Minimization of Non-deterministic Weighted Tree Automata

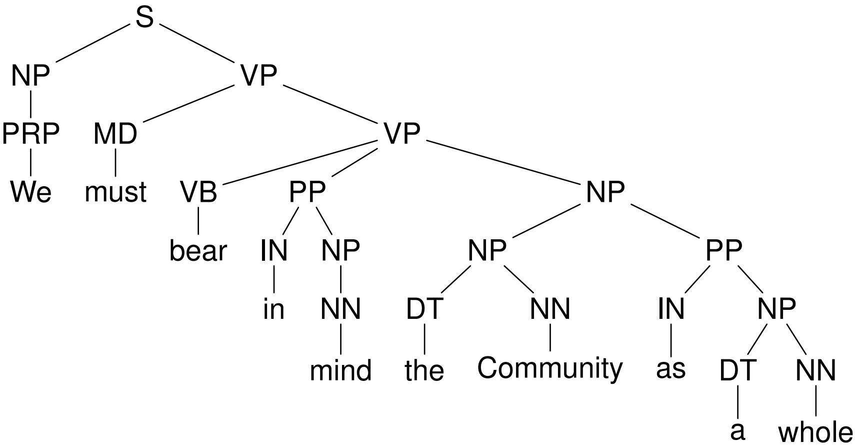
Andreas Maletti

Institute for Natural Language Processing
Universität Stuttgart, Germany

maletti@ims.uni-stuttgart.de

Szeged — December 11, 2012

Parse Trees


S
NP
VP
PRP
MD
VP
We
must
VB
PP
NP
bear
IN
NP
NP
PP
in
NN
DT
NN
IN
NP
mind
the
Community
as
DT
NN
a
whole

Parse Forest of a CFG

Example

$$\text{S} \to \text{NP VP} \qquad \text{VP} \to \text{MD VP}$$
$$\text{NP} \to \text{NP PP} \qquad \text{VP} \to \text{VB PP NP}$$
$$\text{MD} \to \text{must} \qquad \dots$$

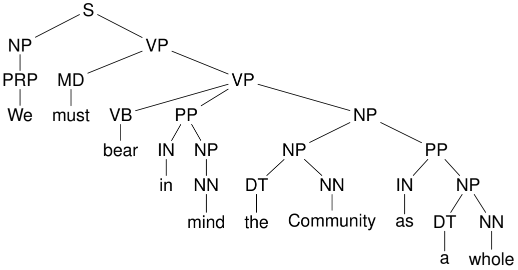
Parse Forest of a CFG

Example

$$\text{S} \to \text{NP VP} \qquad \text{VP} \to \text{MD VP}$$

$$\text{NP} \to \text{NP PP} \qquad \text{VP} \to \text{VB PP NP}$$

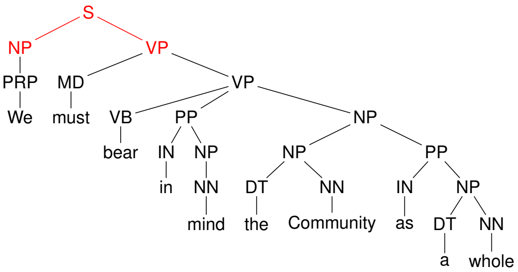
$$\text{MD} \to \text{must} \qquad \dots$$



Parse Forest of a CFG

Example

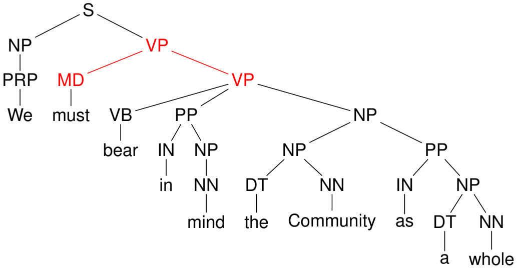
$$\text{S} \to \text{NP VP} \qquad \text{VP} \to \text{MD VP}$$
$$\text{NP} \to \text{NP PP} \qquad \text{VP} \to \text{VB PP NP}$$
$$\text{MD} \to \text{must} \qquad \dots$$



Parse Forest of a CFG

Example

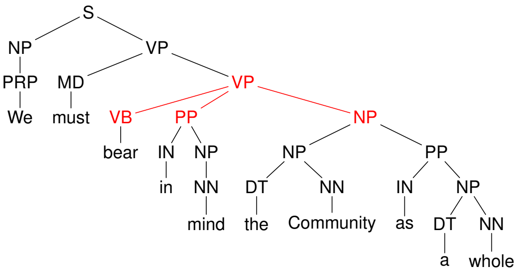
$$S \to NP\ VP \qquad VP \to MD\ VP$$
$$NP \to NP\ PP \qquad VP \to VB\ PP\ NP$$
$$MD \to must \qquad \ldots$$



