Applications of Tree Automata Theory
Lecture I: Tree Automata

Andreas Maletti

Institute of Computer Science
Universität Leipzig, Germany

on leave from: Institute for Natural Language Processing
Universität Stuttgart, Germany

maletti@ims.uni-stuttgart.de

Yekaterinburg — August 23, 2014

Roadmap

1. Theory of Tree Automata
2. Parsing — Basics and Evaluation
3. Parsing — Advanced Topics
4. Machine Translation — Basics and Evaluation
5. Theory of Tree Transducers
6. Machine Translation — Advanced Topics

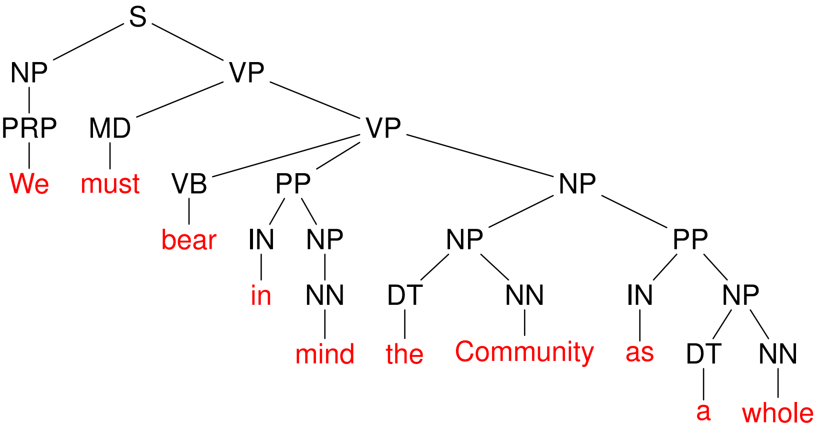
Always ask questions right away!

Trees

Motivation and Notation

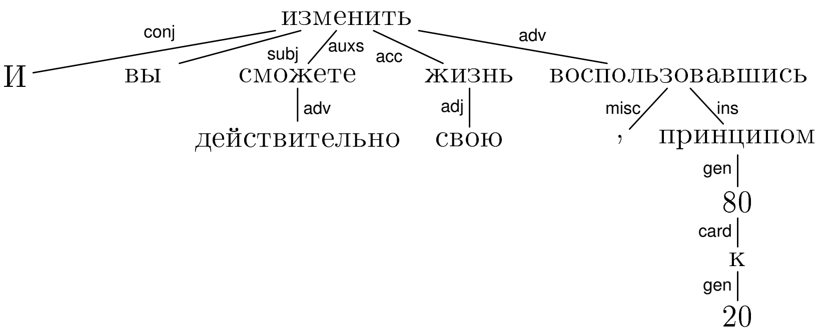
Trees — Parses

We must bear in mind the Community as a whole



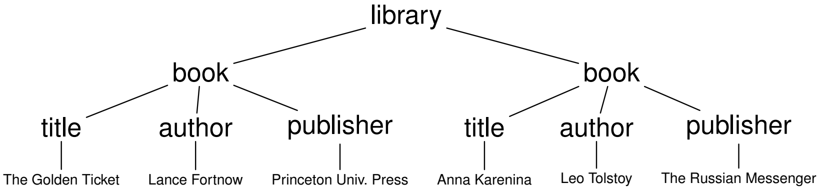
Trees — Parses

И вы действительно сможете изменить свою жизнь, воспользовавшись принципом 80 к 20



Trees — XML

```
<library>
 <book>
  <title>The Golden Ticket</title>
  <author>Lance Fortnow</author>
  <publisher>Princeton Univ. Press</publisher>
 </book><book>
  <title>Anna Karenina</title>
  <author>Leo Tolstoy</author>
  <publisher>The Russian Messenger</publisher>
 </book>
</library>
```



Trees

Sets Σ and Q

Definition

Set $T_\Sigma(Q)$ of Σ-trees indexed by Q is smallest T

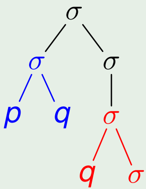
- $q \in T$ for all $q \in Q$
- $\sigma(t_1, \ldots, t_k) \in T$ for all $k \in \mathbb{N}$, $\sigma \in \Sigma$, and $t_1, \ldots, t_k \in T$

We assume $\Sigma \cap Q = \emptyset$ and write σ instead of $\sigma()$

Trees

Example

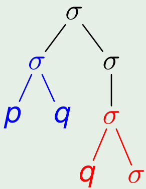
- $\Sigma = \{\sigma, \alpha\}$ and $Q = \{q, p\}$
- $\sigma(\sigma(p, q), \sigma(\sigma(q, \sigma))) \in T_\Sigma(Q)$



Trees

Example

- $\Sigma = \{\sigma, \alpha\}$ and $Q = \{q, p\}$
- $\sigma(\sigma(p, q), \sigma(\sigma(q, \sigma))) \in T_\Sigma(Q)$



Notes

- obvious recursion & induction principle

Trees — Recursion

Definition (Gorn address)

Mapping $\text{pos}\colon T_\Sigma(Q) \to 2^{\mathbb{N}^*}$ assigning positions

$$\text{pos}(q) = \{\varepsilon\}$$

$$\text{pos}(\sigma(t_1, \ldots, t_k)) = \{\varepsilon\} \cup \{i.w \mid 1 \leq i \leq k,\ w \in \text{pos}(t_i)\}$$

Trees — Recursion

Definition (Gorn address)

Mapping $\text{pos}\colon T_\Sigma(Q) \to 2^{\mathbb{N}^*}$ assigning positions

$$\text{pos}(q) = \{\varepsilon\}$$

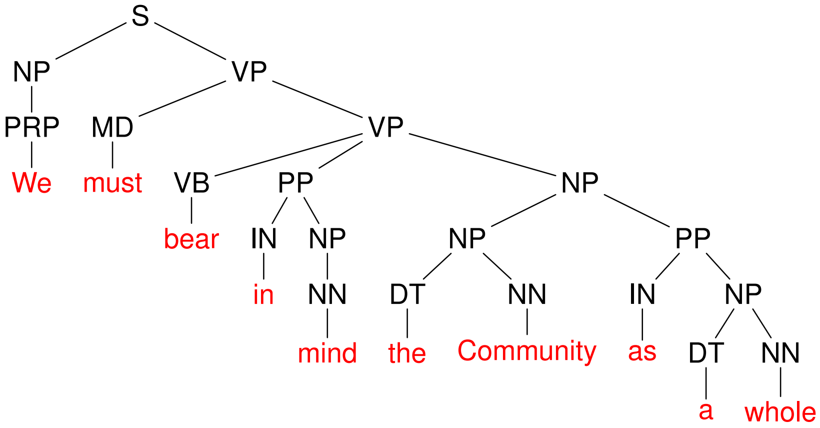
$$\text{pos}(\sigma(t_1, \dots, t_k)) = \{\varepsilon\} \cup \{i.w \mid 1 \leq i \leq k,\ w \in \text{pos}(t_i)\}$$

