Characterizations of subregular tree languages

Andreas Maletti

Institute of Computer Science, Universität Leipzig, Germany

andreas.maletti@uni-leipzig.de

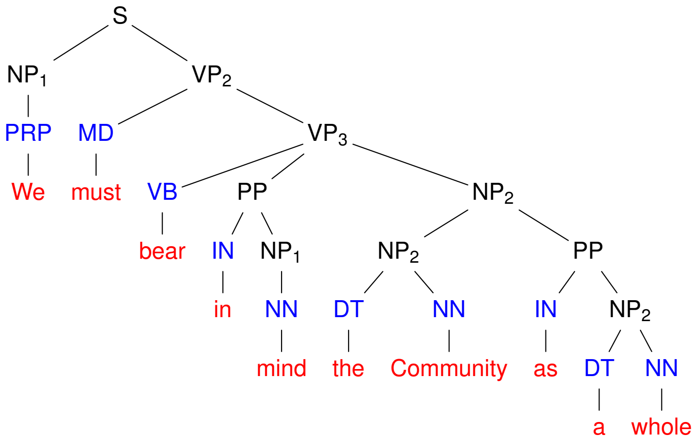
Max-Planck-Institut für Mathematik in den Naturwissenschaften

MAX-PLANCK-GESELLSCHAFT

MPI, Leipzig — November 29, 2018

Constituent Syntax Tree

Syntax tree for We must bear in mind the Community as a whole



Constituent Syntax Tree

Definition (Tree)

For sets Σ and V and $\mathrm{rk}\colon \Sigma \to \mathbb{N}$, let $T_{(\Sigma,\mathrm{rk})}(V)$ be the least set T s.t.

1. $V \subseteq T$
2. $\sigma(t_1, \ldots, t_{\mathrm{rk}(\sigma)}) \in T$ for all $\sigma \in \Sigma$ and $t_1, \ldots, t_{\mathrm{rk}(\sigma)} \in T$

Constituent Syntax Tree

Definition (Tree)

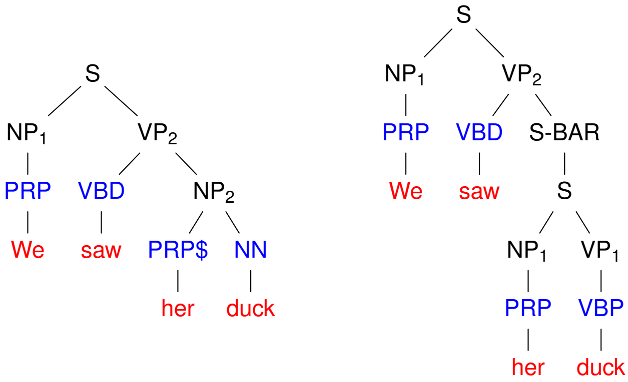
For sets Σ and V and $\mathrm{rk}\colon \Sigma \to \mathbb{N}$, let $T_{(\Sigma,\mathrm{rk})}(V)$ be the least set T s.t.

1. $V \subseteq T$
2. $\sigma(t_1, \ldots, t_{\mathrm{rk}(\sigma)}) \in T$ for all $\sigma \in \Sigma$ and $t_1, \ldots, t_{\mathrm{rk}(\sigma)} \in T$

- 2nd item: top concatenation
- 'rk' often implicit (we often write $T_\Sigma(V)$ instead of $T_{(\Sigma,\mathrm{rk})}(V)$)
- (Σ-)tree language = set $L \subseteq T_\Sigma(\emptyset)$ of trees

Constituent Syntax Trees

Syntax tree is not unique (weights are used for disambiguation)



Tree Languages

Representations

- enumeration

Tree Languages

Representations

- ~~enumeration~~
- local tree languages
- tree substitution languages
- regular tree languages

Tree Languages

Representations

- ~~enumeration~~
- local tree languages
- tree substitution languages
- regular tree languages

Definition (Regular tree language [Brainerd 1984])

$L \subseteq T_\Sigma(\emptyset)$ regular iff $\exists$ congruence $\cong$ (top-concatenation) on $T_\Sigma(\emptyset)$ s.t.