Parse Forest of a CFG

Example

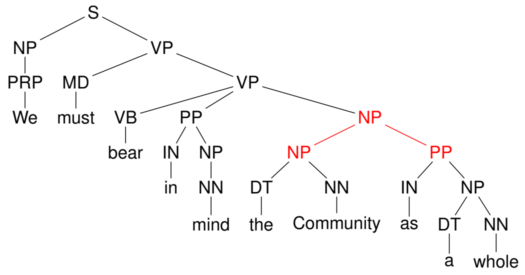
$$S \to NP\ VP \qquad VP \to MD\ VP$$
$$NP \to NP\ PP \qquad VP \to VB\ PP\ NP$$
$$MD \to must \qquad \ldots$$



Parse Forest of a CFG

Example

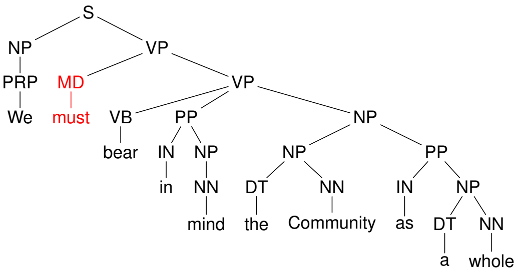
$$S \to NP\ VP \qquad VP \to MD\ VP$$
$$NP \to NP\ PP \qquad VP \to VB\ PP\ NP$$
$$MD \to must \qquad \ldots$$



Parse Forest of a CFG

Example

$$S \to NP\ VP \qquad VP \to MD\ VP$$
$$NP \to NP\ PP \qquad VP \to VB\ PP\ NP$$
$$MD \to must \qquad \ldots$$



Local Tree Grammar

Definition (Gécseg, Steinby 1984)

A local tree grammar G is a finite set of CFG productions (together with a start nonterminal S)

Definition (Generated tree language)

$L(G)$ contains exactly the trees in which

- the root is labeled S
- "label $\to$ child labels" is a production of G for each internal node

Local Tree Grammar

Definition (Gécseg, Steinby 1984)

A local tree grammar G is a finite set of CFG productions (together with a start nonterminal S)

Definition (Generated tree language)

$L(G)$ contains exactly the trees in which

- the root is labeled S
- "label $\to$ child labels" is a production of G for each internal node

Local Tree Grammar

Theorem

Local tree grammars recognize exactly the parse forests of CFG

Properties

- ✓ simple
- ✓ minimality trivial
- ✓ no ambiguity (unique explanation for each recognized tree)
- ✗ not closed under BOOLEAN operations (union/intersection/complement: ✗/✓/✗)
- ✗ not closed under (non-injective) relabelings
- ✗ ...

Local Tree Grammar

Theorem

Local tree grammars recognize exactly the parse forests of CFG

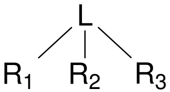
Properties

- ✓ simple
- ✓ minimality trivial
- ✓ no ambiguity (unique explanation for each recognized tree)
- ✗ not closed under BOOLEAN operations (union/intersection/complement: ✗/✓/✗)
- ✗ not closed under (non-injective) relabelings
- ✗ …

Local Tree Grammar

Generalization

- CFG production $L \rightarrow R_1\ R_2\ R_3$ represented by tree



- "Glue" fragments together to obtain larger trees:

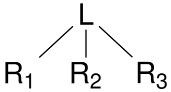
S

- But why only small tree fragments?

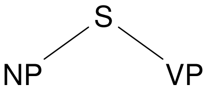
Local Tree Grammar

Generalization

- CFG production $\mathrm{L} \rightarrow \mathrm{R}_1\ \mathrm{R}_2\ \mathrm{R}_3$ represented by tree



- “Glue” fragments together to obtain larger trees:

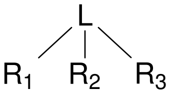


- But why only small tree fragments?

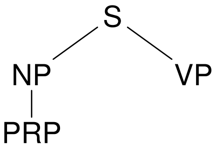
Local Tree Grammar

Generalization

- CFG production $L \to R_1\ R_2\ R_3$ represented by tree



- “Glue” fragments together to obtain larger trees:

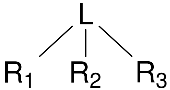


- But why only small tree fragments?