Definition

Leaves of $t \in T_\Sigma(Q)$:

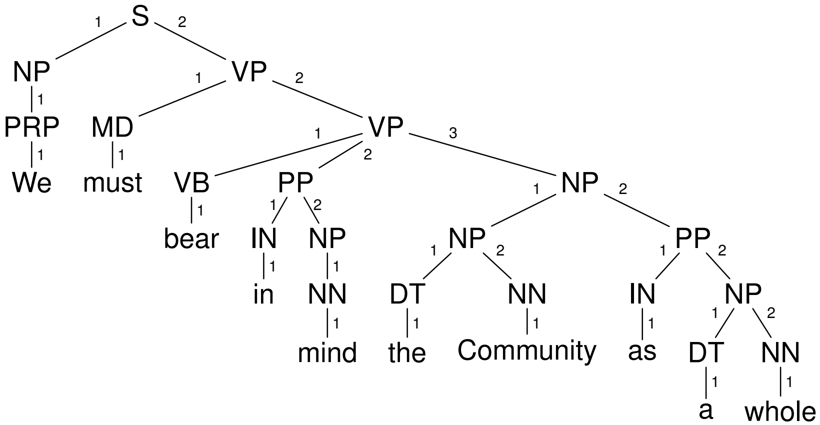
$$\text{leaves}(t) = \{w \in \text{pos}(t) \mid w.1 \notin \text{pos}(t)\}$$

Trees — Recursion

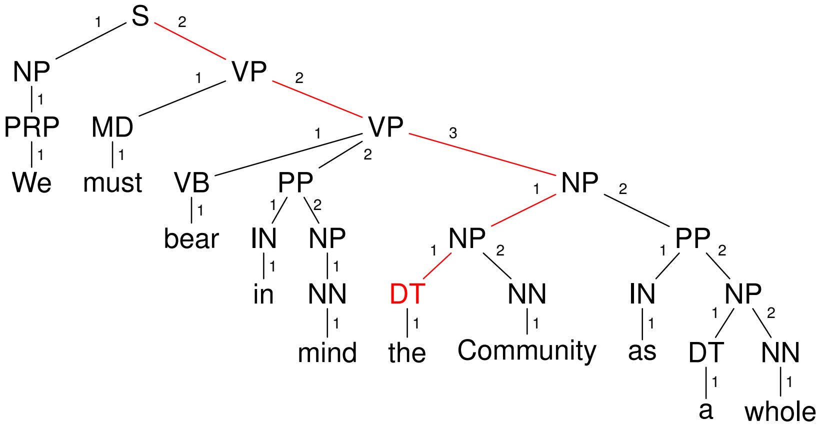


Leaves marked in red

Trees — Recursion



Trees — Recursion



Address of marked ‘DT’: 2.2.3.1.1

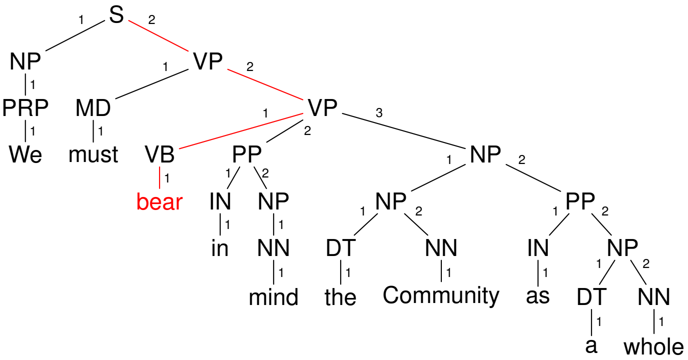
Trees

Trees $t, u \in T_\Sigma(Q)$ and position $w \in \text{pos}(t)$ in t

Notation

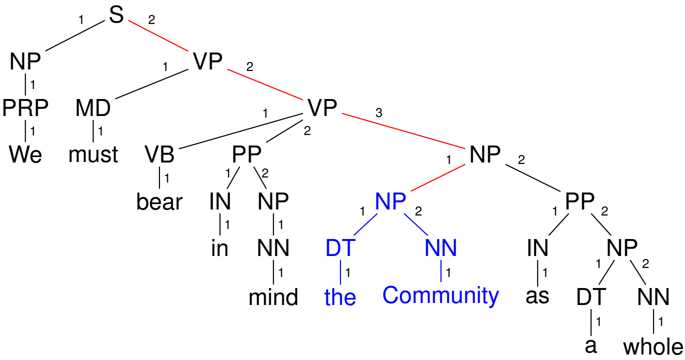
- $t(w)$ = label of t at position w
- $t|_w$ = subtree rooted in w
- $t[u]_w$ = tree obtained by replacing the subtree at w in t by u

Trees



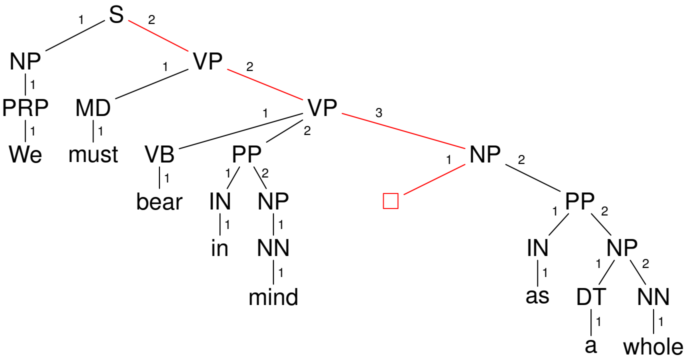
- $t(2.2.1.1) =$ bear

Trees



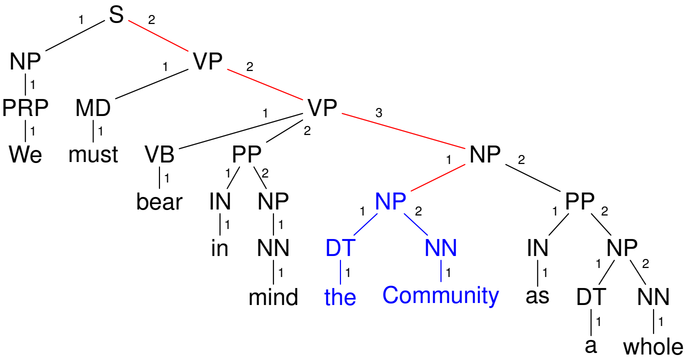
- $t(2.2.1.1) =$ bear
- $t|_{2.2.3.1} =$ NP(DT(the), NN(Community))