1. $\cong$ has finite index (finitely many equiv. classes)
2. $\cong$ saturates L; i.e. $L = \bigcup_{t \in L} [t]_\cong$

Regular Tree Languages

Examples for $\Sigma = \{\sigma/2, \delta/2, \alpha/0\}$:

- 2 equivalence classes (L and $T_\Sigma(\emptyset) \setminus L$)

$$L = \{t \in T_\Sigma(\emptyset) \mid t \text{ contains odd number of } \alpha\}$$

- 3 equivalence classes (“no σ”, “some σ, but legal”, illegal)

$$L' = \{t \in T_\Sigma(\emptyset) \mid \sigma \text{ never below } \delta\}$$

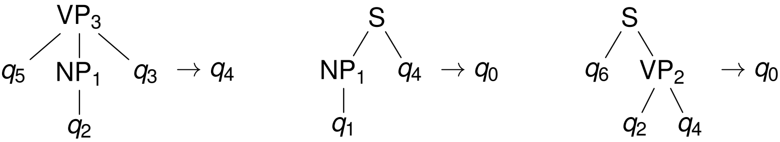
Regular Tree Languages

Definition (Regular tree grammar [Brainerd, 1969])

Regular tree grammar $G = (Q, \Sigma, I, P)$

- alphabet Q of nonterminals and initial nonterminals $I \subseteq Q$
- alphabet of terminals Σ
- finite set of productions $P \subseteq T_\Sigma(Q) \times Q$
  (we write $r \to q$ for productions (r, q))

Example productions

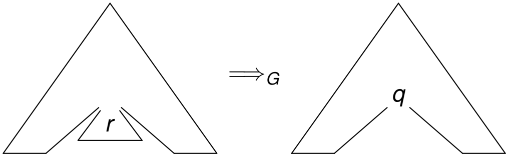


Regular Tree Languages

Derivation semantics and recognized tree language

Regular tree grammar $G = (Q, \Sigma, I, P)$

- for each production $r \to q \in P$



- generated tree language

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

Regular Tree Languages

Recall 3 equivalence classes ("no σ", "some σ, but legal", illegal)

$$L' = \{t \in T_\Sigma(\emptyset) \mid \sigma \text{ never below } \delta\}$$

$\mathcal{C}_1 = [\alpha]$ $\quad$ $\mathcal{C}_2 = [\sigma(\alpha, \alpha)]$ $\quad$ $\mathcal{C}_3 = [\delta(\sigma(\alpha, \alpha), \alpha)]$

Regular Tree Languages

Recall 3 equivalence classes ("no σ", "some σ, but legal", illegal)

$$L' = \{t \in T_\Sigma(\emptyset) \mid \sigma \text{ never below } \delta\}$$

$\mathcal{C}_1 = [\alpha]$ $\qquad$ $\mathcal{C}_2 = [\sigma(\alpha, \alpha)]$ $\qquad$ $\mathcal{C}_3 = [\delta(\sigma(\alpha, \alpha), \alpha)]$

Productions with nonterminals $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$

$$\alpha \to \mathcal{C}_1 \qquad \delta(\mathcal{C}_1, \mathcal{C}_1) \to \mathcal{C}_1$$

$$\sigma(\mathcal{C}_1, \mathcal{C}_1) \to \mathcal{C}_2 \qquad \sigma(\mathcal{C}_1, \mathcal{C}_2) \to \mathcal{C}_2 \qquad \sigma(\mathcal{C}_2, \mathcal{C}_1) \to \mathcal{C}_2 \qquad \sigma(\mathcal{C}_2, \mathcal{C}_2) \to \mathcal{C}_2$$

$$\delta(\mathcal{C}_1, \mathcal{C}_2) \to \mathcal{C}_3 \qquad \delta(\mathcal{C}_1, \mathcal{C}_3) \to \mathcal{C}_3 \qquad \delta(\mathcal{C}_2, \mathcal{C}_1) \to \mathcal{C}_3 \qquad \delta(\mathcal{C}_2, \mathcal{C}_2) \to \mathcal{C}_3$$

$$\delta(\mathcal{C}_2, \mathcal{C}_3) \to \mathcal{C}_3 \qquad \delta(\mathcal{C}_3, \mathcal{C}_1) \to \mathcal{C}_3 \qquad \delta(\mathcal{C}_3, \mathcal{C}_2) \to \mathcal{C}_3 \qquad \delta(\mathcal{C}_3, \mathcal{C}_3) \to \mathcal{C}_3$$

