Characterizations of subregular tree languages

Andreas Maletti

Universität Leipzig, Germany
andreas.maletti@uni-leipzig.de

CAALM, Chennai — January 24, 2019
We must bear in mind the Community as a whole
Constituent Syntax Tree

$T_{\Sigma}(V)$ for sets $\Sigma$ 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 \( \Sigma \) 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
Syntax tree is not unique
(weights are used for disambiguation)
Parses

Representations
- enumeration

Regular tree language $L \subseteq T_{\Sigma}(\emptyset)$ regular iff $\exists$ congruence $\sim = \text{top-concatenation}$ on $T_{\Sigma}(\emptyset)$ s.t.

1. $\sim$ has finite index (finitely many equiv. classes)
2. $\sim$ saturates $L$; i.e. $L = \bigcup_{t \in L} [t] \sim$
Representations
- enumeration
- proof trees of combinatory categorial grammars
- local tree languages
- tree substitution languages
- regular tree languages
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. $L = \bigcup_{t \in L} [t] \cong$
Examples for $\Sigma = \{\sigma, \delta, \alpha\}$:

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

$$L = \{ t \in T_\Sigma(\emptyset) \mid t \text{ contains odd number of } \alpha \}$$
Examples for $\Sigma = \{\sigma, \delta, \alpha\}$:

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

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

- 3 equivalence classes (“no $\sigma$”, “some $\sigma$, but legal”, illegal)

  $$L' = \{ t \in T_{\Sigma}(\emptyset) \mid \sigma \text{ never below } \delta \}$$
Regular tree grammar [Brainerd, 1969]

\[ G = (Q, \Sigma, I, P) \]

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

(we write \( r \rightarrow q \) for productions \( (r, q) \))
Regular tree grammar [Brainerd, 1969]

\[ G = (Q, \Sigma, I, P) \]

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

Example productions

```
VP_3
  \( q_5 \)
  NP_1
    \( q_2 \)
    q_3 \rightarrow q_4
      NP_1
        q_1

S
  q_4 \rightarrow q_0
    NP_1
      q_1

S
  q_6 \rightarrow q_0
    VP_2
      q_2
      q_4
```
Regular Tree Languages

Derivation semantics and recognized tree language

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

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

$$\Rightarrow_{G}$$
Regular Tree Languages

Derivation semantics and recognized tree language

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

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

  generated tree language

  $$L(G) = \{ t \in T_{\Sigma}(\emptyset) \mid \exists q \in I: t \Rightarrow^*_G q \}$$
Recall 3 equivalence classes ("no $\sigma$", "some $\sigma$, but legal", illegal)

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

\[ C_1 = [\alpha] \quad C_2 = [\sigma(\alpha, \alpha)] \quad C_3 = [\delta(\sigma(\alpha, \alpha), \alpha)] \]
Regular Tree Languages

Recall 3 equivalence classes ("no $\sigma$", "some $\sigma$, but legal", illegal)

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

$$C_1 = [\alpha] \quad C_2 = [\sigma(\alpha, \alpha)] \quad C_3 = [\delta(\sigma(\alpha, \alpha), \alpha)]$$

Productions with nonterminals $C_1, C_2, C_3$

$$\begin{align*}
\alpha & \rightarrow C_1 \\
\delta(C_1, C_1) & \rightarrow C_1 \\
\sigma(C_1, C_1) & \rightarrow C_2 \\
\sigma(C_1, C_2) & \rightarrow C_2 \\
\delta(C_1, C_2) & \rightarrow C_3 \\
\delta(C_1, C_3) & \rightarrow C_3 \\
\delta(C_2, C_1) & \rightarrow C_3 \\
\delta(C_2, C_2) & \rightarrow C_3 \\
\delta(C_2, C_3) & \rightarrow C_3 \\
\sigma(C_1, C_3) & \rightarrow C_3 \\
\sigma(C_2, C_3) & \rightarrow C_3 \\
\sigma(C_3, C_1) & \rightarrow C_3 \\
\sigma(C_3, C_2) & \rightarrow C_3 \\
\sigma(C_3, C_3) & \rightarrow C_3
\end{align*}$$
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
- ...
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 $S = \{/,\}\}$ and atomic categories $A$
- e.g. $D/E/E/C$ corresponds to $\\((/((D, E), E), C)\)$
Combinators (Compositions)

Composition rules of degree $k$ are

\[
ax/c, \quad cy \rightarrow axy \quad \text{(forward rule)}
\]
\[
 cy, \quad ax\backslash c \rightarrow axy \quad \text{(backward rule)}
\]

with $y = |c_1 |_2 \cdots |c_k$
Combinatory Categorial Grammars

Combinators (Compositions)