Trees



- $t(2.2.1.1) =$ bear
- $t|_{2.2.3.1} = \text{NP}(\text{DT}(\text{the}), \text{NN}(\text{Community}))$
- $t = t'[\text{NP}(\text{DT}(\text{the}), \text{NN}(\text{Community}))]_{2.2.3.1}$

Trees



- $t(2.2.1.1) = \text{bear}$
- $t|_{2.2.3.1} = \text{NP}(\text{DT}(\text{the}), \text{NN}(\text{Community}))$
- $t = t'[\text{NP}(\text{DT}(\text{the}), \text{NN}(\text{Community}))]_{2.2.3.1}$

Tree Language

Representations

Tree Language

Tree language (or forest) = subset of $T_\Sigma(Q)$

Motivation

- parses of a sentence — parse forest
- translations of an input sentence — translation forest
- valid XML documents — XML schema
- …

Tree Language

How to represent a set of trees?

- enumerate them

Tree Language

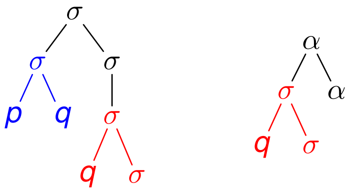
How to represent a set of trees?

- ~~enumerate them~~
- enumerate them cleverly (e.g., add sharing)

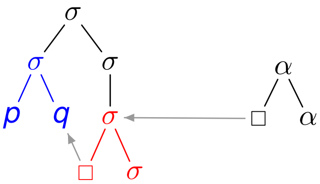
Tree Language

$L = \{\sigma(\sigma(p, q), \sigma(\sigma(q, \sigma)))\,,\ \alpha(\sigma(q, \sigma), \alpha)\}$

Enumeration of L:



Subtree-shared enumeration of L: (packed forest)



Tree Language

How to represent a set of trees?

- ~~enumerate them~~
- enumerate them cleverly (packed forest)

Tree Language

How to represent a set of trees?

- ~~enumerate them~~
- enumerate them cleverly (packed forest)
- parse forest of a CFG

Parse Forest of a CFG

Example

$$S \to NP\ VP \qquad VP \to MD\ VP$$

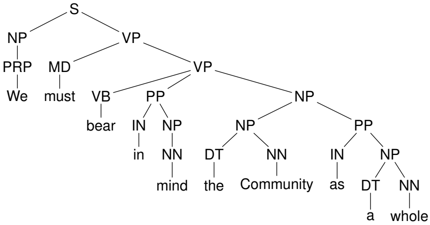
$$NP \to NP\ PP \qquad VP \to VB\ PP\ NP$$

$$MD \to must \qquad \ldots$$

Parse Forest of a CFG

Example

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



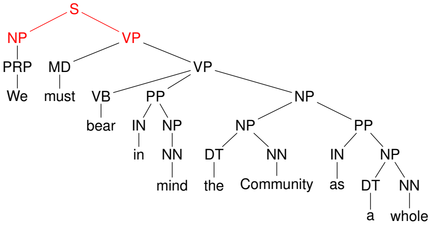
Parse Forest of a CFG

Example

$S \to NP\ VP$ $\qquad$ $VP \to MD\ VP$

$NP \to NP\ PP$ $\qquad$ $VP \to VB\ PP\ NP$

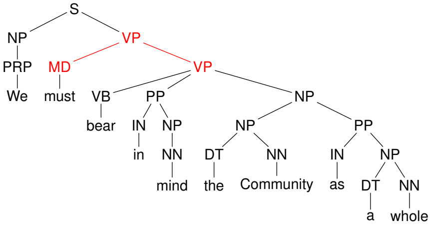
$MD \to must$ $\qquad$ $\ldots$



Parse Forest of a CFG

Example

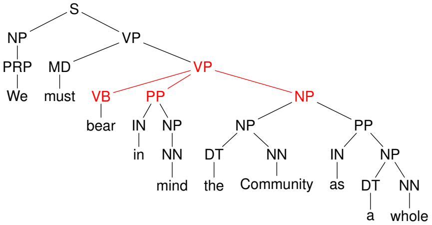
$$S \to NP\ VP \qquad VP \to MD\ VP$$
$$NP \to NP\ PP \qquad VP \to VB\ PP\ NP$$
$$MD \to \text{must} \qquad \ldots$$



Parse Forest of a CFG

Example

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



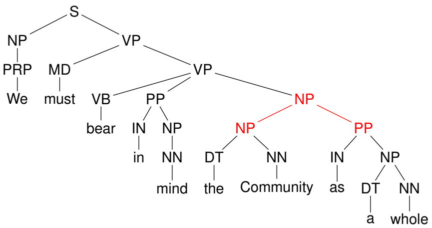
Parse Forest of a CFG

Example

$$S \to NP\ VP \qquad VP \to MD\ VP$$

$$NP \to NP\ PP \qquad VP \to VB\ PP\ NP$$

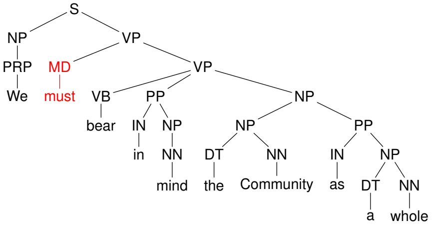
$$MD \to must \qquad \ldots$$



Parse Forest of a CFG

Example

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



Local Tree Grammar

Definition

A local tree grammar (LTG) is a CFG $G = (N, Q, S, P)$

- finite set N — nonterminals
- finite set Q — terminals
- $S \subseteq N$ — start nonterminals
- finite set $P \subseteq N \times (N \cup Q)^*$ — productions

It will compute the derivation trees of the CFG

Local Tree Grammar

LTG $G = (N, Q, S, P)$

Definition (Generated tree language)

$L(G)$ contains exactly the trees $t \in T_N(Q)$

- $t(\varepsilon) \in S$ — root label in S
- $t(w) \to t(w.1) \cdots t(w.k) \in P$ for every internal node $w \in \text{pos}(t)$ with $\{i \in \mathbb{N} \mid w.i \in \text{pos}(t)\} = \{1, \ldots, k\} \neq \emptyset$