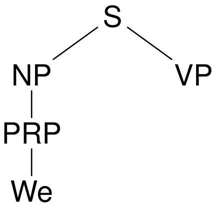
Local Tree Grammar

Generalization

- CFG production $L \to R_1\ R_2\ R_3$ represented by tree



- “Glue” fragments together to obtain larger trees:

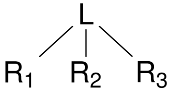


- But why only small tree fragments?

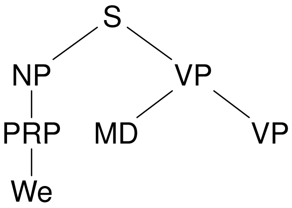
Local Tree Grammar

Generalization

- CFG production $L \to R_1\ R_2\ R_3$ represented by tree



- "Glue" fragments together to obtain larger trees:

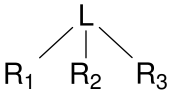


- But why only small tree fragments?

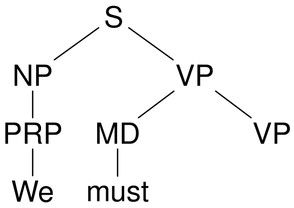
Local Tree Grammar

Generalization

- CFG production $L \to R_1\ R_2\ R_3$ represented by tree



- “Glue” fragments together to obtain larger trees:

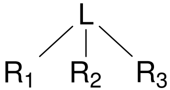


- But why only small tree fragments?

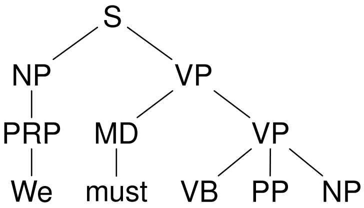
Local Tree Grammar

Generalization

- CFG production $L \to R_1\ R_2\ R_3$ represented by tree



- "Glue" fragments together to obtain larger trees:

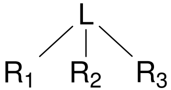


- But why only small tree fragments?

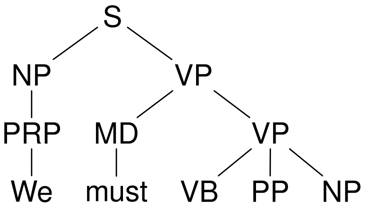
Local Tree Grammar

Generalization

- CFG production $L \to R_1\ R_2\ R_3$ represented by tree



- "Glue" fragments together to obtain larger trees:



- But why only small tree fragments?

Tree Substitution Grammar

Definition

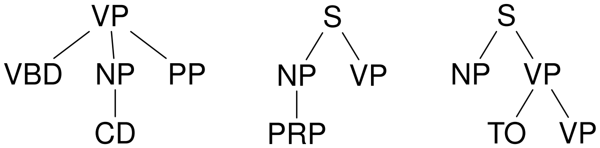
A tree substitution grammar is a finite set of tree fragments (together with a start nonterminal S)

Tree Substitution Grammar

Definition

A tree substitution grammar is a finite set of tree fragments (together with a start nonterminal S)

Example (Typical fragments [Post, ACL 2011])



Tree Substitution Grammar

Properties

- ✓ simple
- ✓ more expressive than local tree grammars
- ✗ minimality not simple
- ✗ ambiguity (several explanations for a recognized tree)
- ✗ not closed under BOOLEAN operations (union/intersection/complement: ✗/✗/✗)
- ✗ not closed under (non-injective) relabelings
- ✗ . . .

Tree Substitution Grammar

Experiment [Post, Gildea, ACL 2009]

grammar	size	Prec.	Recall	F1
CFG	46k	75.37	70.05	72.61
“spinal” TSG	190k	80.30	78.10	79.18
“minimal subset” TSG	2,560k	76.40	78.29	77.33

(on WSJ Sect. 23)

Tree Substitution Grammar with Latent Variables

Definition (SHINDO et al., ACL 2012 best paper)

A tree substitution grammar with latent variables is
a tree substitution grammar together with a functional relabeling

Remark

Typically symbols that are relabeled to X are written as X-*n*

Tree Substitution Grammar with Latent Variables

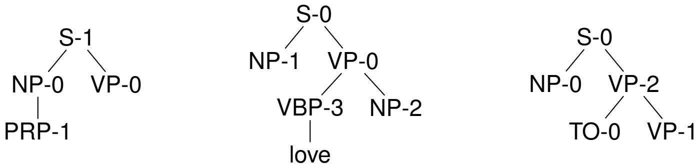
Definition (Shindo et al., ACL 2012 best paper)