Composition rules of degree $k$ are

$$ax/c, \quad cy \rightarrow axy$$  \hspace{1cm} \text{(forward rule)}

$$cy, \quad ax\backslash c \rightarrow axy$$  \hspace{1cm} \text{(backward rule)}

with $y = |_{1}c_{1}|_{2} \cdots |_{k}c_{k}$

Examples:

$$\begin{array}{c}
C & D/E/D\backslash C \\
\hline
D/E/D
\end{array}$$  \hspace{1cm} \text{degree 0}

$$\begin{array}{c}
D/E/D & D/E\backslash C \\
\hline
D/E/E\backslash C
\end{array}$$  \hspace{1cm} \text{degree 2}
Combinatory Categorial Grammar (CCG)

$(\Sigma, A, k, I, L)$

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

$(\Sigma, A, k, I, L)$

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

Notes:

- always all rules up to the given degree $k$ allowed
- $k$-CCG = CCG using all composition rules up to degree $k$
2-CCG generates string language $\mathcal{L}$ with $\mathcal{L} \cap c^+ d^+ e^+ = \left\{ c^i d^i e^i \mid i \geq 1 \right\}$ 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 \( t \) \textit{min-height bounded by} \( k \)
  if the minimal distance from each node to a leaf is at most \( k \)

\begin{tcolorbox}
\textbf{Theorem}
(Under relabeling) Class of proof trees of 0-CCGs
= class of min-height bounded binary regular tree languages
\end{tcolorbox}

joint work with Marco Kuhlmann
Combinatory Categorial Grammars

Theorem

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

joint work with Marco Kuhlmann
(Under relabeling) Class of proof trees of 1-CCGs $\subseteq$ class of binary regular tree languages

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

joint work with Marco Kuhlmann
Combinatory Categorial Grammars

\[
\frac{ax/(by) \; by\alpha/c}{\quad \frac{by\alpha/c \; c}{\quad ax\alpha}} \quad R1 \quad \frac{by\alpha/c \; c}{\quad \frac{by\alpha}{\quad ax\alpha}}
\]

\[
\frac{by\alpha \backslash c \; ax \backslash (by)}{\quad \frac{c \; ax\alpha \backslash c}{\quad ax\alpha \backslash c}} \quad R2 \quad \frac{c \; by\alpha \backslash c}{\quad \frac{by\alpha}{\quad ax\backslash (by)}}
\]

\[
\frac{ax/(by) \; by\alpha \backslash c}{\quad \frac{by\alpha \backslash c \; c}{\quad ax\alpha \backslash c}} \quad R3 \quad \frac{c \; by\alpha \backslash c}{\quad \frac{by\alpha}{\quad ax\backslash (by)}}
\]

\[
\frac{by\alpha/c \; ax \backslash (by)}{\quad \frac{by\alpha/c \; c}{\quad ax\alpha \backslash (by)}} \quad R4 \quad \frac{by\alpha/c \; c}{\quad \frac{by\alpha}{\quad ax\backslash (by)}}
\]

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]

A local tree grammar is a finite set of legal branchings (together with a set of root labels).

Given a local tree grammar $G = (\Sigma, I, P)$ with $I \subseteq \Sigma$ and $P \subseteq \bigcup_{k \in \mathbb{N}} \Sigma \times \Sigma^k$, it defines a set of legal trees.
### Local Tree Languages

**Example (with root label S)**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Left-hand side</th>
<th>Right-hand side</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP₁ VP₂</td>
<td>VP₂ → MD VP₃</td>
<td></td>
</tr>
<tr>
<td>NP₂ → NP₂ PP</td>
<td>VP₃ → VB PP NP₂</td>
<td></td>
</tr>
<tr>
<td>MD → must</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Local Tree Languages

Example (with root label S)

\[
S \rightarrow NP_1 \ VP_2 \\
NP_2 \rightarrow NP_2 \ PP \\
MD \rightarrow \text{must} \\
VP_2 \rightarrow MD \ VP_3 \\
VP_3 \rightarrow VB \ PP \ NP_2 \\
\ldots
\]
Local Tree Languages

Example (with root label S)

\[
S \rightarrow NP_1 \ VP_2 \\
NP_2 \rightarrow NP_2 \ PP \\
MD \rightarrow \text{must} \\

VP_2 \rightarrow \text{MD} \ VP_3 \\
VP_3 \rightarrow \text{VB} \ PP \ NP_2
\]
Local Tree Languages

