 0\}$ for $\tau \colon T_\Sigma \to A$

Three variants

Two weighted tree languages $\tau_1, \tau_2 \colon T_\Sigma \to A$ are almost equal if

- $\mathrm{supp}(\tau_1)$ and $\mathrm{supp}(\tau_2)$ are almost equal

  reduces to the unweighted case

- $\{t \in T_\Sigma \mid \tau_1(t) \neq \tau_2(t)\}$ finite
- $\sum_{t \in T_\Sigma} d(\tau_1(t), \tau_2(t)) \leq n$ for some distance d and $n \in \mathbb{N}$

  difficult

Weighted hyper-minimization

$\mathrm{supp}(\tau) = \{t \in T_\Sigma \mid \tau(t) \neq 0\}$ for $\tau \colon T_\Sigma \to A$

Three variants

Two weighted tree languages $\tau_1, \tau_2 \colon T_\Sigma \to A$ are almost equal if

- $\mathrm{supp}(\tau_1)$ and $\mathrm{supp}(\tau_2)$ are almost equal
  reduces to the unweighted case
- $\{t \in T_\Sigma \mid \tau_1(t) \neq \tau_2(t)\}$ finite $\leftarrow$ discussed here
- $\sum_{t \in T_\Sigma} d(\tau_1(t), \tau_2(t)) \leq n$ for some distance d and $n \in \mathbb{N}$
  difficult

Weighted hyper-minimization

Definition

Two weighted tree languages τ_1 and τ_2 are almost equal if

$$\tau_1(t) = \tau_2(t)$$

for almost all $t \in T_\Sigma$

Weighted hyper-minimization

Definition

States p and q are almost equivalent if there exists $a \in A \setminus \{0\}$ such that

$$\|M\|(c[p]) = a \cdot \|M\|(c[q])$$

for almost all contexts $c \in C_\Sigma$

Weighted hyper-minimization

Definition

States p and q are almost equivalent if there exists $a \in A \setminus \{0\}$ such that

$$\|M\|(c[p]) = a \cdot \|M\|(c[q])$$

for almost all contexts $c \in C_\Sigma$

Definition

- q-run = non-zero weighted run with root label q
- preamble state q = finitely many q-runs

Weighted hyper-minimization

Definition

States p and q are almost equivalent if there exists $a \in A \setminus \{0\}$ such that

$$\|M\|(c[p]) = a \cdot \|M\|(c[q])$$

for almost all contexts $c \in C_\Sigma$

Definition

- q-run = non-zero weighted run with root label q
- preamble state q = finitely many q-runs

Theorem

A minimal deterministic WTA is hyper-minimal
$\iff$ *no pair of different, but almost equivalent states*
of which one is a preamble state

Algorithm

Overview

Hyper-minimization algorithm

1. Minimize $\mathcal{O}(m \log n)$
2. Compute preamble states $\mathcal{O}(m)$
3. Compute co-preamble states $\mathcal{O}(m)$
4. Identify almost equivalent states $\mathcal{O}(m \log n)$
5. Merge preamble states that are almost equivalent to another state $\mathcal{O}(m)$

Overview

Hyper-minimization algorithm

1. Minimize $\mathcal{O}(m \log n)$
2. Compute preamble states $\mathcal{O}(m)$
3. Compute co-preamble states $\mathcal{O}(m)$
4. Identify almost equivalent states $\mathcal{O}(m \log n)$
5. Merge preamble states that are almost equivalent to another state $\mathcal{O}(m)$

Definition

Co-preamble state q = finitely many $c \in C_\Sigma$ such that $\|M\|(c[q]) \neq 0$

Overview

Hyper-minimization algorithm

1. Minimize $\mathcal{O}(m \log n)$
2. Compute preamble states $\mathcal{O}(m)$
3. Compute co-preamble states $\mathcal{O}(m)$
4. Identify almost equivalent states $\mathcal{O}(m \log n)$
5. Merge preamble states that are almost equivalent to another state $\mathcal{O}(m)$