A tree substitution grammar with latent variables is
a tree substitution grammar together with a functional relabeling

Remark

Typically symbols that are relabeled to X are written as X-n

Example (Typical fragments)



Tree Substitution Grammar with Latent Variables

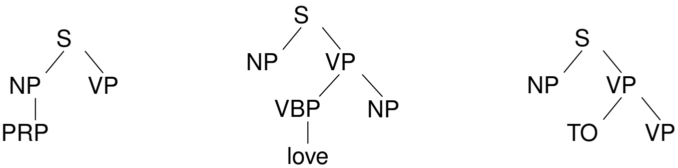
Definition (Shindo et al., ACL 2012 best paper)

A tree substitution grammar with latent variables is
a tree substitution grammar together with a functional relabeling

Remark

Typically symbols that are relabeled to X are written as X-*n*

Example (Typical fragments)



Tree Substitution Grammar with Latent Variables

Experiment [Shindo et al., ACL 2012 best paper]

grammar	F1 score $\lvert w \rvert \leq 40$	F1 score full
TSG [Post, Gildea, 2009]	82.6	
TSG [Cohn et al., 2010]	85.4	84.7
CFGlv [Collins, 1999]	88.6	88.2
CFGlv [Petrov, Klein, 2007]	90.6	90.1
CFGlv [Petrov, 2010]		91.8
TSGlv (single)	91.6	91.1
TSGlv (multiple)	92.9	92.4
Discriminative Parsers		
Carreras et al., 2008		91.1
Charniak, Johnson, 2005	92.0	91.4
Huang, 2008	92.3	91.7

Overview

Regular Tree Grammar

Definition (Brainerd, 1969)

A regular tree grammar is a tuple $G = (Q, \Sigma, I, P)$ with

- alphabet of nonterminals Q
- alphabet of terminals Σ
- initial nonterminals $I \subseteq Q$
- finite set of productions $P \subseteq Q \times T_\Sigma(Q)$

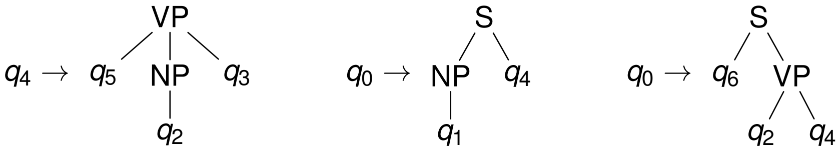
Remark

Instead of (q, r) we write $q \to r$

Regular Tree Grammar

Example

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$
- $\Sigma = \{\mathrm{VP}, \mathrm{NP}, \mathrm{S}, \dots\}$
- $I = \{q_0\}$
- and the following productions:



Regular Tree Grammar

Definition (Derivation Semantics)

Sentential forms: $t, u \in T_\Sigma(Q)$

$$t \Rightarrow_G u$$

if there exist position $w \in \text{pos}(t)$ and production $q \to r \in P$

- $t = t[q]_w$
- $u = t[r]_w$

Definition (Recognized tree language)

$$L(G) = \{t \in T_\Sigma \mid \exists q \in I \colon q \Rightarrow_G^* t\}$$

Regular Tree Grammar

Definition (Derivation Semantics)

Sentential forms: $t, u \in T_\Sigma(Q)$

$$t \Rightarrow_G u$$

if there exist position $w \in \text{pos}(t)$ and production $q \to r \in P$

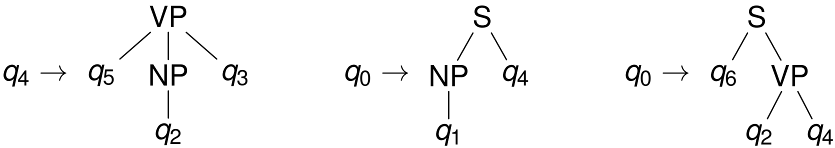
- $t = t[q]_w$
- $u = t[r]_w$

Definition (Recognized tree language)

$$L(G) = \{t \in T_\Sigma \mid \exists q \in I \colon q \Rightarrow_G^* t\}$$

Regular Tree Grammar

Example (Productions)