$$\sigma(\mathcal{C}_1, \mathcal{C}_3) \to \mathcal{C}_3 \qquad \sigma(\mathcal{C}_2, \mathcal{C}_3) \to \mathcal{C}_3 \qquad \sigma(\mathcal{C}_3, \mathcal{C}_1) \to \mathcal{C}_3 \qquad \sigma(\mathcal{C}_3, \mathcal{C}_2) \to \mathcal{C}_3$$

$$\sigma(\mathcal{C}_3, \mathcal{C}_3) \to \mathcal{C}_3$$

Regular Tree Languages

Properties

- ✓ simple
- ✓ most expressive class we consider
- ✗ ambiguity, (several explanations for a generated tree)
  but can be removed
- ✓ closed under all Boolean operations
  (union/intersection/complement: ✓/✓/✓)
- ✓ all relevant properties decidable (emptiness, inclusion, . . .)

Regular Tree Languages

Characterizations

- finite index congruences
- regular tree grammars
- (deterministic) tree automata
- regular tree expressions
- second-order logic formulas
- ...

Tree Languages

Representations

- ~~enumerate trees~~
- local tree languages
- tree substitution languages
- regular tree languages

Tree Languages

Representations

- ~~enumerate trees~~
- local tree languages
- tree substitution languages
- regular tree languages

Definition (Local tree grammar [Gécseg, Steinby 1984])

Local tree grammar = finite set of legal branchings
(together with a set of root labels)

$G = (\Sigma, I, P)$ with $I \subseteq \Sigma$ and $P \subseteq \bigcup_{k \in \mathbb{N}} \mathrm{rk}^{-1}(k) \times \Sigma^k$

Local Tree Languages

Example (with root label S)

$$
\begin{aligned}
S &\to NP_1\ VP_2 & VP_2 &\to MD\ VP_3 \\
NP_2 &\to NP_2\ PP & VP_3 &\to VB\ PP\ NP_2 \\
MD &\to \text{must} & &\dots
\end{aligned}
$$

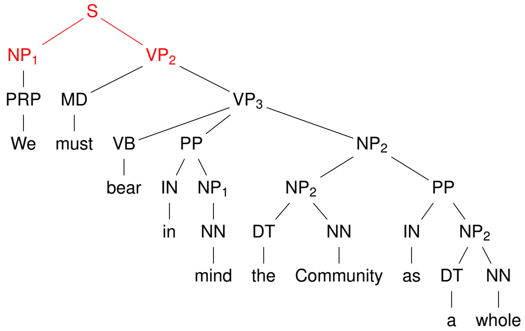
Local Tree Languages

Example (with root label S)

$$\color{red}{S \to NP_1\ VP_2} \qquad VP_2 \to MD\ VP_3$$

$$NP_2 \to NP_2\ PP \qquad VP_3 \to VB\ PP\ NP_2$$

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



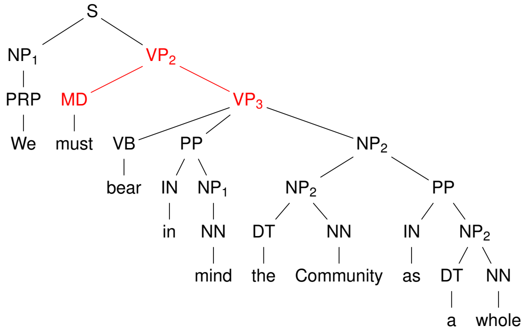
Local Tree Languages

Example (with root label S)

$$S \to NP_1\ VP_2 \qquad VP_2 \to MD\ VP_3$$

$$NP_2 \to NP_2\ PP \qquad VP_3 \to VB\ PP\ NP_2$$

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



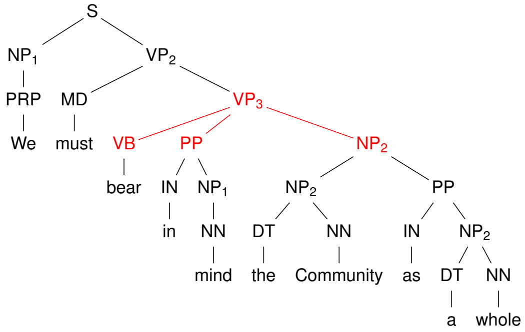
Local Tree Languages

Example (with root label S)

$$S \to NP_1\ VP_2 \qquad VP_2 \to MD\ VP_3$$

$$NP_2 \to NP_2\ PP \qquad VP_3 \to VB\ PP\ NP_2$$

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



Local Tree Languages

Example (with root label S)

