Characterizations of subregular tree languages

Andreas Maletti

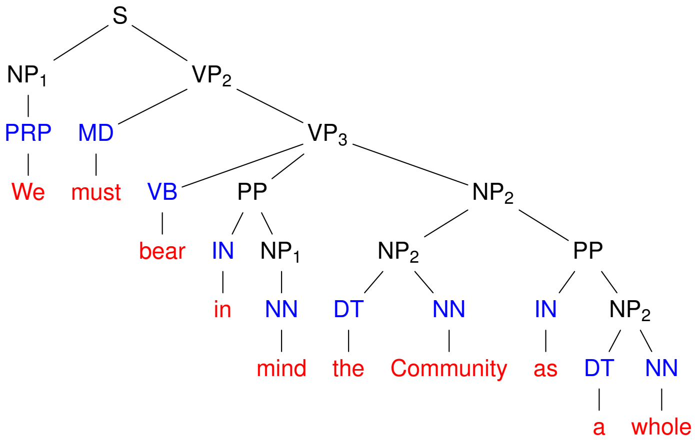
Universität Leipzig, Germany

andreas.maletti@uni-leipzig.de

CAALM, Chennai — January 24, 2019

Constituent Syntax Tree

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



Constituent Syntax Tree

Tree

$T_\Sigma(V)$ for sets Σ and V is least set T of trees s.t.

1. Variables: $V \subseteq T$
2. Top concatenation: $\sigma(t_1, \ldots, t_k) \in T$ for $k \in \mathbb{N}$, $\sigma \in \Sigma$, $t_1, \ldots, t_k \in T$

Constituent Syntax Tree

Tree

$T_\Sigma(V)$ for sets Σ and V is least set T of trees s.t.

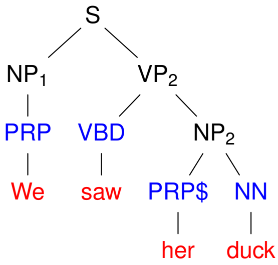
1. Variables: $V \subseteq T$
2. Top concatenation: $\sigma(t_1, \ldots, t_k) \in T$ for $k \in \mathbb{N}$, $\sigma \in \Sigma$, $t_1, \ldots, t_k \in T$

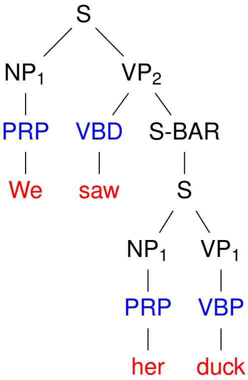
- tree language = set of trees

Constituent Syntax Trees

Syntax tree is not unique

(weights are used for disambiguation)





Parses

Representations

- enumeration

Parses

Representations

- ~~enumeration~~
- proof trees of combinatory categorial grammars
- local tree languages
- tree substitution languages
- regular tree languages

Parses

Representations

- ~~enumeration~~
- proof trees of combinatory categorial grammars
- local tree languages
- tree substitution languages
- regular tree languages

Regular tree language

$L \subseteq T_\Sigma(\emptyset)$ 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 $\Sigma = \{\sigma, \delta, \alpha\}$:

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

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

Regular Tree Languages

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

- 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

Regular tree grammar [Brainerd, 1969]

$G = (Q, \Sigma, I, P)$

- 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))

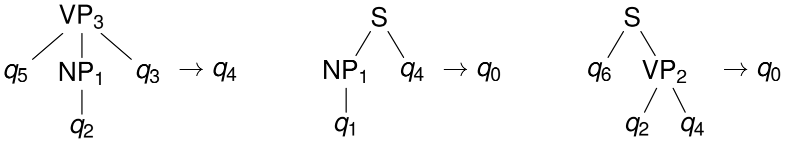
Regular Tree Languages

Regular tree grammar [Brainerd, 1969]