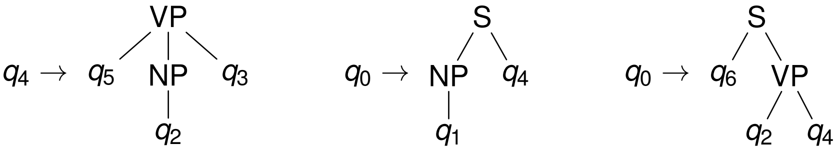


Example (Derivation)

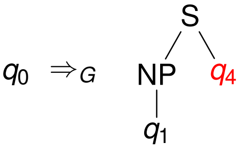
q_0

Regular Tree Grammar

Example (Productions)

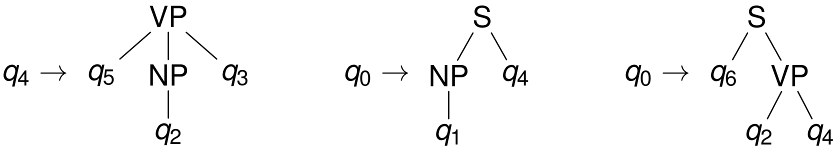


Example (Derivation)

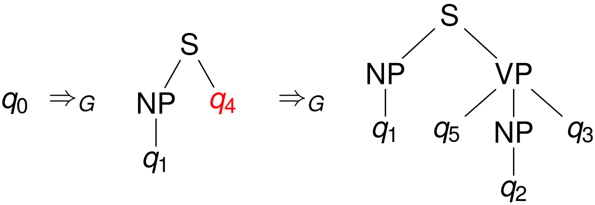


Regular Tree Grammar

Example (Productions)



Example (Derivation)



Regular Tree Grammar

Properties

- ✓ simple
- ✓ more expressive than tree substitution grammars
- ✗ minimization difficult
- ✗ ambiguity (several explanations for a recognized tree)
- ✓ closed under all BOOLEAN operations (union/intersection/complement: ✓/✓/✓)
- ✓ closed under (non-injective) relabelings
- ✓ ...

Regular Tree Grammar

Definition (BRAINERD, 1969)

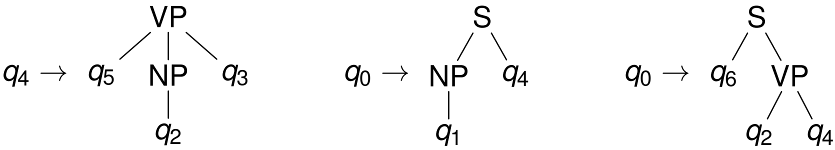
G is in normal form if $r = \sigma(q_1, \dots, q_k)$ with $\sigma \in \Sigma$ and $q_1, \dots, q_k \in Q$ for all $q \to r \in P$

Regular Tree Grammar

Definition (Brainerd, 1969)

G is in normal form if $r = \sigma(q_1, \ldots, q_k)$ with $\sigma \in \Sigma$ and $q_1, \ldots, q_k \in Q$ for all $q \to r \in P$

Example (Productions)



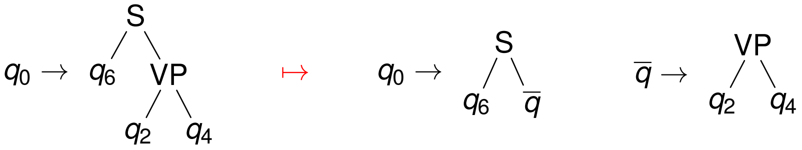
Regular Tree Grammar

Theorem (Brainerd, 1969)

Any G is equivalent to a regular tree grammar in normal form

Proof.

Simply cut large rules introducing new states

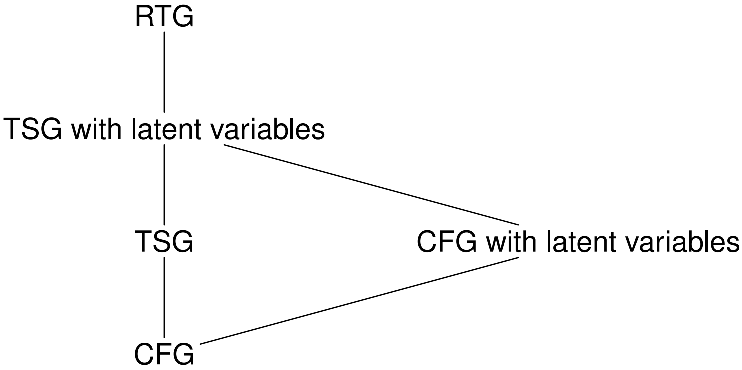


□

Regular Tree Grammar

Theorem (Folk, Lore, 1972)