  "label $\to$ child labels" is a production of G
- $t(w) \in Q$ for all $w \in \text{leaves}(t)$ — leaves labeled by Q

Local Tree Grammar

Observation

LTGs generate exactly the parse forests of CFGs

Local Tree Grammar

Observation

LTGs generate exactly the parse forests of CFGs

Theorem

Local tree languages (LTL) are generated by LTGs

closed under	**CFL (strings)**	**LTL (trees)**
union		
intersection		
(rank-bounded) complement		
relabeling		

Local Tree Grammar

Observation

LTGs generate exactly the parse forests of CFGs

Theorem

Local tree languages (LTL) are generated by LTGs

closed under	**CFL (strings)**	**LTL (trees)**
union	✓	
intersection		
(rank-bounded) complement		
relabeling		

Local Tree Grammar

Observation

LTGs generate exactly the parse forests of CFGs

Theorem

Local tree languages (LTL) are generated by LTGs

closed under	CFL (strings)	LTL (trees)
union	✓	
intersection	✗	
(rank-bounded) complement		
relabeling		

Local Tree Grammar

Observation

LTGs generate exactly the parse forests of CFGs

Theorem

Local tree languages (LTL) are generated by LTGs

closed under	**CFL (strings)**	**LTL (trees)**
union	✓	
intersection	✗	
(rank-bounded) complement	✗	
relabeling		

Local Tree Grammar

Observation

LTGs generate exactly the parse forests of CFGs

Theorem

Local tree languages (LTL) are generated by LTGs

closed under	CFL (strings)	LTL (trees)
union	✓	
intersection	✗	
(rank-bounded) complement	✗	
relabeling	✓	

Local Tree Grammar

Observation

LTGs generate exactly the parse forests of CFGs

Theorem

Local tree languages (LTL) are generated by LTGs

closed under	**CFL (strings)**	**LTL (trees)**
union	✓	✗
intersection	✗	
(rank-bounded) complement	✗	
relabeling	✓	

Local Tree Grammar

Observation

LTGs generate exactly the parse forests of CFGs

Theorem

Local tree languages (LTL) are generated by LTGs

closed under	CFL (strings)	LTL (trees)
union	✓	✗
intersection	✗	✓
(rank-bounded) complement	✗	
relabeling	✓	

Local Tree Grammar

Observation

LTGs generate exactly the parse forests of CFGs

Theorem

Local tree languages (LTL) are generated by LTGs

closed under	CFL (strings)	LTL (trees)
union	✓	✗
intersection	✗	✓
(rank-bounded) complement	✗	✗
relabeling	✓	

Local Tree Grammar

Observation

LTGs generate exactly the parse forests of CFGs

Theorem

Local tree languages (LTL) are generated by LTGs

closed under	**CFL (strings)**	**LTL (trees)**
union	✓	✗
intersection	✗	✓
(rank-bounded) complement	✗	✗
relabeling	✓	✗

Local Tree Grammar

Observation

LTGs generate exactly the parse forests of CFGs

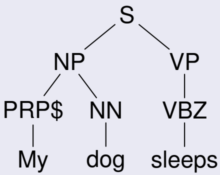
Properties

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

Local Tree Grammar

LTG $G = (N, Q, S, P)$

No ambiguity



is in $L(G)$ if and only if

1. S is a start nonterminal
2. all the productions in it are in P
3. all leaves are labeled by elements of Q

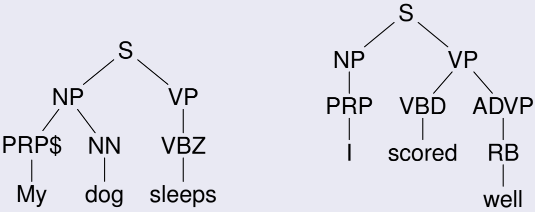
Local Tree Grammar

Observation

Local tree languages are not closed under union

Proof.

The following single-element tree languages are local:



But their union is not local as it must also generate trees for
My dog scored well and **I sleeps* □

Tree Language

How to represent a set of trees?

- ~~enumerate them~~
- enumerate them cleverly (packed forest)
- parse forest of a CFG (local tree languages)

Tree Language

How to represent a set of trees?

- ~~enumerate them~~
- enumerate them cleverly (packed forest)
- ~~parse forest of a CFG~~ (local tree languages)

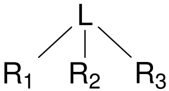
Tree Language

How to represent a set of trees?

- ~~enumerate them~~
- enumerate them cleverly (packed forest)
- ~~parse forest of a CFG~~ (local tree languages)
- tree substitution grammar

Local Tree Grammar

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

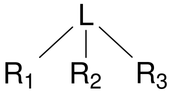


- “Glue” fragments together to obtain larger trees:

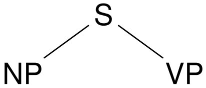
S

Local Tree Grammar

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

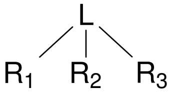


- "Glue" fragments together to obtain larger trees:

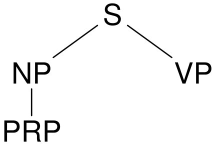


Local Tree Grammar

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

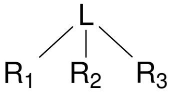


- "Glue" fragments together to obtain larger trees:

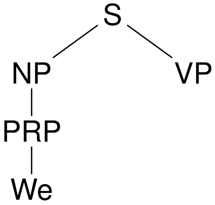


Local Tree Grammar

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

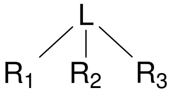


- "Glue" fragments together to obtain larger trees:

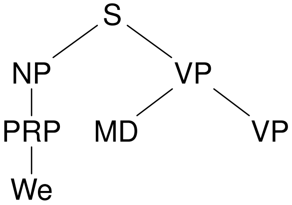


Local Tree Grammar

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

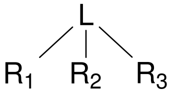


- "Glue" fragments together to obtain larger trees:

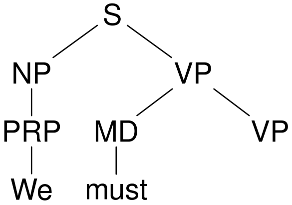


Local Tree Grammar

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

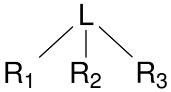


- "Glue" fragments together to obtain larger trees:

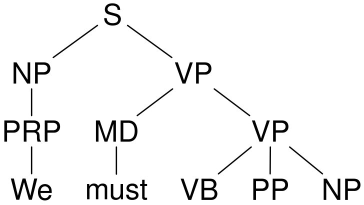


Local Tree Grammar

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

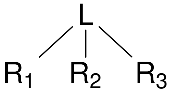


- “Glue” fragments together to obtain larger trees:

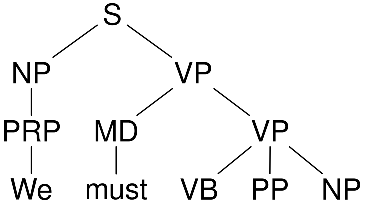


Local Tree Grammar

- 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 (Joshi 1969)

A tree substitution grammar (TSG) is a tuple (N, Q, S, P)

- finite set N — nonterminals
- finite set Q — terminals
- $S \subseteq N$ — start nonterminals
- finite set $P \subseteq T_N(Q) \setminus Q$ — tree fragments

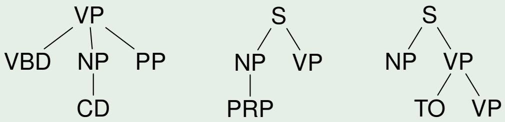
Tree Substitution Grammar

Definition (JOSHI 1969)

A tree substitution grammar (TSG) is a tuple (N, Q, S, P)

- finite set N — nonterminals
- finite set Q — terminals
- $S \subseteq N$ — start nonterminals
- finite set $P \subseteq T_N(Q) \setminus Q$ — tree fragments

Typical fragments [POST 2011]



Tree Substitution Grammar

TSG $G = (N, Q, S, P)$ and sentential forms $\xi, \zeta \in T_N(Q)$

Definition

$\xi \Rightarrow_G \zeta$ if there exist

- a tree fragment $t \in P$ and
- a position $w \in \text{leaves}(\xi)$

such that $\xi = \xi[t(\varepsilon)]_w$ and $\zeta = \xi[t]_w$

Tree Substitution Grammar

TSG $G = (N, Q, S, P)$ and sentential forms $\xi, \zeta \in T_N(Q)$

Definition

$\xi \Rightarrow_G \zeta$ if there exist

- a tree fragment $t \in P$ and
- a position $w \in \text{leaves}(\xi)$

such that $\xi = \xi[t(\varepsilon)]_w$ and $\zeta = \xi[t]_w$

Intuition

In a derivation step:

1. Find a leaf labeled by $n \in N$
2. Find a fragment $t \in P$ such that $t(\varepsilon) = n$
3. Replace selected n by t

Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

S

Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

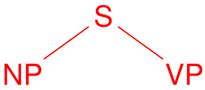
S

Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

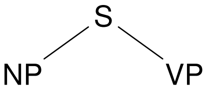


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

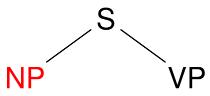


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

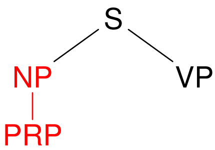


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

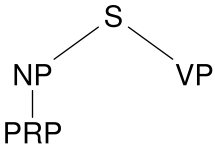


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

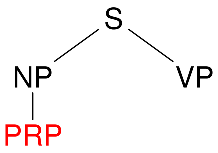


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

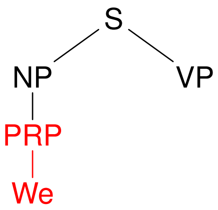


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

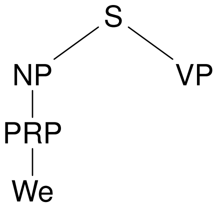


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

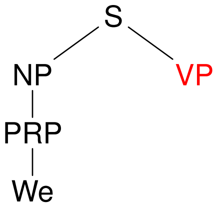


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

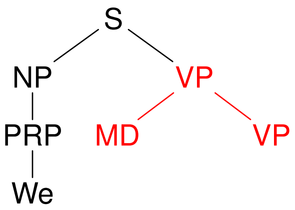


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)



Tree Substitution Grammar

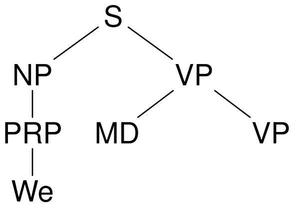
Example

Fragments:

S(NP, VP) NP(PRP)

VP(MD, VP) PRP(We)

VP(VB, PP, NP) MD(must)

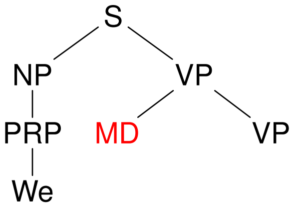


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

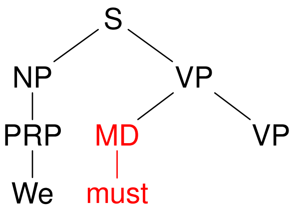


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)



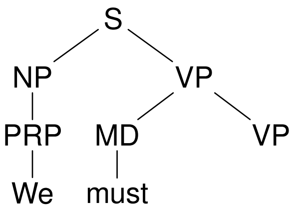
Tree Substitution Grammar

Example

Fragments:

S(NP, VP)
VP(MD, VP)
VP(VB, PP, NP)

NP(PRP)
PRP(We)
MD(must)

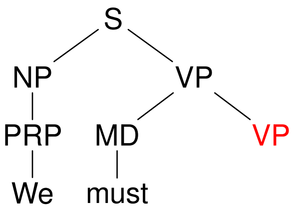


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

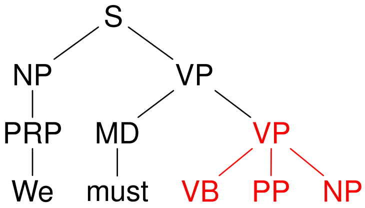


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)

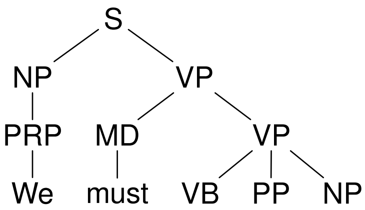


Tree Substitution Grammar

Example

Fragments:

S(NP, VP)	NP(PRP)
VP(MD, VP)	PRP(We)
VP(VB, PP, NP)	MD(must)



Tree Substitution Grammar

TSG $G = (N, Q, S, P)$

Definition

- for all $n \in N$:

  $$L(G, n) = \{t \in T_N(Q) \mid \forall w \in \text{leaves}(t) \colon t(w) \in Q,\ n \Rightarrow_G^* t\}$$