$G = (Q, \Sigma, I, P)$

- 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

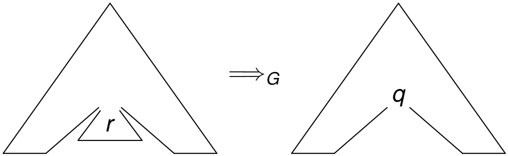


Regular Tree Languages

Derivation semantics and recognized tree language

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

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

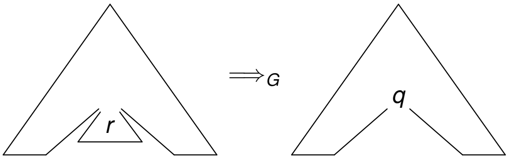


Regular Tree Languages

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)

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

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

Regular Tree Languages

<u>Recall</u> 3 equivalence classes (“no σ”, “some σ, but legal”, illegal)

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

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

<u>Productions</u> 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
- monadic second-order formulas
- ...

Parses

Representations

- ~~enumeration~~
- proof trees of combinatory categorial grammars
- local tree languages
- tree substitution languages
- regular tree languages

Parses

Representations

- ~~enumeration~~
- proof trees of combinatory categorial grammars
- local tree languages
- tree substitution languages
- regular tree languages

Categories

- category = tree of $T_S(A)$ with $S = \{/, \backslash\}$ and atomic categories A
- e.g. $D/E/E\backslash C$ corresponds to $\backslash(/(/(D, E), E), C)$

Combinatory Categorial Grammars

Combinators (Compositions)

Composition rules of degree k are

$$
\begin{aligned}
ax/c, \quad cy \quad &\rightarrow \quad axy && \text{(forward rule)} \\
cy, \quad ax\backslash c \quad &\rightarrow \quad axy && \text{(backward rule)}
\end{aligned}
$$

with $y = |_1 c_1 \, |_2 \cdots |_k c_k$

Combinatory Categorial Grammars

Combinators (Compositions)

Composition rules of degree k are

$$ax/c, \quad cy \quad \rightarrow \quad axy \qquad \text{(forward rule)}$$

$$cy, \quad ax\backslash c \quad \rightarrow \quad axy \qquad \text{(backward rule)}$$

with $y = |_1 c_1 \, |_2 \cdots |_k c_k$

Examples:

$$\underbrace{\frac{C \quad D/E/D\backslash C}{D/E/D}}_{\text{degree } 0} \qquad \underbrace{\frac{D/E/D \quad D/E\backslash C}{D/E/E\backslash C}}_{\text{degree } 2}$$

Combinatory Categorial Grammar (CCG)

(Σ, A, k, I, L)

- terminal alphabet Σ and atomic categories A
- maximal degree $k \in \mathbb{N} \cup \{\infty\}$ of composition rules
- initial categories $I \subseteq A$
- lexicon $L \subseteq \Sigma \times \mathcal{C}(A)$ with $\mathcal{C}(A)$ categories over A

Combinatory Categorial Grammars

Combinatory Categorial Grammar (CCG)

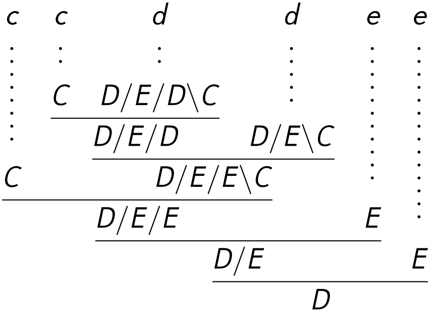
(Σ, A, k, I, L)

- terminal alphabet Σ and atomic categories A
- maximal degree $k \in \mathbb{N} \cup \{\infty\}$ of composition rules
- initial categories $I \subseteq A$
- lexicon $L \subseteq \Sigma \times \mathcal{C}(A)$ with $\mathcal{C}(A)$ categories over A

Notes:

- always all rules up to the given degree k allowed
- k-CCG = CCG using all composition rules up to degree k

Combinatory Categorial Grammars



2-CCG generates string language $\mathcal{L}$ with $\mathcal{L} \cap c^+ d^+ e^+ = \{c^i d^i e^i \mid i \geq 1\}$ for initial categories $\{D\}$

$$L(c) = \{C\}$$
$$L(d) = \{D/E\backslash C,\ D/E/D\backslash C\}$$
$$L(e) = \{E\}$$