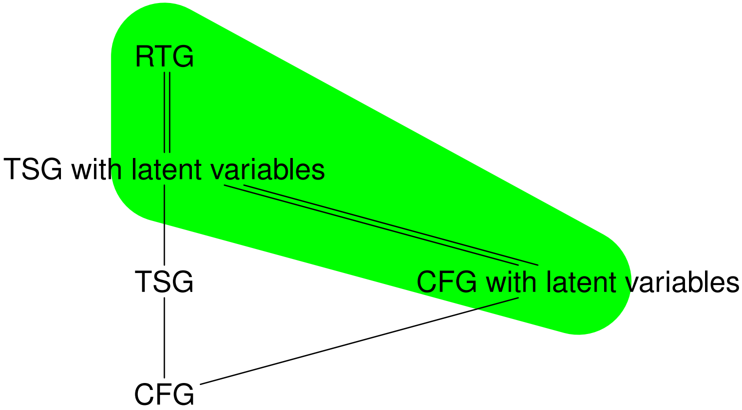
regular tree languages = *relabeled local tree languages*



Regular Tree Grammar

Theorem (Folk, Lore, 1972)

regular tree languages = relabeled local tree languages



Berkeley Parser

Example (Berkeley parser — English grammar)

S-1 $\rightarrow$ ADJP-2 S-1	$0.0035453455987323125 \cdot 10^{0}$
S-1 $\rightarrow$ ADJP-1 S-1	$2.108608433271444 \cdot 10^{-6}$
S-1 $\rightarrow$ VP-5 VP-3	$1.6367163259885093 \cdot 10^{-4}$
S-2 $\rightarrow$ VP-5 VP-3	$9.724998692152419 \cdot 10^{-8}$
S-1 $\rightarrow$ PP-7 VP-0	$1.0686659961009547 \cdot 10^{-5}$
S-9 $\rightarrow$ “ NP-3	$0.012551243773149695 \cdot 10^{0}$

$\rightsquigarrow$ Regular tree grammar

Overview

Tree Automaton

Definition (THATCHER, 1970; ROUNDS, 1970)

tree automaton is a regular tree grammar in normal form

Definition

A tree automaton is minimal in $\mathcal{C}$
if all equivalent tree automata of $\mathcal{C}$ are at least as large

Theorem

Complexity of minimization problems:

outp. \ inp. model	DTA	NTA
$\mathcal{C} =$ DTA	NL	(EXPTIME)
$\mathcal{C} =$ NTA	PSPACE	PSPACE

Weighted Tree Automaton

Definition (BERSTEL, REUTENAUER, 1982)

A weighted tree automaton is a tree automaton together with a map $c\colon P \to S$

Semantics

- S forms a semiring $(S, +, \cdot, 0, 1)$
- production weights are multiplied $(\cdot)$ in a derivation
- weights of multiple (left-most) derivations for the same tree are summed $(+)$

Weighted Tree Automaton

Definition (Berstel, Reutenauer, 1982)

A weighted tree automaton is a tree automaton together with a map $c\colon P \to S$

Semantics

- S forms a semiring $(S, +, \cdot, 0, 1)$
- production weights are multiplied $(\cdot)$ in a derivation
- weights of multiple (left-most) derivations for the same tree are summed $(+)$

Weighted Tree Automaton

Definition (Berstel, Reutenauer, 1982)

A weighted tree automaton is a tree automaton together with a map $c\colon P \to S$

Semantics

- S forms a semiring $(S, +, \cdot, 0, 1)$
- production weights are multiplied $(\cdot)$ in a derivation
- weights of multiple (left-most) derivations for the same tree are summed $(+)$

Weighted Tree Automaton

Definition (Berstel, Reutenauer, 1982)

A weighted tree automaton is a tree automaton together with a map $c\colon P \to S$

Semantics

- S forms a semiring $(S, +, \cdot, 0, 1)$
- production weights are multiplied $(\cdot)$ in a derivation
- weights of multiple (left-most) derivations for the same tree are summed $(+)$