- $L(G) = \bigcup_{s \in S} L(G, s)$

Tree Substitution Grammar

TSG $G = (N, Q, S, P)$ and TSL $= \{L(G) \mid G \text{ TSG}\}$

Definition

- for all $n \in N$:

  $$L(G, n) = \{t \in T_N(Q) \mid \forall w \in \text{leaves}(t)\colon\ t(w) \in Q,\ n \Rightarrow_G^* t\}$$

- $L(G) = \bigcup_{s \in S} L(G, s)$

Theorem

1. FIN $\subsetneq$ TSL — all finite languages are TSL

Tree Substitution Grammar

TSG $G = (N, Q, S, P)$ and $\text{TSL} = \{L(G) \mid G \text{ TSG}\}$

Definition

- for all $n \in N$:

  $$L(G, n) = \{t \in T_N(Q) \mid \forall w \in \text{leaves}(t) \colon t(w) \in Q,\ n \Rightarrow_G^* t\}$$

- $L(G) = \bigcup_{s \in S} L(G, s)$

Theorem

1. $\text{FIN} \subsetneq \text{TSL}$ — all finite languages are TSL
2. $\text{LTL} \subsetneq \text{TSL}$ — all local tree languages are TSL

Tree Substitution Grammar

Theorem

Tree substitution languages (TSL) have the following properties:

closed under	**CFL**	**LTL**	**TSL**
union	✓	✗	
intersection	✗	✓	
(rank-bounded) complement	✗	✗	
relabeling	✓	✗	

Tree Substitution Grammar

Theorem

Tree substitution languages (TSL) have the following properties:

closed under	**CFL**	**LTL**	**TSL**
union	✓	✗	✗
intersection	✗	✓	
(rank-bounded) complement	✗	✗	
relabeling	✓	✗	

Tree Substitution Grammar

Theorem

Tree substitution languages (TSL) have the following properties:

closed under	CFL	LTL	TSL
union	✓	✗	✗
intersection	✗	✓	✗
(rank-bounded) complement	✗	✗	
relabeling	✓	✗	

Tree Substitution Grammar

Theorem

Tree substitution languages (TSL) have the following properties:

closed under	CFL	LTL	TSL
union	✓	✗	✗
intersection	✗	✓	✗
(rank-bounded) complement	✗	✗	✗
relabeling	✓	✗	

Tree Substitution Grammar

Theorem

Tree substitution languages (TSL) have the following properties:

closed under	**CFL**	**LTL**	**TSL**
union	✓	✗	✗
intersection	✗	✓	✗
(rank-bounded) complement	✗	✗	✗
relabeling	✓	✗	✗

Tree Substitution Grammar

Properties

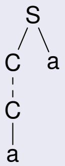
- ✓ simple
- ✓ more expressive than local tree grammars
- ✗ ambiguity (several explanations for a generated tree)
- ✗ not closed under BOOLEAN operations
  (union/intersection/complement: ✗/✗/✗)
- ✗ not closed under (non-injective) relabelings
- ✗ …

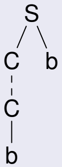
Theorem

Tree substitution languages are not closed under union

Proof.

Counterexample must be infinite $\rightsquigarrow$ artificial example





$L_1 = \{S(C^n(a), a) \mid n \in \mathbb{N}\}$ $L_2 = \{S(C^n(b), b) \mid n \in \mathbb{N}\}$

Their union is not a tree substitution language □

Tree Substitution Grammar

Theorem

Tree substitution languages are not closed under intersection

Proof.

Ideas? □

Tree Language

How to represent a set of trees?

- ~~enumerate them~~
- enumerate them cleverly (packed forest)
- ~~parse forest of a CFG~~ (local tree languages)
- tree substitution grammar
- regular tree grammar

Regular Tree Grammar

Definition (Brainerd, 1969)

A regular tree grammar (RTG) is a tuple $G = (N, \Sigma, S, P)$ with

- finite set N — nonterminals
- finite set Σ — terminals
- $S \subseteq N$ — start nonterminals
- finite set $P \subseteq N \times T_\Sigma(N)$ — productions

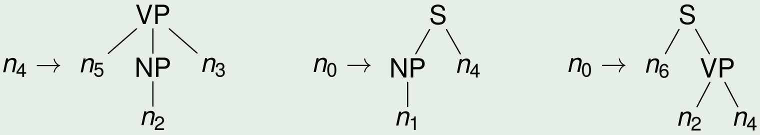
Remark

Instead of (n, t) we write $n \to t$

Regular Tree Grammar

Example

- $N = \{n_0, n_1, n_2, n_3, n_4, n_5, n_6\}$
- $\Sigma = \{\mathrm{VP}, \mathrm{NP}, \mathrm{S}, \dots\}$
- $S = \{n_0\}$
- and the following productions:



Regular Tree Grammar

RTG $G = (N, \Sigma, S, P)$ and sentential forms $\xi, \zeta \in T_\Sigma(N)$

Definition (Derivation Semantics)

$\xi \Rightarrow_G \zeta$ if there exist

- a production $n \to t \in P$ and
- a position $w \in \text{leaves}(\xi)$

such that $\xi = \xi[n]_w$ and $\zeta = \xi[t]_w$

Regular Tree Grammar

RTG $G = (N, \Sigma, S, P)$ and sentential forms $\xi, \zeta \in T_\Sigma(N)$

Definition (Derivation Semantics)

$\xi \Rightarrow_G \zeta$ if there exist

- a production $n \to t \in P$ and
- a position $w \in \text{leaves}(\xi)$

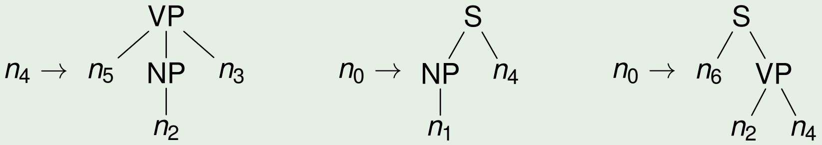
such that $\xi = \xi[n]_w$ and $\zeta = \xi[t]_w$

Definition (Recognized tree language)

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

Regular Tree Grammar

Productions

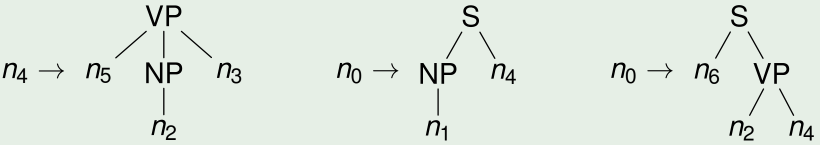


Derivation:

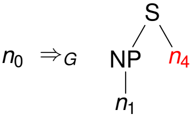
n_0

Regular Tree Grammar

Productions

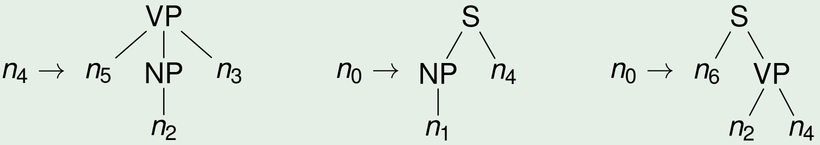


Derivation:

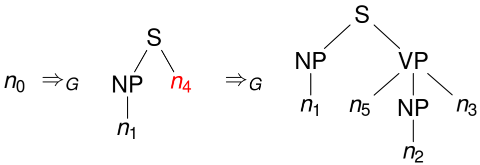


Regular Tree Grammar

Productions



Derivation:



Regular Tree Grammar

regular tree languages $\text{RTL} = \{L(G) \mid G \text{ RTG}\}$

Theorem

tree substitution languages $\subsetneq$ *regular tree languages*

Proof.

We can express the union counterexample easily □

Regular Tree Grammar

Theorem

Regular tree languages (RTL) have the following properties:

closed under	CFL	LTL	TSL	RTL
union	✓	✗	✗	
intersection	✗	✓	✗	
(rank-bounded) complement	✗	✗	✗	
relabeling	✓	✗	✗	

Regular Tree Grammar

Theorem

Regular tree languages (RTL) have the following properties:

closed under	CFL	LTL	TSL	RTL
union	✓	✗	✗	✓
intersection	✗	✓	✗	
(rank-bounded) complement	✗	✗	✗	
relabeling	✓	✗	✗	

Regular Tree Grammar

Theorem

Regular tree languages (RTL) have the following properties:

closed under	CFL	LTL	TSL	RTL
union	✓	✗	✗	✓
intersection	✗	✓	✗	✓
(rank-bounded) complement	✗	✗	✗	
relabeling	✓	✗	✗	

Regular Tree Grammar

Theorem

Regular tree languages (RTL) have the following properties:

closed under	**CFL**	**LTL**	**TSL**	**RTL**
union	✓	✗	✗	✓
intersection	✗	✓	✗	✓
(rank-bounded) complement	✗	✗	✗	✓
relabeling	✓	✗	✗	

Regular Tree Grammar

Theorem

Regular tree languages (RTL) have the following properties:

closed under	**CFL**	**LTL**	**TSL**	**RTL**
union	✓	✗	✗	✓
intersection	✗	✓	✗	✓
(rank-bounded) complement	✗	✗	✗	✓
relabeling	✓	✗	✗	✓

Regular Tree Grammar

Properties

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

Regular Tree Grammar

RTG $G = (N, \Sigma, S, P)$

Definition (Brainerd, 1969)

G is in normal form if $t = \sigma(n_1, \ldots, n_k)$ with $\sigma \in \Sigma$ and $n_1, \ldots, n_k \in N$ for all $n \to t \in P$

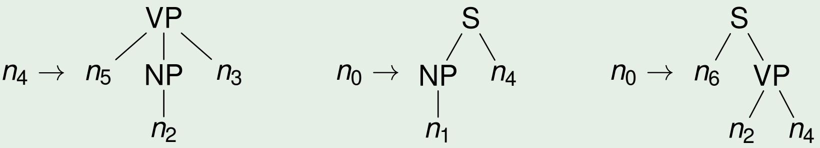
Regular Tree Grammar

RTG $G = (N, \Sigma, S, P)$

Definition (Brainerd, 1969)

G is in normal form if $t = \sigma(n_1, \ldots, n_k)$ with $\sigma \in \Sigma$ and $n_1, \ldots, n_k \in N$ for all $n \to t \in P$

Example productions



Regular Tree Grammar

Theorem (Brainerd, 1969)

Every RTG is equivalent to an RTG in normal form

Proof.

Simply cut large rules introducing new states

$$n_0 \to S(n_6, VP(n_2, n_4)) \quad \mapsto \quad n_0 \to S(n_6, \overline{n}) \qquad \overline{n} \to VP(n_2, n_4)$$

$\square$

Standard Representation

Tree Automata

Tree Automaton

Definition (Thatcher, 1970; Rounds, 1970)

A tree automaton (TA) is an RTG in normal form

Tree Automaton

Definition (THATCHER, 1970; ROUNDS, 1970)

A tree automaton (TA) is an RTG in normal form

Remarks

- bottom-up: rules written as $\sigma(n_1, \ldots, n_k) \to n$
- top-down: rules written as $n \to \sigma(n_1, \ldots, n_k)$

Tree Automaton

Definition

- top-down deterministic if $\forall n \in N, k \in \mathbb{N}, \sigma \in \Sigma$
  $\exists$ at most one $n_1, \ldots, n_k \in N$: $n \to \sigma(n_1, \ldots, n_k) \in P$
  and $|S| = 1$
- bottom-up deterministic if $\forall k \in \mathbb{N}, \sigma \in \Sigma, n_1, \ldots, n_k \in N$
  $\exists$ at most one $n \in N$: $\sigma(n_1, \ldots, n_k) \to n \in P$

(red determines blue)

Tree Automaton

Theorem (Thatcher, Wright, 1968; Doner, 1970)

top-down det. RTL $\subsetneq$ bottom-up det. RTL $=$ RTL

Tree Automaton

Theorem (Thatcher, Wright, 1968; Doner, 1970)

top-down det. RTL $\subsetneq$ bottom-up det. RTL $=$ RTL

Proof.

Let $G = (N, \Sigma, S, P)$ be a tree automaton. We construct bottom-up det. TA $G' = (2^N, \Sigma, S', P')$ with

- $S' = \{N' \subseteq N \mid N' \cap S \neq \emptyset\}$ contains start nonterminal
- for each $\sigma \in \Sigma$, $N_1, \dots, N_k \subseteq N$, and $k \in \text{rk}_P(\sigma)$

$$\{n \mid n \to \sigma(n_1, \dots, n_k) \in P,\ n_i \in N_i\} \to \sigma(N_1, \dots, N_k) \in P'$$

- no other productions are in P'

□

Tree Automaton

Theorem (Thatcher, Wright, 1968; Doner, 1970)

top-down det. RTL $\subsetneq$ bottom-up det. RTL $=$ RTL

Proof.