Combinatory Categorial Grammars

- allow (deterministic) relabeling (to allow arbitrary labels)
- tree t min-height bounded by k
  if the minimal distance from each node to a leaf is at most k

Theorem

(Under relabeling) Class of proof trees of 0-CCGs
$=$ class of min-height bounded binary regular tree languages

joint work with Marco Kuhlmann

Combinatory Categorial Grammars

Theorem

(Under relabeling) Class of proof trees of 1-CCGs
$\subsetneq$ class of binary regular tree languages

Combinatory Categorial Grammars

Theorem

(Under relabeling) Class of proof trees of 1-CCGs
$\subsetneq$ class of binary regular tree languages

Theorem

(Under relabeling*) Class of proof trees of ∞-CCGs
$\subsetneq$ class of simple context-free tree languages

joint work with Marco Kuhlmann

Combinatory Categorial Grammars

$$\dfrac{\dfrac{ax/(by) \quad by\alpha/c}{ax\alpha/c} \quad c}{ax\alpha} \xrightarrow{\text{R1}} \dfrac{ax/(by) \quad \dfrac{by\alpha/c \quad c}{by\alpha}}{ax\alpha}$$

$$\dfrac{c \quad \dfrac{by\alpha\backslash c \quad ax\backslash(by)}{ax\alpha\backslash c}}{ax\alpha} \xrightarrow{\text{R2}} \dfrac{\dfrac{c \quad by\alpha\backslash c}{by\alpha} \quad ax\backslash(by)}{ax\alpha}$$

$$\dfrac{c \quad \dfrac{ax/(by) \quad by\alpha\backslash c}{ax\alpha\backslash c}}{ax\alpha} \xrightarrow{\text{R3}} \dfrac{ax/(by) \quad \dfrac{c \quad by\alpha\backslash c}{by\alpha}}{ax\alpha}$$

$$\dfrac{\dfrac{by\alpha/c \quad ax\backslash(by)}{ax\alpha/c} \quad c}{ax\alpha} \xrightarrow{\text{R4}} \dfrac{\dfrac{by\alpha/c \quad c}{by\alpha} \quad ax\backslash(by)}{ax\alpha}$$

joint work with Marco Kuhlmann

Combinatory Categorial Grammars

Properties

- ✓ simple
- ✗ ambiguity (several explanations for each recognized tree)
- ✗ not closed under Boolean operations
  (union/intersection/complement: ✓/?/✗*)
- ✓ closed under (non-injective) relabelings
- ? decidability of membership for subregular classes (0-CCG & 1-CCG) of a regular tree language

Tree Languages

Representations

- ~~enumerate trees~~
- proof trees of combinatory categorial grammars
- local tree languages
- tree substitution languages
- regular tree languages

Tree Languages

Representations

- ~~enumerate trees~~
- proof trees of combinatory categorial grammars
- local tree languages
- tree substitution languages
- regular tree languages

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 $P \subseteq \bigcup_{k \in \mathbb{N}} \Sigma \times \Sigma^k$

Local Tree Languages

Example (with root label S)

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

$$NP_2 \rightarrow NP_2\ PP \qquad VP_3 \rightarrow VB\ PP\ NP_2$$

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

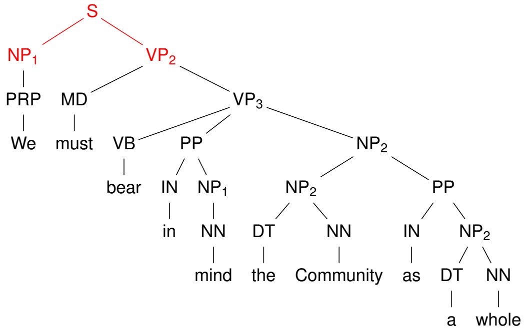
Local Tree Languages

Example (with root label S)

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

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

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



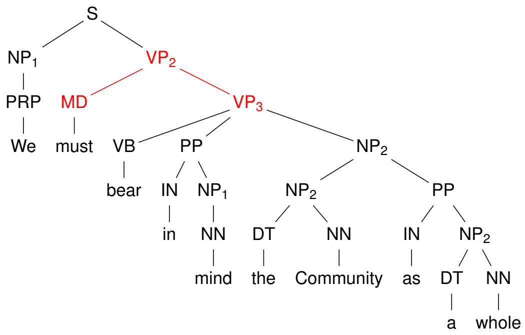
Local Tree Languages

Example (with root label S)