Weighted Tree Automaton

Definition

Recursive bottom-up semantics: $c\colon T_\Sigma(Q) \to S^Q$

$$c(p)_q = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{otherwise} \end{cases}$$

$$c(\sigma(t_1, \ldots, t_k))_q = \sum_{q_1, \ldots, q_k \in Q} c(q \to \sigma(q_1, \ldots, q_k)) \cdot \prod_{1 \leq i \leq k} c(t_i)_{q_i}$$

for all $\sigma \in \Sigma$, $p, q \in Q$, and $t_1, \ldots, t_k \in T_\Sigma$

Theorem

recursive bottom-up semantics $=$ derivation semantics

Weighted Tree Automaton

Definition

Recursive bottom-up semantics: $c \colon T_\Sigma(Q) \to S^Q$

$$c(p)_q = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{otherwise} \end{cases}$$

$$c(\sigma(t_1, \dots, t_k))_q = \sum_{q_1, \dots, q_k \in Q} c(q \to \sigma(q_1, \dots, q_k)) \cdot \prod_{1 \leq i \leq k} c(t_i)_{q_i}$$

for all $\sigma \in \Sigma$, $p, q \in Q$, and $t_1, \dots, t_k \in T_\Sigma$

Theorem

recursive bottom-up semantics $=$ derivation semantics

Weighted Tree Automaton

Remarks

- BERKELEY parser uses weighted tree automata
- but has a best-derivation semantics VITERBI parse

Theoretical research

- Minimization wrt. best-derivation semantics
- Minimization wrt. n-best-derivation semantics
- Foundational investigation of those semantics

Weighted Tree Automaton

Remarks

- BERKELEY parser uses weighted tree automata
- but has a best-derivation semantics VITERBI parse

Theoretical research

- Minimization wrt. best-derivation semantics
- Minimization wrt. *n*-best-derivation semantics
- Foundational investigation of those semantics

Weighted Tree Automaton

Remarks

- BERKELEY parser uses weighted tree automata
- but has a best-derivation semantics — VITERBI parse

Theoretical research

- Minimization wrt. best-derivation semantics
- Minimization wrt. *n*-best-derivation semantics
- Foundational investigation of those semantics

Weighted Tree Automaton

Remarks

- BERKELEY parser uses weighted tree automata
- but has a best-derivation semantics VITERBI parse

Theoretical research

- Minimization wrt. best-derivation semantics
- Minimization wrt. *n*-best-derivation semantics
- Foundational investigation of those semantics

Weighted Tree Automaton

Remarks

- BERKELEY parser uses weighted tree automata
- but has a best-derivation semantics VITERBI parse

Theoretical research

- Minimization wrt. best-derivation semantics
- Minimization wrt. *n*-best-derivation semantics
- Foundational investigation of those semantics

Weighted Tree Automaton

Method	Complexity	Reference
Forw. Bisimulation	$\mathcal{O}(rm \log n)$	Högberg, M., May 2007
Backw. Bisimulation	$\mathcal{O}(r^2 m \log n)$	Högberg, M., May 2007
Backw. Simulation	$\mathcal{O}(r^2 mn)$	Abdulla et al 2008
Full minimization	P	Bozapalidis, Seidl 1991

Notation

- m: number of transitions
- n: number of states
- r: maximal rank of the input symbols

Weighted Tree Automaton

Definition (Forward bisimulation)

Equivalence relation $\sim$ on states such that

- $p \in I \iff p' \in I$
- for every symbol $\sigma \in \Sigma$ and states q and ...

$$\sum_{r \in [q]} c(r \to \sigma(\dots, p, \dots)) = \sum_{r \in [q]} c(r \to \sigma(\dots, p', \dots))$$

for every $p \sim p'$

Minimization

Reduction of G by coarsest forward bisimulation $\cong$ on G $\qquad G/{\cong}$

Weighted Tree Automaton

Definition (Forward bisimulation)

Equivalence relation $\sim$ on states such that

- $p \in I \iff p' \in I$
- for every symbol $\sigma \in \Sigma$ and states q and ...

$$\sum_{r \in [q]} c(r \to \sigma(\dots, p, \dots)) = \sum_{r \in [q]} c(r \to \sigma(\dots, p', \dots))$$

for every $p \sim p'$

Minimization

Reduction of G by coarsest forward bisimulation $\cong$ on G $G/{\cong}$

Weighted Tree Automaton

Definition (Backward bisimulation)

Equivalence relation $\sim$ on states such that

$$\sum_{q_1 \cdots q_k \in B_1 \cdots B_k} c(p \to \sigma(q_1, \ldots, q_k)) = \sum_{q_1 \cdots q_k \in B_1 \cdots B_k} c(p' \to \sigma(q_1, \ldots, q_k))$$

for every $p \sim p'$, symbol $\sigma \in \Sigma$, and blocks $B_1, \ldots, B_k$

Minimization

Reduction of G by coarsest backward bisimulation $\cong$ on G $G/{\cong}$

Weighted Tree Automaton

Definition (Backward bisimulation)