Example (with root label S)

\[
S \rightarrow NP_1 \ VP_2 \\
NP_2 \rightarrow NP_2 \ PP \\
MD \rightarrow must \\
VP_2 \rightarrow MD \ VP_3 \\
VP_3 \rightarrow VB \ PP \ NP_2
\]

\[
\text{We must bear in mind the Community as a whole}
\]
Local Tree Languages

Example (with root label S)

\[ S \rightarrow NP_1 \ VP_2 \]
\[ NP_2 \rightarrow NP_2 \ PP \]
\[ MD \rightarrow \text{must} \]
\[ VP_2 \rightarrow MD \ VP_3 \]
\[ VP_3 \rightarrow VB \ PP \ NP_2 \]

...
Local Tree Languages

Example (with root label $S$)

$$S \rightarrow NP_1 \ VP_2$$
$$NP_2 \rightarrow NP_2 \ PP$$
$$VP_2 \rightarrow MD \ VP_3$$
$$VP_3 \rightarrow VB \ PP \ NP_2$$
$$MD \rightarrow must$$

We must bear in mind the Community as a whole.

\[
\begin{align*}
S & \quad | \quad NP_1 \quad | \quad VP_2 \\
| & \quad | \quad PRP \quad | \quad MD \quad | \quad VP_2 \\
| & \quad | \quad We \quad | \quad must \quad | \quad VB \quad | \quad PP \\
| & \quad | \quad bear \quad | \quad IN \quad | \quad NP_1 \\
| & \quad | \quad in \quad | \quad NN \\
| & \quad | \quad mind \quad | \quad DT \quad | \quad NN \\
| & \quad | \quad the \quad | \quad Community \\
| & \quad | \quad as \quad | \quad IN \\
| & \quad | \quad a \quad | \quad DT \quad | \quad NN \\
| & \quad | \quad whole
\end{align*}
\]
Local Tree Languages

not closed under union

- 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: X/✓/X)
✗ 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 substitution grammar \[ \text{Joshi, Schabes 1997} \]

Tree substitution grammar = 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 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)

\[ G = (\Sigma, I, P) \text{ with } I \subseteq \Sigma \text{ and finite } P \subseteq T_\Sigma(\Sigma) \]
Tree Substitution Languages

Typical fragments

[Post 2011]

\[ \xi \Rightarrow_G \zeta \]

- \( \xi = c[root(t)] \) and \( \zeta = c[t] \) for some context \( c \) and fragment \( t \in P \)
Tree substitution grammar $G = (\Sigma, I, P)$

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

```
\sigma
```

$L(G(t)) = \{ t \in \Sigma^* | \exists \sigma \in I : \sigma \Rightarrow^* G(t) \}$
Tree substitution grammar $G = (\Sigma, I, P)$

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

- generated tree language

$$L(G) = \{ t \in T_{\Sigma}(\emptyset) \mid \exists \sigma \in I: \sigma \Rightarrow^*_G t \}$$
Fragments

\[ S(\text{NP}_1(\text{PRP}), \text{VP}_2) \] 
\[ \text{PRP(We)} \]

\[ \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2)) \] 
\[ \text{MD(must)} \]

Derivation

\[ S \]
Tree Substitution Languages

Fragments

\[ S(\text{NP}_1(\text{PRP}), \text{VP}_2) \]

\[ \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2)) \]

\[ \text{PRP(We)} \]

\[ \text{MD(must)} \]

Derivation

\[ S \]
Tree Substitution Languages

Fragments

\[ S(NP_1(\text{PRP}), \text{VP}_2) \]
\[ \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2)) \]

Derivation

\[
S \\
NP_1 \\
| \\
PRP \\
\]

\[
\text{VP}_2 \\
| \\
\text{MD} \\
\]

\[
\text{PRP(We)} \\
\text{MD(must)} \\
\]
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
  / \       /
NP_1  VP_2
  /     /
PRP
```
Fragments

\[ S(NP_1(\text{PRP}), VP_2) \]
\[ VP_2(\text{MD}, VP_3(\text{VB}, \text{PP}, NP_2)) \]

Derivation

```
S
   /\    /
  /  \  /  
NP_1  VP_2
     /    /
    PRP
```
Tree Substitution Languages

Fragments

\[ S(\text{NP}_1(\text{PRP}), \text{VP}_2) \]
\[ \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2)) \]
\[ \text{PRP}(\text{We}) \]
\[ \text{MD}(\text{must}) \]

Derivation

```
S
  /\  \\
 NP_1  VP_2
   /\    \\
  PRP   \\
   /\    \\
  We    \\
```
Tree Substitution Languages

Fragments

\[ S(\text{NP}_1(\text{PRP}), \text{VP}_2) \]
\[ \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2)) \]

Derivation