$S \to NP_1\ VP_2$

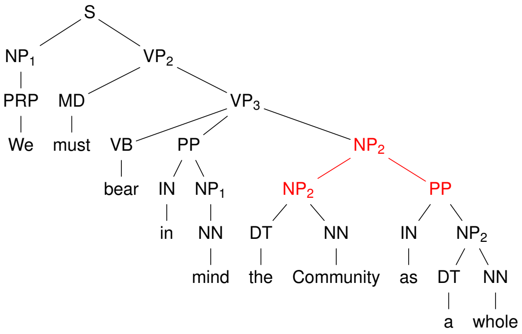
$NP_2 \to NP_2\ PP$

$MD \to \text{must}$

$VP_2 \to MD\ VP_3$

$VP_3 \to VB\ PP\ NP_2$

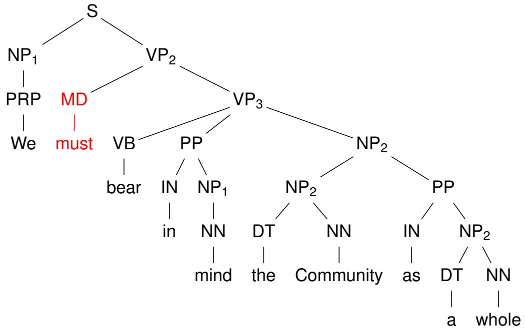
$\dots$



Local Tree Languages

Example (with root label S)

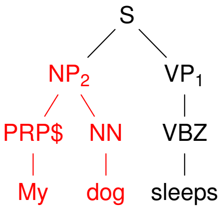
$$S \to NP_1\ VP_2 \qquad VP_2 \to MD\ VP_3$$
$$NP_2 \to NP_2\ PP \qquad VP_3 \to VB\ PP\ NP_2$$
$$MD \to \text{must} \qquad \ldots$$

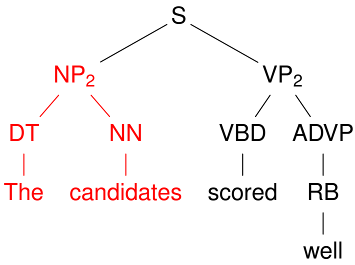


Local Tree Languages

<u>not closed under union</u>

- these singletons are local



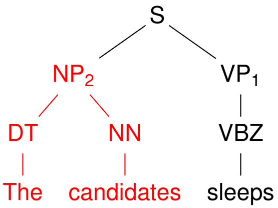


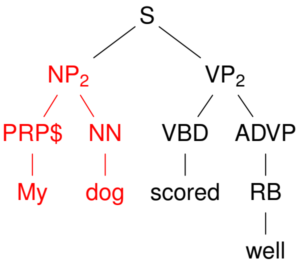
- but their union cannot be local

Local Tree Languages

<u>not closed under union</u>

- these singletons are local





- but their union cannot be local
  (as we also generate these trees — overgeneralization)

Local Tree Languages

<u>not closed under complement</u>

- this tree language L is local

$$\begin{array}{c} S \\ | \\ A \\ | \\ A' \\ | \\ a \end{array} \in L \qquad \begin{array}{c} S \\ | \\ B \\ | \\ B' \\ | \\ b \end{array} \in L$$

- but its complement cannot be local

Local Tree Languages

<u>not closed under complement</u>

- this tree language L is local

$$\begin{array}{c} S \\ | \\ A \\ | \\ A' \\ | \\ a \end{array} \in L \qquad \begin{array}{c} S \\ | \\ B \\ | \\ B' \\ | \\ b \end{array} \in L \qquad\qquad \begin{array}{c} S \\ | \\ A \\ | \\ A' \\ | \\ b \end{array} \notin L \qquad \begin{array}{c} S \\ | \\ B \\ | \\ A' \\ | \\ a \end{array} \notin L$$

- but its complement cannot be local
  (as we also generate these trees — overgeneralization)