Equivalence relation $\sim$ on states such that

$$\sum_{q_1 \cdots q_k \in B_1 \cdots B_k} c(p \to \sigma(q_1, \ldots, q_k)) = \sum_{q_1 \cdots q_k \in B_1 \cdots B_k} c(p' \to \sigma(q_1, \ldots, q_k))$$

for every $p \sim p'$, symbol $\sigma \in \Sigma$, and blocks $B_1, \ldots, B_k$

Minimization

Reduction of G by coarsest backward bisimulation $\cong$ on G $\qquad G/\cong$

Bisimulation Minimization

(needs additive cancellation)

Experiment with BERKELEY parser

	states		productions	
English grammar	1,133	100%	1,842,218	100%
backward minimal	548	48%	626,600	34%
forward minimal	791	70%	767,153	42%
backward/forward minimal	366	32%	272,675	15%
forward/backward minimal	381	34%	309,845	17%

The results for forward are buggy

Bisimulation Minimization

(needs additive cancellation)

Experiment with BERKELEY parser

	states		productions	
English grammar	1,133	100%	1,842,218	100%
backward minimal	548	48%	626,600	34%
forward minimal	791	70%	767,153	42%
backward/forward minimal	366	32%	272,675	15%
forward/backward minimal	381	34%	309,845	17%

The results for forward are buggy

Weighted Tree Automaton

Definition (Context)

- context is a tree with exactly one occurrence of special symbol $\square$
- C_Σ contains all contexts over Σ
- $u[t]$ tree obtained from context u by replacing $\square$ by tree t

Definition (Extension of c)

Extend the function c to $c' \colon S^Q \to S^{C_\Sigma}$ such that

$$c'(f)_u = \sum_{\substack{p \in I \\ q \in Q}} f(q) \cdot c(u[q])_p$$

for all $f \in S^Q$ and $u \in C_\Sigma$

Weighted Tree Automaton

Properties

- wTA G forms an S-Σ-algebra
- $\{c'(c(t)) \mid t \in T_\Sigma\}$ yields the minimal representation
- by the first isomorphism theorem this image is ismorphic to the quotient with respect to $\ker(c')$

$\rightsquigarrow$ compute $B = \{\varphi \in c(T_\Sigma) \mid c'(\varphi) = \vec{0}\}$

Computation

- compute a "small" basis $\langle c(t_1), \dots, c(t_k) \rangle$ of G
- compute approximations

$$B_i = \{\varphi \in c(T_\Sigma) \mid c'(\varphi)_u = 0 \text{ for all small } u \in C_\Sigma\}$$

- $B = \bigcap_{i \in \mathbb{N}} B_i$

Weighted Tree Automaton

Properties

- wTA G forms an S-Σ-algebra
- $\{c'(c(t)) \mid t \in T_\Sigma\}$ yields the minimal representation
- by the first isomorphism theorem this image is ismorphic to the quotient with respect to $\ker(c')$

$\rightsquigarrow$ compute $B = \{\varphi \in c(T_\Sigma) \mid c'(\varphi) = \vec{0}\}$

Computation

- compute a "small" basis $\langle c(t_1), \dots, c(t_k)\rangle$ of G
- compute approximations

$$B_i = \{\varphi \in c(T_\Sigma) \mid c'(\varphi)_u = 0 \text{ for all small } u \in C_\Sigma\}$$

- $B = \bigcap_{i \in \mathbb{N}} B_i$

Weighted Tree Automaton

Theorem (BOZAPALIDIS 1991 and SEIDL 1992)

Weighted tree automata over fields can effectively be minimized

Remarks

- even smaller than bisimulation-minimal wTA
- implementations for weighted string automata are efficient
- no implementation for wTA yet

Weighted Tree Automaton

Method	Complexity	Reference
Forw. Bisimulation	$\mathcal{O}(rm \log n)$	Högberg, M., May 2007
Backw. Bisimulation	$\mathcal{O}(r^2 m \log n)$	Högberg, M., May 2007
Backw. Simulation	$\mathcal{O}(r^2 mn)$	Abdulla et al 2008
Full minimization	P	Bozapalidis, Seidl 1991

	states		productions	
English grammar	1,133	100%	1,842,218	100%
backward minimal	548	48%	626,600	34%