Definition

Co-preamble state q = finitely many $c \in C_\Sigma$ such that $\|M\|(c[q]) \neq 0$

Identification of almost equivalent states

Definition

Transition context c is of the shape $\sigma(t_1, \ldots, t_k)$ with

- $t_1, \ldots, t_k \in Q \cup \{\Box\}$
- exactly one $\Box$ occurs

Identification of almost equivalent states

Definition

Transition context c is of the shape $\sigma(t_1, \ldots, t_k)$ with

- $t_1, \ldots, t_k \in Q \cup \{\Box\}$
- exactly one $\Box$ occurs

Assumptions

- total order on transition contexts

Identification of almost equivalent states

Definition

Transition context c is of the shape $\sigma(t_1, \ldots, t_k)$ with

- $t_1, \ldots, t_k \in Q \cup \{\Box\}$
- exactly one $\Box$ occurs

Assumptions

- total order on transition contexts
- c_q smallest transition context such that $c_q[q]$ evaluates to a co-kernel (i.e., not a co-preamble) state
  for each $q \in Q$

Signature

Definition

Signature of q: $\{\langle c, q', a' \rangle \mid \cdots\}$

- c = transition context
- q' = evaluation of $c[q]$
- a' = transition weight of $c[q]$

Signature

Definition

Signature of q: $\{\langle c, q', a' \rangle \mid \cdots\}$

- c = transition context
- q' = evaluation of $c[q]$
- a' = transition weight of $c[q]$

Definition

Standardized signature of q: $\{\langle c, q', a' \rangle \mid q' \text{ co-kernel state, } \cdots\}$

- c = transition context
- q' = evaluation of $c[q]$
- a' = transition weight of $c[q]$ “divided by” transition weight of $c_q[q]$

Signature

Lemma

If two states have the same signature, then they are almost equivalent

Signature

Lemma

If two states have the same signature, then they are almost equivalent

Lemma

If two different states are almost equivalent,
then there exist two different states that have the same signature

Finding almost equivalent states

Approach

1. Find two different states of equal signature
2. Merge them using a scaling factor
3. Go to 1.

This will merge more states than desired,
but identifies almost equivalent states

Overview

Hyper-minimization algorithm

1. Minimize $\mathcal{O}(m \log n)$
2. Compute preamble states $\mathcal{O}(m)$
3. Compute co-preamble states $\mathcal{O}(m)$
4. Identify almost equivalent states $\mathcal{O}(m \log n)$
5. Merge preamble states that are almost equivalent to another state $\mathcal{O}(m)$

Hyper-minimization algorithm

Theorem

We can hyper-minimize deterministic WTA in time $\mathcal{O}(m \log n)$

Summary

Solved

- hyper-minimization for deterministic WTA over semifields
- almost equality = finitely many trees with different weight

Open

- Error optimization
- Stronger "almost equality"
- Avoiding requirements
  (semifield; commutativity; determinism; etc.)

Thank you!

References

1. Badr, Geffert, Shipman: *Hyper-minimizing minimized deterministic finite state automata*. ITA 43, 2009
2. Borchardt: *The Myhill-Nerode theorem for recognizable tree series*. Proc. DLT 2003
3. Borchardt, Vogler: *Determinization of finite state weighted tree automata*. JALC 8, 2003
4. Büchse, May, Vogler: *Determinization of weighted tree automata using factorizations*. JALC 15, 2010
5. Gawrychowski, Jeż: *Hyper-minimisation made efficient*. Proc. MFCS 2009
6. Holzer, Maletti: *An $n \log n$ algorithm for hyper-minimizing states in a (minimized) deterministic automaton*. Proc. CIAA 2009
7. Maletti, Quernheim: *Pushing for weighted tree automata*. Proc. MFCS 2011
8. May, Knight: *A better n-best list: practical determinization of weighted finite tree automata*. Proc. HLT-NAACL 2006