Characterizations of subregular tree languages

Andreas Maletti

Institute of Computer Science, Universität Leipzig, Germany

andreas.maletti@uni-leipzig.de

MPI, Leipzig — November 29, 2018
We must bear in mind the Community as a whole
For sets $\Sigma$ and $V$ and $rk: \Sigma \to \mathbb{N}$, let $T_{(\Sigma, rk)}(V)$ be the least set $T$ s.t.

1. $V \subseteq T$
2. $\sigma(t_1, \ldots, t_{rk(\sigma)}) \in T$ for all $\sigma \in \Sigma$ and $t_1, \ldots, t_{rk(\sigma)} \in T$
For sets $\Sigma$ and $V$ and $\text{rk}: \Sigma \rightarrow \mathbb{N}$, let $T_{(\Sigma,\text{rk})}(V)$ be the least set $T$ s.t.

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

- 2nd item: top concatenation
- ‘rk’ often implicit (we often write $T_{\Sigma}(V)$ instead of $T_{(\Sigma,\text{rk})}(V)$)
- $(\Sigma)$-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

Definition (Regular tree language [Brainerd 1984])

$L \subseteq T^\Sigma(\emptyset)$ is regular if there exists a congruence $\sim = (\text{top-concatenation})$ on $T^\Sigma(\emptyset)$ such that:

1. $\sim = \text{has finite index (finitely many equivalence classes)}$
2. $\sim$ saturates $L$; i.e. $L = \bigcup_{t \in L} [t] \sim$
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. $L = \bigcup_{t \in L} [t]_{\cong}$
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 $\sigma$”, “some $\sigma$, but legal”, illegal)

  $$L' = \{t \in T_\Sigma(\emptyset) \mid \sigma \text{ never below } \delta\}$$
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 $\Sigma$
- finite set of productions $P \subseteq T_{\Sigma}(Q) \times Q$
  (we write $r \rightarrow q$ for productions $(r, q)$)

Example productions

1. $V P_3 \xrightarrow{q_5} N P_1 \xrightarrow{q_3} q_4$
2. $S \xrightarrow{NP_1 q_4} q_0$
3. $S \xrightarrow{q_6} V P_2 \xrightarrow{q_2 q_4} q_0$
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)

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

$C_1 = [\alpha]$ \hspace{1cm} $C_2 = [\sigma(\alpha, \alpha)]$ \hspace{1cm} $C_3 = [\delta(\sigma(\alpha, \alpha), \alpha)]$
Regular Tree Languages

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

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

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

Productions with nonterminals \( C_1, C_2, C_3 \)

\[
\begin{align*}
\alpha & \rightarrow C_1 \\
\delta(C_1, C_1) & \rightarrow C_1 \\
\sigma(C_1, C_1) & \rightarrow C_2 \\
\sigma(C_1, C_2) & \rightarrow C_2 \\
\sigma(C_2, C_1) & \rightarrow C_2 \\
\sigma(C_2, C_2) & \rightarrow C_2 \\
\delta(C_1, C_2) & \rightarrow C_3 \\
\delta(C_1, C_3) & \rightarrow C_3 \\
\delta(C_2, C_1) & \rightarrow C_3 \\
\delta(C_2, C_2) & \rightarrow C_3 \\
\delta(C_2, C_3) & \rightarrow C_3 \\
\delta(C_3, C_1) & \rightarrow C_3 \\
\delta(C_3, C_2) & \rightarrow C_3 \\
\delta(C_3, C_3) & \rightarrow C_3 \\
\sigma(C_1, C_3) & \rightarrow C_3 \\
\sigma(C_2, C_3) & \rightarrow C_3 \\
\sigma(C_3, C_1) & \rightarrow C_3 \\
\sigma(C_3, C_2) & \rightarrow C_3 \\
\sigma(C_3, C_3) & \rightarrow C_3
\end{align*}
\]
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
- second-order logic formulas
- ...