Local Tree Languages

Properties

- ✓ simple
- ✓ no ambiguity (unique explanation for each recognized tree)
- ✗ not closed under Boolean operations
  (union/intersection/complement: ✗/✓/✗)
- ✗ not closed under (non-injective) relabelings
- ✓ locality of a regular tree language decidable

Local Tree Languages

$$\alpha \to \mathcal{C}_1 \qquad \delta(\mathcal{C}_1, \mathcal{C}_1) \to \mathcal{C}_1$$

(irrelevant productions omitted)

$$\sigma(\mathcal{C}_1, \mathcal{C}_1) \to \mathcal{C}_2 \qquad \sigma(\mathcal{C}_1, \mathcal{C}_2) \to \mathcal{C}_2 \qquad \sigma(\mathcal{C}_2, \mathcal{C}_1) \to \mathcal{C}_2 \qquad \sigma(\mathcal{C}_2, \mathcal{C}_2) \to \mathcal{C}_2$$

Local Tree Languages

$$\alpha \to \mathcal{C}_1 \qquad \delta(\mathcal{C}_1, \mathcal{C}_1) \to \mathcal{C}_1$$

(irrelevant productions omitted)

$$\sigma(\mathcal{C}_1, \mathcal{C}_1) \to \mathcal{C}_2 \qquad \sigma(\mathcal{C}_1, \mathcal{C}_2) \to \mathcal{C}_2 \qquad \sigma(\mathcal{C}_2, \mathcal{C}_1) \to \mathcal{C}_2 \qquad \sigma(\mathcal{C}_2, \mathcal{C}_2) \to \mathcal{C}_2$$

1. extract all possible branches and root labels

$$\begin{aligned}\{&\delta \to \alpha\,\alpha,\ \delta \to \alpha\,\delta,\ \delta \to \delta\,\alpha,\ \delta \to \delta\,\delta,\\ &\sigma \to \alpha\,\alpha,\ \sigma \to \alpha\,\delta,\ \sigma \to \delta\,\alpha,\ \sigma \to \delta\,\delta,\\ &\sigma \to \alpha\,\sigma,\ \sigma \to \delta\,\sigma,\ \sigma \to \sigma\,\alpha,\ \sigma \to \sigma\,\delta,\ \sigma \to \sigma\,\sigma\}\end{aligned}$$

Local Tree Languages

$$\alpha \to \mathcal{C}_1 \qquad \delta(\mathcal{C}_1, \mathcal{C}_1) \to \mathcal{C}_1$$

(irrelevant productions omitted)

$$\sigma(\mathcal{C}_1, \mathcal{C}_1) \to \mathcal{C}_2 \qquad \sigma(\mathcal{C}_1, \mathcal{C}_2) \to \mathcal{C}_2 \qquad \sigma(\mathcal{C}_2, \mathcal{C}_1) \to \mathcal{C}_2 \qquad \sigma(\mathcal{C}_2, \mathcal{C}_2) \to \mathcal{C}_2$$

1. extract all possible branches and root labels

$$\begin{aligned}\{&\delta \to \alpha\,\alpha,\ \delta \to \alpha\,\delta,\ \delta \to \delta\,\alpha,\ \delta \to \delta\,\delta,\\ &\sigma \to \alpha\,\alpha,\ \sigma \to \alpha\,\delta,\ \sigma \to \delta\,\alpha,\ \sigma \to \delta\,\delta,\\ &\sigma \to \alpha\,\sigma,\ \sigma \to \delta\,\sigma,\ \sigma \to \sigma\,\alpha,\ \sigma \to \sigma\,\delta,\ \sigma \to \sigma\,\sigma\}\end{aligned}$$

2. check whether this local tree grammar G overgeneralizes
   (check whether $L(G) \subseteq L$)

Local Tree Languages

Characterizations

- local tree grammars
- parse trees of context-free grammars
- (not much available, but seems well understood)

Tree Languages

Representations

- ~~enumerate trees~~
- local tree languages
- tree substitution languages
- regular tree languages

Tree Languages

Representations

- ~~enumerate trees~~
- local tree languages
- tree substitution languages
- regular tree languages

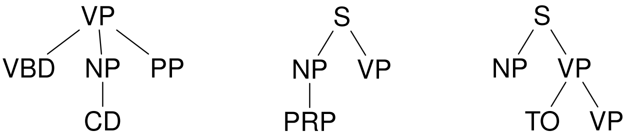
Definition (Tree substitution grammar [Joshi, Schabes 1997])