Let $G = (N, \Sigma, S, P)$ be a tree automaton. We construct bottom-up det. TA $G' = (2^N, \Sigma, S', P')$ with

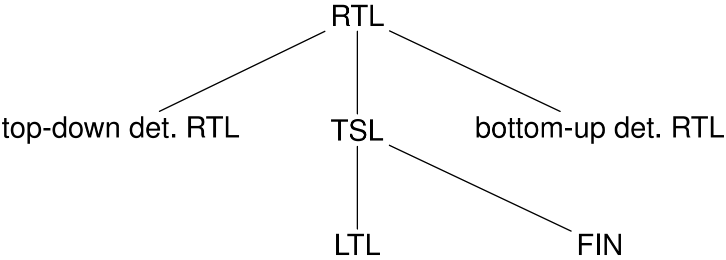
- $S' = \{N' \subseteq N \mid N' \cap S \neq \emptyset\}$ contains start nonterminal
- for each $\sigma \in \Sigma$, $N_1, \ldots, N_k \subseteq N$, and $k \in \text{rk}_P(\sigma)$

$$\{n \mid n \to \sigma(n_1, \ldots, n_k) \in P,\ n_i \in N_i\} \to \sigma(N_1, \ldots, N_k) \in P'$$

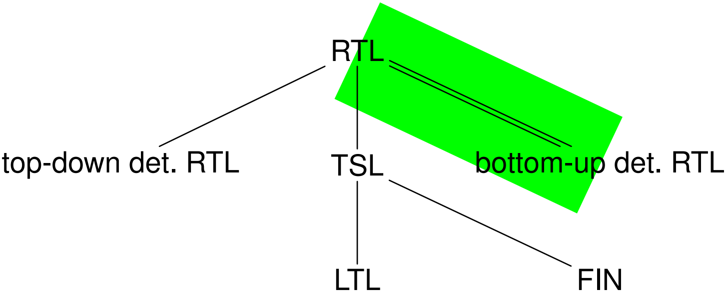
- no other productions are in P'

Strictness by the tree language $L = \{\sigma(\alpha, \beta), \sigma(\beta, \alpha)\}$ □

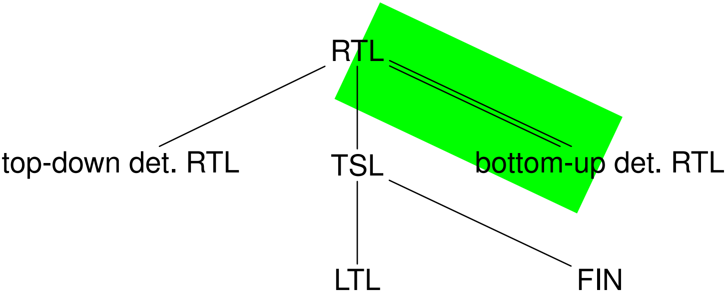
Tree Automaton



Tree Automaton



Tree Automaton



Remark

finite tree languages $\not\subseteq$ top-down deterministic RTL

Tree Automaton

Theorem

Regular tree languages are closed under

- all BOOLEAN operations
- substitution (quotients) and iteration
- (non-deterministic) relabelings
- linear homomorphisms
- inverse homomorphisms

Tree Automaton

Theorem

Regular tree languages are closed under substitution

Definition

$L, L' \subseteq T_\Sigma$ tree languages and $\alpha \in \Sigma$

$$L[\alpha \leftarrow L']$$

contains all trees obtained from a tree of L
by replacing each leaf labeled α by a tree of L'

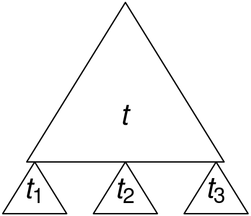
different occurrences can be differently replaced

Tree Automaton

Theorem

Regular tree languages are closed under substitution

$$L[\alpha \leftarrow L']$$



$t \in L$

$t_1, t_2, t_3 \in L'$

Tree Automaton

DTA = bottom-up deterministic tree automaton

Definition

A TA is minimal in $\mathcal{C}$ if all equivalent TAs of $\mathcal{C}$ have at least as many states

Theorem

Complexity of minimization problems:

outp. $\mathcal{C}$ \ inp. model	DTA	TA
DTA	PTime	ExpTime
TA	PSpace-hard in ExpTime	ExpTime

Tree Automaton

Definition

Height of a tree $t \in T_\Sigma(Q)$:

$$\begin{aligned} \text{height}(q) &= 0 \\ \text{height}(\sigma(t_1, \ldots, t_k)) &= 1 + \max \{\text{height}(t_i) \mid 1 \leq i \leq k\} \end{aligned}$$

Tree Automaton

Definition

Height of a tree $t \in T_\Sigma(Q)$:

$$\begin{aligned} \text{height}(q) &= 0 \\ \text{height}(\sigma(t_1, \ldots, t_k)) &= 1 + \max \{\text{height}(t_i) \mid 1 \leq i \leq k\} \end{aligned}$$

Intuition

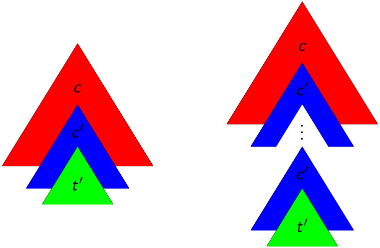
Height of t
$=$ number of edges in a maximal path from the root to a leaf

Tree Automaton

Theorem (Pumping Lemma)

For every regular tree language $L \subseteq T_\Sigma$ there exists $n \in \mathbb{N}$ such that for every $t \in L$ with $\mathrm{height}(t) \geq n$ there exist

- *contexts $c, c' \in C_\Sigma$ and $t' \in T_\Sigma$ such that $t = c[c'[t']]$*
- $c' \neq \Box$
- *$c[c'[c' \cdots c'[t'] \cdots]] \in L$ for any number of c'*



Tree Automaton

Problem	Complexity	
	DTA	TA
Emptiness	PTime	PTime
Finiteness	PTime	PTime
Universality	PTime	ExpTime
Inclusion	PTime	ExpTime
Equivalence	PTime	ExpTime
Membership	in logDCFL	logCFL

Literature

Textbooks

- COMON, DAUCHET et al.
  Tree Automata — Techniques and Applications
  `http://tata.gforge.inria.fr/`, 2008
- GÉCSEG, STEINBY
  Tree Languages
  Handbook of Formal Languages, vol. 3, Springer, 1997
- GÉCSEG, STEINBY
  Tree Automata
  Akadémiai Kiadó, 1984