Tree Languages

Representations

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

\[ G = (\Sigma, I, P) \text{ with } I \subseteq \Sigma \text{ and } P \subseteq \bigcup_{k \in \mathbb{N}} rk^{-1}(k) \times \Sigma^k \]
Local Tree Languages

<table>
<thead>
<tr>
<th>Example (with root label S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S  →  NP₁ VP₂</td>
</tr>
<tr>
<td>NP₂  →  NP₂ PP</td>
</tr>
<tr>
<td>MD  →  must</td>
</tr>
<tr>
<td>VP₂  →  MD VP₃</td>
</tr>
<tr>
<td>VP₃  →  VB PP NP₂</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Local Tree Languages

Example (with root label S)

\[
\begin{align*}
S & \rightarrow NP_1 \ VP_2 \\
NP_2 & \rightarrow NP_2 \ PP \\
MD & \rightarrow must \\
VP_2 & \rightarrow MD \ VP_3 \\
VP_3 & \rightarrow VB \ PP \ NP_2 \\
\ldots
\end{align*}
\]
Local Tree Languages

Example (with root label S)

\[
\begin{align*}
S & \rightarrow NP_1 \ VP_2 \\
NP_2 & \rightarrow NP_2 \ PP \\
MD & \rightarrow \text{must} \\
VP_2 & \rightarrow \text{MD} \ VP_3 \\
VP_3 & \rightarrow \text{VB} \ PP \ NP_2
\end{align*}
\]
Local Tree Languages

Example (with root label S)

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

\[
VP_2 \rightarrow MD \ VP_3 \\
VP_3 \rightarrow VB \ PP \ NP_2 \\
\ldots
\]
Local Tree Languages

Example (with root label S)

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

\[ VP_2 \rightarrow MD \; VP_3 \]
\[ VP_3 \rightarrow \text{VB} \; PP \; NP_2 \]

...
Local Tree Languages

Example (with root label S)

S → NP₁ VP₂
NP₂ → NP₂ PP
MD → must

VP₂ → MD VP₃
VP₃ → VB PP NP₂

S
  / \
NP₁ /   \
PRP MD
  /    \
We must
  /    \
bear IN NP₁
    /   \
in NN DT
    /    \
mind the Community
      /   \
as IN DT
      /    \
a whole
Local Tree Languages

**not closed under union**

- these singletons are local

```
S
  /\  \
NP₂  VP₁
  /\  \\
PRP$ NN VBZ
  |  |  |
My dog sleeps
```

- but their union cannot be local

```
S
  /\  \
NP₂  VP₂
  /\  \\
DT NN VBD \\
|  | scored RB
|  |  | well
```

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

not closed under complement

- this tree language \( L \) is local

\[
\begin{align*}
S & \quad S \\
| & \quad | \\
A & \quad B \\
| & \quad | \\
A' & \quad B' \\
| & \quad | \\
a & \quad b \\
\end{align*}
\]

\( a \in L \)

\( b \in L \)

- but its complement cannot be local
Local Tree Languages

not closed under complement

- this tree language \( L \) is local

\[
\begin{align*}
S & \quad S \\
A & \quad B \\
A' & \in L & B' & \in L
\end{align*}
\]

\[
\begin{align*}
S & \quad S \\
A & \quad B \\
A' & \quad A' \\
\end{align*}
\]

- 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

\[
\begin{align*}
\alpha & \rightarrow C_1 & \delta(C_1, C_1) & \rightarrow C_1 \\
\sigma(C_1, C_1) & \rightarrow C_2 & \sigma(C_1, C_2) & \rightarrow C_2 & \sigma(C_2, C_1) & \rightarrow C_2 & \sigma(C_2, C_2) & \rightarrow C_2
\end{align*}
\]

(irrelevant productions omitted)
Local Tree Languages

\( \alpha \rightarrow C_1 \quad \delta(C_1, C_1) \rightarrow C_1 \quad \) (irrelevant productions omitted)
\( \sigma(C_1, C_1) \rightarrow C_2 \quad \sigma(C_1, C_2) \rightarrow C_2 \quad \sigma(C_2, C_1) \rightarrow C_2 \quad \sigma(C_2, C_2) \rightarrow C_2 \)

1. extract all possible branches and root labels

\( \{ \delta \rightarrow \alpha \alpha, \delta \rightarrow \alpha \delta, \delta \rightarrow \delta \alpha, \delta \rightarrow \delta \delta, \sigma \rightarrow \alpha \alpha, \sigma \rightarrow \alpha \delta, \sigma \rightarrow \delta \alpha, \sigma \rightarrow \delta \delta, \sigma \rightarrow \alpha \sigma, \sigma \rightarrow \delta \sigma, \sigma \rightarrow \sigma \alpha, \sigma \rightarrow \sigma \delta, \sigma \rightarrow \sigma \sigma \} \)
Local Tree Languages

\[\alpha \rightarrow C_1 \quad \delta(C_1, C_1) \rightarrow C_1\]  
(irrelevant productions omitted)

\[\sigma(C_1, C_1) \rightarrow C_2 \quad \sigma(C_1, C_2) \rightarrow C_2 \quad \sigma(C_2, C_1) \rightarrow C_2 \quad \sigma(C_2, C_2) \rightarrow C_2\]

