

Minimizing Weighted Tree Automata

Andreas Maletti

Joint work with Johanna Högberg and Jonathan May

Table of Contents

Weighted Tree Automata

Bisimulation Minimization

Minimization of Deterministic Devices

Table of Contents

Weighted Tree Automata

Bisimulation Minimization

Minimization of Deterministic Devices

Overview

Tree Series

- ▶ Assigns a weight to each tree (probabilities will do fine)
- ▶ Weight typically drawn from a semiring; e.g. $(\mathbb{R}, +, \cdot, 0, 1)$

Weighted Tree Automaton

- ▶ Computes a tree series
- ▶ Finitely represents a tree series

Syntax

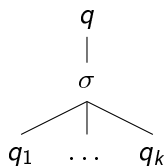
Definition

A *weighted tree automaton* is a tuple $(Q, \Sigma, \mathcal{A}, F, \mu)$ where

- ▶ Q is a finite set (of *states*)
- ▶ Σ is a ranked alphabet (of *input symbols*)
- ▶ $\mathcal{A} = (A, +, \cdot, 0, 1)$ is a semiring (of *weights*)
- ▶ $F \subseteq Q$ (*final states*)
- ▶ $\mu = (\mu_k)_{k \in \mathbb{N}}$ with $\mu_k: \Sigma_k \rightarrow A^{Q \times Q^k}$

Syntax — Illustration

Sample transition:



with weight a

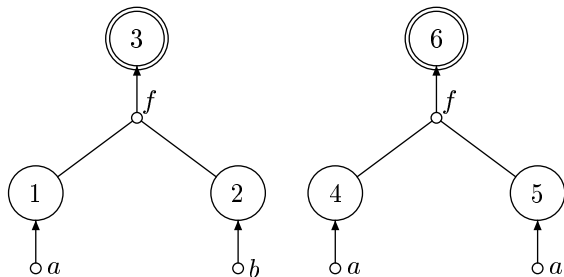


Figure: Example automaton.

Semantics

Definition

Let $t \in T_\Sigma$. A run on t is a map $r: \text{pos}(t) \rightarrow Q$. The weight of r is

$$\text{weight}(r) = \prod_{w \in \text{pos}(t)} \mu_k(t(w))_{r(w), r(w_1) \dots r(w_k)}$$

The recognized tree series is

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

Semantics — Illustration (without weights)

- ▶ Sample input tree:



- ▶ Unsuccessful run:



- ▶ Successful run:



Semantics — Illustration (without weights)

- ▶ Sample input tree:



- ▶ Unsuccessful run:



- ▶ Successful run:



Semantics — Illustration (without weights)

- ▶ Sample input tree:



- ▶ Unsuccessful run:



- ▶ Successful run:



Semantics — Illustration (without weights)

- ▶ Sample input tree:



- ▶ Unsuccessful run:



- ▶ Successful run:



Table of Contents

Weighted Tree Automata

Bisimulation Minimization

Minimization of Deterministic Devices

Overview

Application

- ▶ Minimize nondeterministic wta
- ▶ Minimize unweighted deterministic tree automata
- ▶ Very efficient

Types

- ▶ Based on [3]
- ▶ *Forward*: approx. 10% reduction, useful on det. devices
- ▶ *Backward*: approx. 40% reduction

Forward Bisimulation

Definition

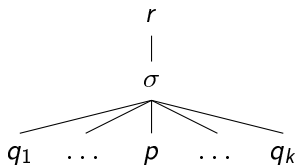
A *forward bisimulation* \mathcal{R} is an equivalence relation such that for every $(p, q) \in \mathcal{R}$

- ▶ $p \in F$ if and only if $q \in F$
- ▶ for every $\sigma \in \Sigma_k$, $q_1, \dots, q_k \in Q$, $i \in [k]$, and $D \in Q/\mathcal{R}$

$$\sum_{r \in D} \mu_k(\sigma)_{r, q_1 \dots q_{i-1} p q_{i+1} \dots q_k} = \sum_{r \in D} \mu_k(\sigma)_{r, q_1 \dots q_{i-1} q q_{i+1} \dots q_k}$$

Unweighted Case

If there exists a transition



Forward Bisimulation

Definition

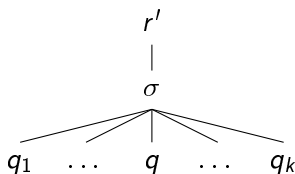
A *forward bisimulation* \mathcal{R} is an equivalence relation such that for every $(p, q) \in \mathcal{R}$

- ▶ $p \in F$ if and only if $q \in F$
- ▶ for every $\sigma \in \Sigma_k$, $q_1, \dots, q_k \in Q$, $i \in [k]$, and $D \in Q/\mathcal{R}$

$$\sum_{r \in D} \mu_k(\sigma)_{r, q_1 \dots q_{i-1} p q_{i+1} \dots q_k} = \sum_{r \in D} \mu_k(\sigma)_{r, q_1 \dots q_{i-1} q q_{i+1} \dots q_k}$$