```
  S
 / \   /
NP_1 PRP VP_2
 |     |
PRP
 |     |
We
```
Tree Substitution Languages

Fragments

\[ S(\text{NP}_1(\text{PRP}), \text{VP}_2) \]
\[ \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2)) \]

Derivation

\[
\begin{array}{c}
\text{S} \\
\text{NP}_1 \\
\text{PRP} \\
\text{We} \\
\text{VP}_2 \\
\end{array}
\]

\[ \text{PRP(We)} \]
\[ \text{MD(must)} \]
Tree Substitution Languages

Fragments

\[ S(\text{NP}_1(\text{PRP}), \text{VP}_2) \]
\[ \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2)) \]

Derivation

[Diagram of the tree structure of the sentence]
Tree Substitution Languages

Fragments

\[ S(NP_1(PRP), VP_2) \]
\[ VP_2(MD, VP_3(VB, PP, NP_2)) \]

Derivation

```
S
   / \   / \   / \\
NP_1 PRP MD VP_3
   |     |   |     |
   PRP MD VB PP NP_2
```

PRP(We)
MD(must)
Tree Substitution Languages

Fragments

\[ S(\text{NP}_1(\text{PRP}), \text{VP}_2) \]
\[ \text{VP}_2(\text{MD}, \text{VP}_3(\text{VB}, \text{PP}, \text{NP}_2)) \]

Derivation

[Diagram showing the derivation process with labels for each node.]
Tree Substitution Languages

Fragments

\[ \text{S(NP}_1\text{(PRP), VP}_2\text{)} \]
\[ \text{VP}_2\text{(MD, VP}_3\text{(VB, PP, NP}_2\text{))} \]

Derivation

```
S
  /\  /
 /\ /\  /
/\ /\ /\  /
PRP MD VP3
  /\  /
 /\ /\  /
/\ /\ /\  /
We must VB PP NP2
```
Tree Substitution Languages

Fragments

\[ S(NP_1(PRP), VP_2) \]
\[ VP_2(MD, VP_3(VB, PP, NP_2)) \]

Derivation

```
                               S
                              /   \
                             NP_1   VP_2
                              |     /
                             PRP   MD
                              |   /  \
                             We  must VP_3
                              |    |   /
                             VB  PP  NP_2
```
Tree Substitution Languages

not closed under union

- 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

not closed under union

- 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)
not closed under intersection

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

- 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

not closed under complement

- this language $L$ is a tree substitution language

\[
\begin{align*}
S & \rightarrow A & S & \rightarrow B \\
A & \in L & B & \in L \\
A' & & B' & \\
A' & a & b & \\
& & & \\
A' & a & b & \\
& & & \\
S & \rightarrow A & S & \rightarrow B \\
A & \notin L & B & \notin L \\
A' & & A' & \\
A' & b & a & \\
& & & \\
A' & b & a & \\
& & & \\
\end{align*}
\]

- 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)
Open questions

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

Thank you for your attention!
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
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!
## Experiment

<table>
<thead>
<tr>
<th>Grammar</th>
<th>Size</th>
<th>Prec.</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>local</td>
<td>46k</td>
<td>75.37</td>
<td>70.05</td>
<td>72.61</td>
</tr>
<tr>
<td>“spinal” TSG</td>
<td>190k</td>
<td>80.30</td>
<td>78.10</td>
<td>79.18</td>
</tr>
<tr>
<td>“minimal subset” TSG</td>
<td>2,560k</td>
<td>76.40</td>
<td>78.29</td>
<td>77.33</td>
</tr>
</tbody>
</table>

(On WSJ Sect. 23)
<table>
<thead>
<tr>
<th>Grammar</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
</tr>
<tr>
<td>TSG [Post, Gildea, 2009]</td>
<td>82.6</td>
</tr>
<tr>
<td>TSG [Cohn et al., 2010]</td>
<td>85.4</td>
</tr>
<tr>
<td>CFGlv [Collins, 1999]</td>
<td>88.6</td>
</tr>
<tr>
<td>CFGlv [Petrov, Klein, 2007]</td>
<td>90.6</td>
</tr>
<tr>
<td>CFGlv [Petrov, 2010]</td>
<td></td>
</tr>
<tr>
<td>TSGlv (single)</td>
<td>91.6</td>
</tr>
<tr>
<td>TSGlv (multiple)</td>
<td>92.9</td>
</tr>
</tbody>
</table>

**Discriminative Parsers**

<table>
<thead>
<tr>
<th>Parsers</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carreras et al., 2008</td>
<td>91.1</td>
</tr>
<tr>
<td>Charniak, Johnson, 2005</td>
<td>92.0</td>
</tr>
<tr>
<td>Huang, 2008</td>
<td>92.3</td>
</tr>
</tbody>
</table>