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

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

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



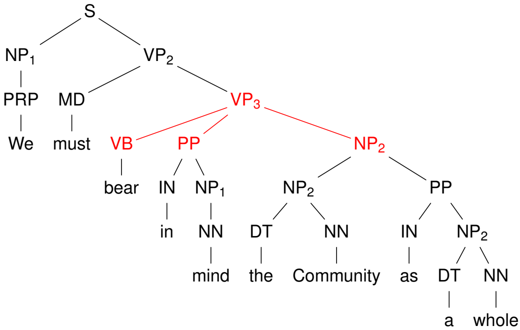
Local Tree Languages

Example (with root label S)

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

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

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



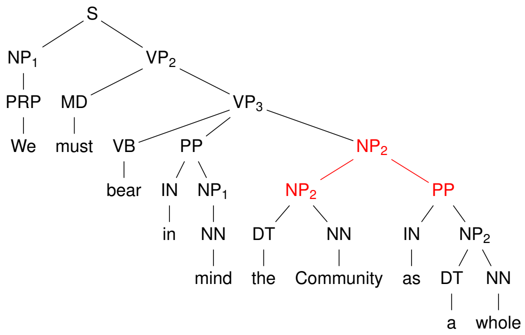
Local Tree Languages

Example (with root label S)

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

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

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



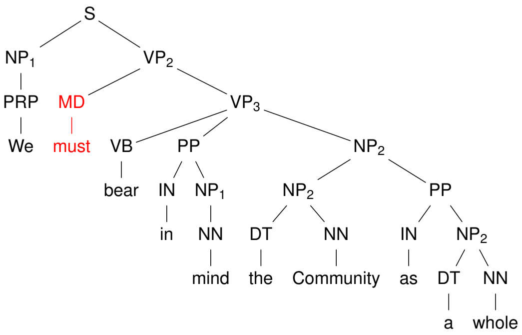
Local Tree Languages

Example (with root label S)

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

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

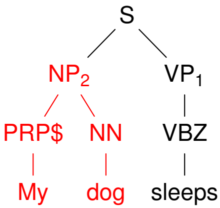
$$MD \to must \qquad \ldots$$

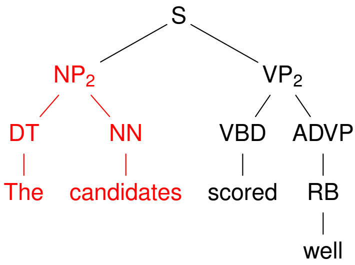


Local Tree Languages

<u>not closed under union</u>

- these singletons are local



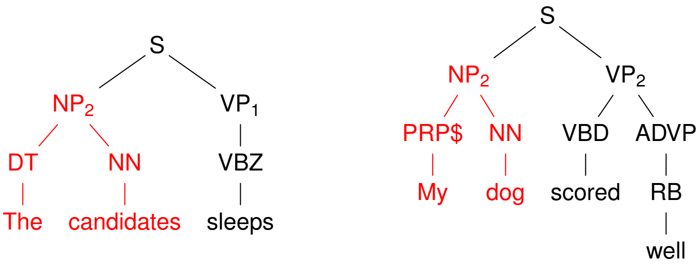


- but their union cannot be local

Local Tree Languages

not closed under union

- these singletons are local



- but their union 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

Tree Languages

Representations

- ~~enumerate trees~~
- proof trees of combinatory categorial grammars
- local tree languages
- tree substitution languages
- regular tree languages

Tree Languages

Representations

- ~~enumerate trees~~
- proof trees of combinatory categorial grammars
- local tree languages
- tree substitution languages
- regular tree languages

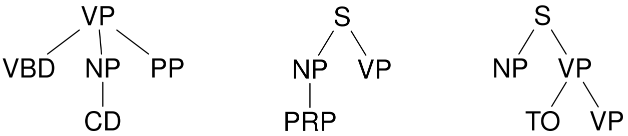
Tree substitution grammar [Joshi, Schabes 1997]