Unweighted Case

then there exists $r' \in [r]$ and



Forward Bisimulation Minimization

```
[input:   a wta  $M = (Q, \Sigma, \mathcal{A}, F, \mu)$ ;  
initially:  
   $\mathcal{P}_0 := Q^2$ ;  
   $\mathcal{R}_0 := ((Q \setminus F)^2 \cup F^2) \setminus \text{split}(Q)$ ;  
   $i := 0$ ;  
while  $\mathcal{R}_i \neq \mathcal{P}_i$ :  
  choose  $S_i \in Q/\mathcal{P}_i$  and  $B_i \in Q/\mathcal{R}_i$  such that  
     $B_i \subset S_i$  and  $\text{card}(B_i) \leq \text{card}(S_i)/2$ ;  
   $\mathcal{P}_{i+1} := \mathcal{P}_i \setminus \text{cut}(B_i)$ ;  
   $\mathcal{R}_{i+1} := (\mathcal{R}_i \setminus \text{split}(B_i)) \setminus \text{split}(S_i, B_i)$ ;  
   $i := i + 1$ ;  
return:  the wta  $M/\mathcal{R}_i$ ;
```

Figure: A minimisation algorithm based on forward bisimulation

Forward Bisimulation

Notes

- ▶ Effective on det. wta
- ▶ Very fast $O(rm \log n)$
- ▶ Approx. 10% reduction in test set

Backward Bisimulation

Definition

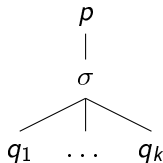
A *backward bisimulation* \mathcal{R} is an equivalence relation such that for every $(p, q) \in \mathcal{R}$

- ▶ for every $\sigma \in \Sigma_k$ and $D_1, \dots, D_k \in Q/\mathcal{R}$

$$\sum_{w \in D_1 \dots D_k} \mu_k(\sigma)_{p,w} = \sum_{w \in D_1 \dots D_k} \mu_k(\sigma)_{q,w}$$

Unweighted Case

If there exists a transition



Backward Bisimulation

Definition

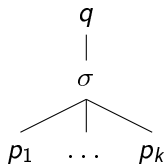
A *backward bisimulation* \mathcal{R} is an equivalence relation such that for every $(p, q) \in \mathcal{R}$

- ▶ for every $\sigma \in \Sigma_k$ and $D_1, \dots, D_k \in Q/\mathcal{R}$

$$\sum_{w \in D_1 \dots D_k} \mu_k(\sigma)_{p,w} = \sum_{w \in D_1 \dots D_k} \mu_k(\sigma)_{q,w}$$

Unweighted Case

then there exists $p_1 \in [q_1], \dots, p_k \in [q_k]$ and



Backward Bisimulation

Notes

- ▶ Generalization of [1]
- ▶ Ineffective on det. wta
- ▶ fast $O(r^2 m \log n)$
- ▶ Approx. 40% reduction in test set

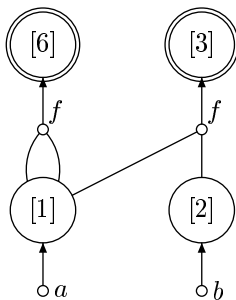


Figure: Reduced wta

Results

TREES	ORIG.	BWD	FWD	AKH	FWD, BWD	BWD, FWD
58	353	252	286	353	185	180
161	953	576	749	953	378	356
231	1373	781	1075	1373	494	468
287	1726	947	1358	1726	595	563

Table: State set reduction after minimisation

Funny Fact

- ▶ Jonathan's implementation (PERL) took < 1 second to run all tests
- ▶ My verification implementation (HASKELL) took > 4 hours for the last example alone

Table of Contents

Weighted Tree Automata

Bisimulation Minimization

Minimization of Deterministic Devices

Deterministic wta

Definition

Wta is *deterministic* if for every $\sigma \in \Sigma_k$ and $w \in Q^k$ there exists at most one $q \in Q$ such that $\mu_k(\sigma)_{q,w} \neq 0$.

Notes

- ▶ Determinization is possible in locally-finite semirings [2]
- ▶ Partial determinization for probabilities [4]

Myhill-Nerode Congruence

Definition

$t \equiv u$ if there exist $a, b \in A \setminus \{0\}$ such that for every context C

$$a \cdot (\|M\|, C[t]) = b \cdot (\|M\|, C[u])$$

Approach


- ▶ Identify dead states (those do not contribute at all)
- ▶ Move from trees to states; define congruence on states
- ▶ Compute the coarsest equivalence on the states subject to the above condition

Minimization of deterministic devices

Notes

- ▶ No real tests yet!
- ▶ Rather efficient $O(rmn^4)$, which can certainly be improved
- ▶ My (!!) implementation takes reasonable time on examples

References

-  P. A. Abdulla, J. Högberg, and L. Kaati.
Bisimulation minimization of tree automata.
Int. J. Foundations of Computer Science, 18(4):699–713, 2007.
-  B. Borchardt and H. Vogler.
Determinization of finite state weighted tree automata.
Journal of Automata, Languages and Combinatorics,
8(3):417–463, 2003.
-  P. Buchholz.
Bisimulation relations for weighted automata.
Theoretical Computer Science, 2008.
to appear.
-  Jonathan May and Kevin Knight.
A better n-best list: Practical determinization of weighted
finite tree automata.
In *HLT-NAACL*, 2006.