Tree substitution grammar = finite set of legal fragments
(together with a set of root labels)

$G = (\Sigma, I, P)$ with $I \subseteq \Sigma$ and finite $P \subseteq T_\Sigma(\Sigma)$

Tree Substitution Languages

Typical fragments [Post 2011]



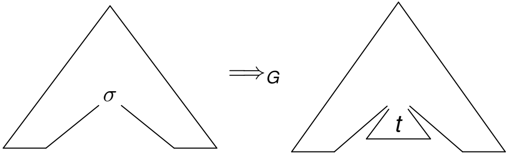
Derivation step $\xi \Rightarrow_G \zeta$

- $\xi = c\big[\mathrm{root}(t)\big]$ and $\zeta = c\big[t\big]$ for some context c and fragment $t \in P$

Tree Substitution Languages

Tree substitution grammar $G = (\Sigma, I, P)$

- for each fragment $t \in P$ with root label σ



- generated tree language

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

Tree Substitution Languages

Fragments

$\mathrm{S}\big(\mathrm{NP}_1(\mathrm{PRP}), \mathrm{VP}_2\big)$ PRP(We)

$\mathrm{VP}_2\big(\mathrm{MD}, \mathrm{VP}_3(\mathrm{VB}, \mathrm{PP}, \mathrm{NP}_2)\big)$ MD(must)

Derivation

S

Tree Substitution Languages

Fragments

$\mathrm{S}\big(\mathrm{NP}_1(\mathrm{PRP}), \mathrm{VP}_2\big)$

$\mathrm{VP}_2\big(\mathrm{MD}, \mathrm{VP}_3(\mathrm{VB}, \mathrm{PP}, \mathrm{NP}_2)\big)$

$\mathrm{PRP}(\mathrm{We})$

$\mathrm{MD}(\mathrm{must})$

Derivation

S

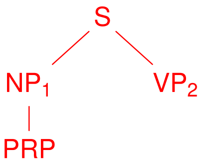
Tree Substitution Languages

Fragments

$S(NP_1(PRP), VP_2)$ $PRP(We)$

$VP_2(MD, VP_3(VB, PP, NP_2))$ $MD(must)$

Derivation



Tree Substitution Languages

Fragments

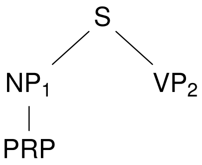
$S\big(NP_1(PRP),\, VP_2\big)$

$VP_2\big(MD,\, VP_3(VB,\, PP,\, NP_2)\big)$

$PRP(We)$

$MD(must)$

Derivation



Tree Substitution Languages

Fragments

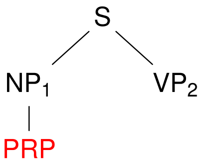
$\mathrm{S}\big(\mathrm{NP}_1(\mathrm{PRP}), \mathrm{VP}_2\big)$

$\mathrm{VP}_2\big(\mathrm{MD}, \mathrm{VP}_3(\mathrm{VB}, \mathrm{PP}, \mathrm{NP}_2)\big)$

PRP(We)

MD(must)

Derivation



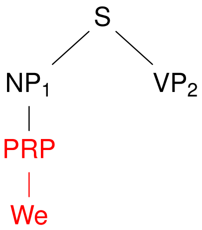
Tree Substitution Languages

Fragments

$S\big(NP_1(PRP), VP_2\big)$ $\quad$ $PRP(We)$

$VP_2\big(MD, VP_3(VB, PP, NP_2)\big)$ $\quad$ $MD(must)$

Derivation



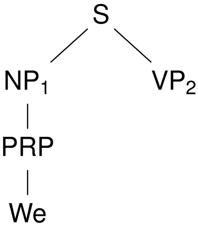
Tree Substitution Languages

Fragments

$\mathrm{S}\big(\mathrm{NP}_1(\mathrm{PRP}), \mathrm{VP}_2\big)$ $\mathrm{PRP}(\mathrm{We})$

$\mathrm{VP}_2\big(\mathrm{MD}, \mathrm{VP}_3(\mathrm{VB}, \mathrm{PP}, \mathrm{NP}_2)\big)$ $\mathrm{MD}(\mathrm{must})$