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]



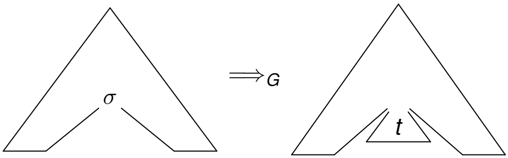
Derivation step $\xi \Rightarrow_G \zeta$

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

Tree Substitution Languages

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

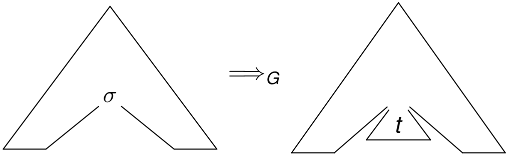
- for each fragment $t \in P$ with root label σ



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

$S(NP_1(PRP), VP_2)$

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

PRP(We)

MD(must)

Derivation

S

Tree Substitution Languages

Fragments

$S(NP_1(PRP), VP_2)$

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

PRP(We)

MD(must)

Derivation

S

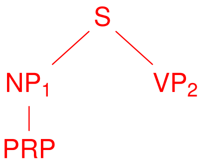
Tree Substitution Languages

Fragments

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

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

Derivation



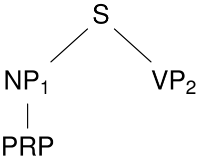
Tree Substitution Languages

Fragments

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

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

Derivation



Tree Substitution Languages

Fragments

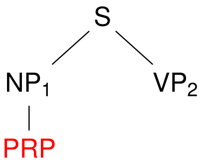
$S(NP_1(PRP), VP_2)$

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

PRP(We)

MD(must)

Derivation



Tree Substitution Languages

Fragments

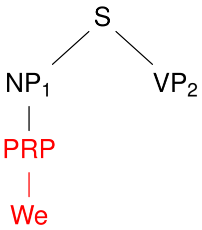
$S\big(NP_1(PRP), VP_2\big)$

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

PRP(We)

MD(must)