1. extract all possible branches and root labels

\[
\{ \delta \rightarrow \alpha \alpha, \delta \rightarrow \alpha \delta, \delta \rightarrow \delta \alpha, \delta \rightarrow \delta \delta, \\
\sigma \rightarrow \alpha \alpha, \sigma \rightarrow \alpha \delta, \sigma \rightarrow \delta \alpha, \sigma \rightarrow \delta \delta, \\
\sigma \rightarrow \alpha \sigma, \sigma \rightarrow \delta \sigma, \sigma \rightarrow \sigma \alpha, \sigma \rightarrow \sigma \delta, \sigma \rightarrow \sigma \sigma \}  
\]

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
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

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

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

- generated tree language

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

\[ S(NP_1(PRP), VP_2) \]

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

Derivation

\[ S \]
Tree Substitution Languages

Fragments

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

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

Derivation

$S$
Fragments

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

Derivation

```
S
  NP_1
    PRP
  VP_2
    MD
```

PRP(We)

MD(must)
Fragments

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

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(We)}

\text{MD(must)}

Derivation

```
\[ S \]
\[ \text{NP}_1 \quad \text{VP}_2 \]
\[ \left\downarrow \right. \quad \left\downarrow \right. \]
\[ \text{PRP} \]
```
Fragments

\[ S(NP_1(\text{PRP}), \text{VP}_2) \]
\[ \text{VP}_2(\text{MD}, \text{VP}_3(VB, PP, NP_2)) \]
\[ \text{PRP(We)} \]
\[ \text{MD(must)} \]

Derivation

```
S
  NP_1    VP_2
    |    |
   PRP   |
  |  |
We
```
Fragments

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

Derivation

```
    S
   /\  
  NP_1  VP_2
   |     /\   
   PRP  PRP(We)
      /   /\   
     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

\( S \)
\( \text{NP}_1 \)
\( \text{PRP} \)
\( \text{We} \)
\( \text{VP}_2 \)
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
   /\  \
  /   \
 NP1  VP2
    / \  /\  /
   /   /   /
  PRP MD VP3
     /   /  /
    /   /   /
   We VB PP NP2
```
Tree Substitution Languages

Fragments

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

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

Derivation

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

Fragments

\[
\begin{align*}
S&(NP_1(PRP), VP_2) \\
VP_2&(MD, VP_3(VB, PP, NP_2)) \\
PRP&(We) \\
MD&(must)
\end{align*}
\]

Derivation

\[
\begin{align*}
S &\quad NP_1 &\quad VP_2 \\
\quad PRP &\quad MD &\quad VP_3 \\
\quad We &\quad VB &\quad PP &\quad NP_2
\end{align*}
\]
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 VP_2
     /\     /
    PRP MD   VP_3
       /\     /
      We must VB PP NP_2
```
Tree Substitution Languages

Fragments

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

PRP(We)
MD(must)

Derivation

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

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

○ but their union is not
  (exchange subtrees below the indicated cuts)
Tree Substitution Languages

not closed under intersection

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

$$S' \ x_1 \ S \ x_2 \ S \ x_3 \ S \ x_n-1 \ S \ x_n \ c \ \in L_1$$

$$S' \ x_1 \ S \ x_2 \ S \ x_3 \ S \ x_n-1 \ S \ x_n \ c \ \in L_2$$

$$S' \ x_1 \ S \ x_2 \ S \ x_3 \ S \ x_n-1 \ S \ x_n \ c \ \in L_2$$

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 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

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

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

Thank you for your attention!
<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>

[Post, Gildea 2009] (on WSJ Sect. 23)
## Experiment

[Shindo et al. 2012]

<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

| CARRERAS et al., 2008        | 91.1     |
| Charniak, Johnson, 2005     | 92.0     | 91.4  |
| Huang, 2008                 | 92.3     | 91.7  |