Derivation



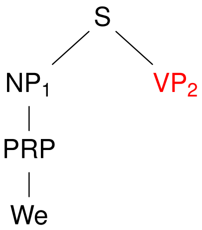
Tree Substitution Languages

Fragments

$S(NP_1(PRP), VP_2)$ $\quad$ $PRP(We)$

$VP_2(MD, VP_3(VB, PP, NP_2))$ $\quad$ $MD(must)$

Derivation



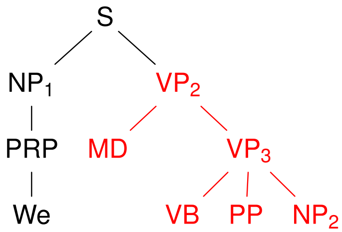
Tree Substitution Languages

Fragments

$S\big(NP_1(PRP),\, VP_2\big)$ $\quad$ $PRP(We)$

$VP_2\big(MD,\, VP_3(VB,\, PP,\, NP_2)\big)$ $\quad$ $MD(must)$

Derivation



Tree Substitution Languages

Fragments

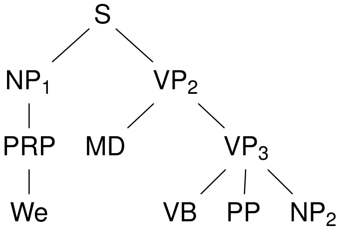
$S\big(NP_1(PRP), VP_2\big)$

$VP_2\big(MD, VP_3(VB, PP, NP_2)\big)$

$PRP(We)$

$MD(must)$

Derivation



Tree Substitution Languages

Fragments

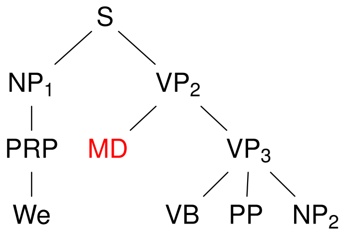
$S\big(NP_1(PRP), VP_2\big)$

$VP_2\big(MD, VP_3(VB, PP, NP_2)\big)$

$PRP(We)$

$MD(must)$

Derivation



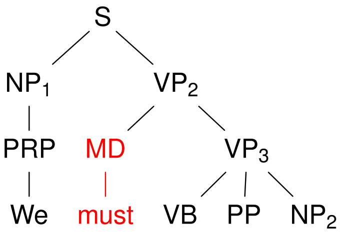
Tree Substitution Languages

Fragments

$S\big(NP_1(PRP), VP_2\big)$ $\quad$ $PRP(We)$

$VP_2\big(MD, VP_3(VB, PP, NP_2)\big)$ $\quad$ $MD(must)$

Derivation



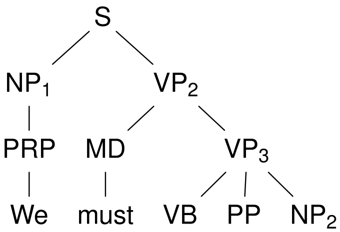
Tree Substitution Languages

Fragments

$S\big(NP_1(PRP), VP_2\big)$ $\qquad$ $PRP(We)$

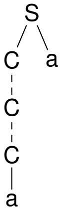
$VP_2\big(MD, VP_3(VB, PP, NP_2)\big)$ $\qquad$ $MD(must)$

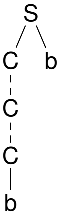
Derivation



<u>not closed under union</u>

- these languages are tree substitution languages individually





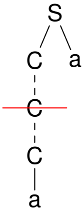
$L_1 = \{\mathsf{S}(\mathsf{C}^n(a), a) \mid n \in \mathbb{N}\}$ $L_2 = \{\mathsf{S}(\mathsf{C}^n(b), b) \mid n \in \mathbb{N}\}$

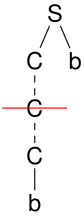
- but their union is not

Tree Substitution Languages

<u>not closed under union</u>

- these languages are tree substitution languages individually





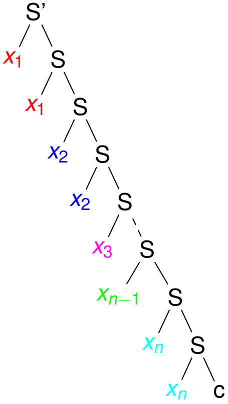
$L_1 = \{S(C^n(a), a) \mid n \in \mathbb{N}\}$ $L_2 = \{S(C^n(b), b) \mid n \in \mathbb{N}\}$