Derivation



Tree Substitution Languages

Fragments

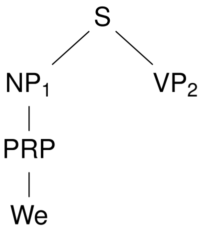
$S(NP_1(PRP), VP_2)$

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

$PRP(We)$

$MD(must)$

Derivation



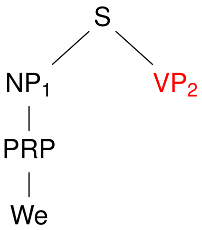
Tree Substitution Languages

Fragments

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

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

Derivation



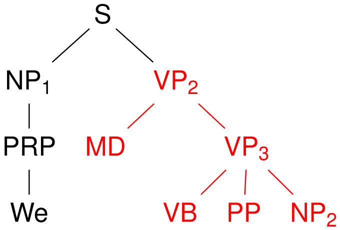
Tree Substitution Languages

Fragments

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

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

Derivation



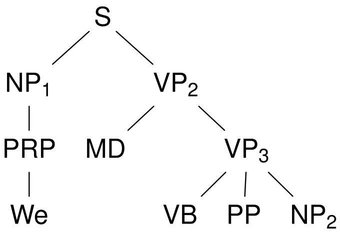
Tree Substitution Languages

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



Tree Substitution Languages

Fragments

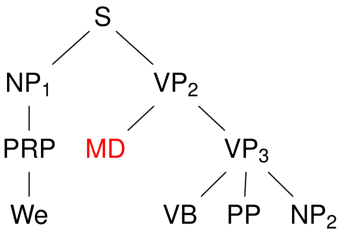
$S\big(NP_1(PRP), VP_2\big)$

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

$PRP(We)$

$MD(must)$

Derivation



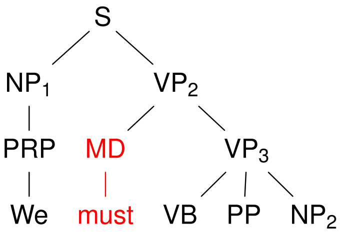
Tree Substitution Languages

Fragments

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

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

Derivation



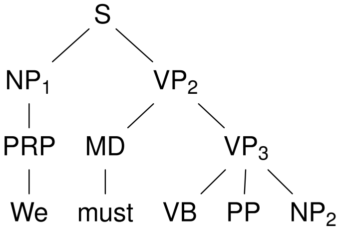
Tree Substitution Languages

Fragments

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

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

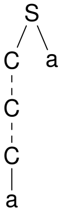
Derivation

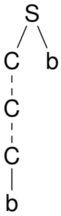


Tree Substitution Languages

<u>not closed under union</u>

- these languages are tree substitution languages individually





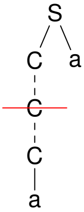
$L_1 = \{S(C^n(a), a) \mid n \in \mathbb{N}\}$ $L_2 = \{S(C^n(b), b) \mid n \in \mathbb{N}\}$

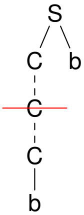
- but their union is not

Tree Substitution Languages

<u>not closed under union</u>

- these languages are tree substitution languages individually





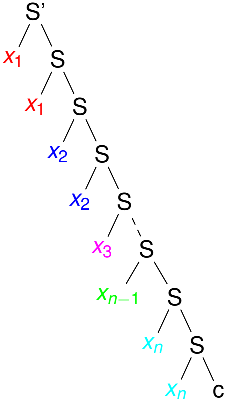
$L_1 = \{\mathsf{S}(\mathsf{C}^n(a), a) \mid n \in \mathbb{N}\}$ $L_2 = \{\mathsf{S}(\mathsf{C}^n(b), b) \mid n \in \mathbb{N}\}$

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

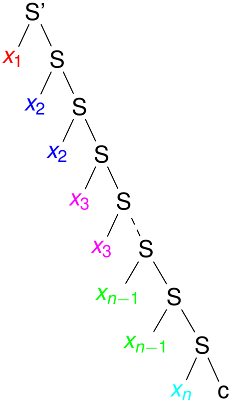
Tree Substitution Languages

not closed under intersection

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



$\in L_1$

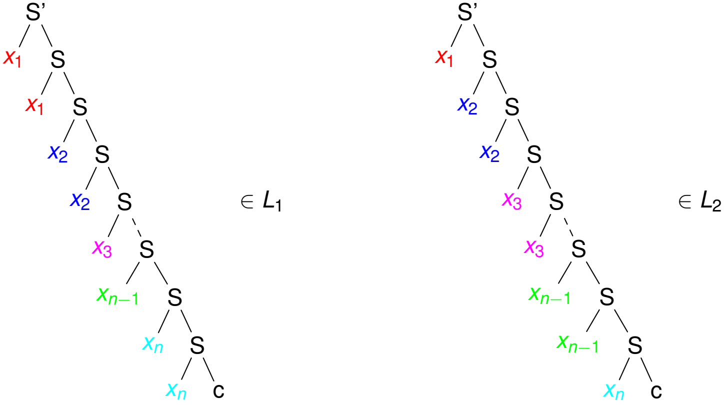


$\in L_2$

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 $x_1, \ldots, x_n \in \{a, b\}$

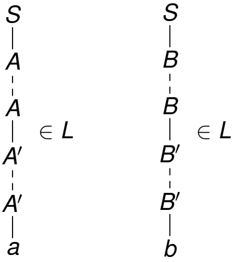


- but their intersection only contains trees with $x_1 = x_2 = \cdots = x_n$ and is not a tree substitution language

Tree Substitution Languages

not closed under complement

- this language L is a tree substitution language

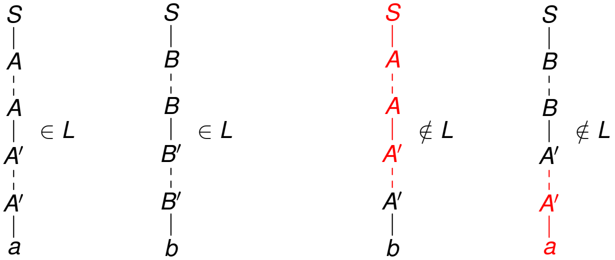


- but its complement is not

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

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 $\lvert w \rvert \leq 40$	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