- but their union is not
  (exchange subtrees below the indicated cuts)

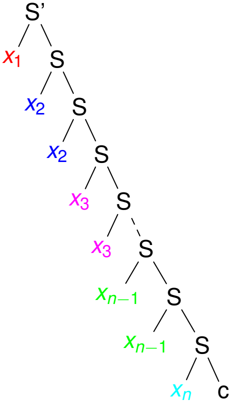
Tree Substitution Languages

<u>not closed under intersection</u>

- these languages L_1 and L_2 are tree substitution languages individually for $n \geq 1$ and arbitrary $x_1, \ldots, x_n \in \{a, b\}$



$\in L_1$

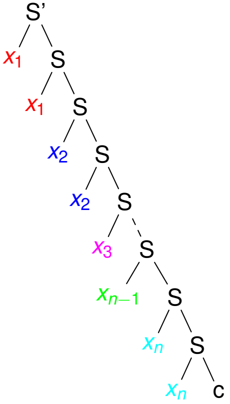


$\in L_2$

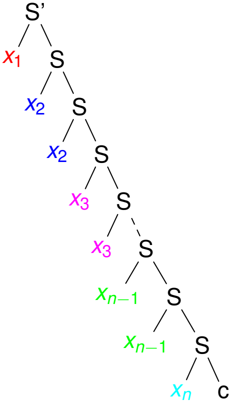
Tree Substitution Languages

<u>not closed under intersection</u>

- these languages L_1 and L_2 are tree substitution languages individually for $n \geq 1$ and arbitrary $x_1, \ldots, x_n \in \{a, b\}$



$\in L_1$

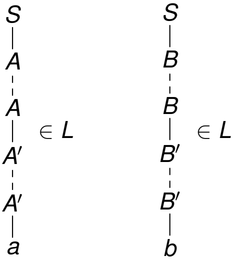


$\in L_2$

- but their intersection only contains trees with $x_1 = x_2 = \cdots = x_n$ and is not a tree substitution language

<u>not closed under complement</u>

- this language L is a tree substitution language

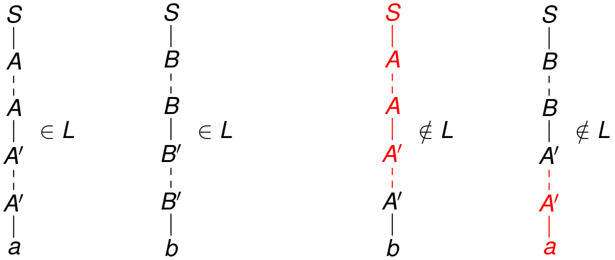


- but its complement is not

Tree Substitution Languages

<u>not closed under complement</u>

- this language L is a tree substitution language



- but its complement is not
  (exchange as indicated in red)

Tree Substitution Languages

Properties

- ✓ simple
- ✓ contain all finite and co-finite tree languages
- ✗ ambiguity (several explanations for a generated tree)
- ✗ not closed under Boolean operations
  (union/intersection/complement: ✗/✗/✗)
- ✓ can express many finite-distance dependencies
  (extended domain of locality)

Tree Substitution Languages

Characterizations

- tree substitution grammars
- ??? (generally badly understood)

Tree Substitution Languages

Characterizations

- tree substitution grammars
- ??? (generally badly understood)

Remark:

- several unions lead to additional power

Tree Substitution Languages

Open questions

- multiple intersections more expressive?
- which regular tree languages are tree substitution languages?
- relation to local tree languages?

Tree Substitution Languages

Open questions

- multiple intersections more expressive?
- which regular tree languages are tree substitution languages?
- relation to local tree languages?
- extension to weights
- application to parsing

Tree Substitution Languages

Open questions

- multiple intersections more expressive?
- which regular tree languages are tree substitution languages?
- relation to local tree languages?
- extension to weights
- application to parsing

Thank you for your attention!

Tree Substitution Languages

Experiment

[Post, Gildea 2009]

grammar	size	Prec.	Recall	F_1
local	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 Languages with Latent Variables

Experiment [Shindo et al. 2012]

grammar	F1 score $\|w\| \leq 40$	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