Applications of Tree Automata Theory
Lecture III: Parsing — Advanced Topics

Andreas Maletti

Institute of Computer Science
Universität Leipzig, Germany

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

maletti@ims.uni-stuttgar.de

Yekaterinburg — August 24, 2014
Roadmap

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

Always ask questions right away!
Overview

Topics

- Foundations of Tree Automata
- Applications of TA in NLP
  (yielding TAs and algorithms for further study)
- Application of TA theory in NLP
  (solving NLP problems)
Lexicalized Grammars
Lexicon

**Merriam-Webster entry for sleep**

- **intransitive verb**
  1. to rest in a state of sleep
  2. to be in a state (as of quiescence or death) resembling sleep
  3. to have sexual relations — usually used with *with*

- **transitive verb**
  1. to be slumbering in
  2. to get rid of or spend in or by sleep

- slept the sleep of the dead
- sleep away the hours, sleep off a headache
- the boat sleeps six
Linguistic Constructions

Fragment examples:

1. VP
   VB
   sleep

2. VP
   VB
   sleep
   PP

3. VP
   VB
   sleep
   PP
   IN
   with
   NP

4. VP
   VB
   sleep
   NP
Linguistic Constructions

Fragments

Constructions

- major research area for traditional linguistics
Lexicalized Grammars

<table>
<thead>
<tr>
<th>Definition (Meta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A grammar with productions $P$ is <strong>lexicalized</strong> if each production $\rho \in P$ contains at least one lexical item</td>
</tr>
</tbody>
</table>
Lexicalized Grammars

Definition (Meta)

A grammar with productions \( P \) is **lexicalized** if each production \( \rho \in P \) contains at least one lexical item.

Example

- NFAs are lexicalized
- CFGs in GREIBACH normal form are lexicalized
Lexicalized Grammars

Benefits

- ideal for parsing
  - bound on number of applied productions
  - grammar pruning based on occurring lexical items
- linguistic constructions evident
- theoretical advantages
Benefits

- ideal for parsing
  - bound on number of applied productions
  - grammar pruning based on occurring lexical items
- linguistic constructions evident
- theoretical advantages

Disadvantages

- have only finite ambiguity
  
  (for every sentence there are only finitely many parses)
  
  theoretical disadvantage
Weak Lexicalization

classes $\mathcal{G}, \mathcal{G}'$ of string grammars

**Definition**

$\mathcal{G}'$ *(weakly)* lexicalizes $\mathcal{G}$ if for every $G \in \mathcal{G}$ there exists an equivalent lexicalized $G' \in \mathcal{G}'$

**Example**

- CFGs weakly lexicalize themselves
  *(via the GREIBACH normal form)*
Weak Lexicalization

Problem

- weak lexicalization only covers generated string language
- parses can look totally different
  (see GREIBACH normalization)
- weak lexicalization destroys linguistic knowledge present in treebanks
Tree Yield

Definition (Yield)

mapping \( yd : T_\Sigma(Q) \rightarrow Q^* \)

\[
yd(q) = q \\
yd(\sigma(t_1, \ldots, t_k)) = yd(t_1) \cdots yd(t_k)
\]

for all \( q \in Q, \sigma \in \Sigma, \) and \( t_1, \ldots, t_k \in T_\Sigma(Q) \)
Yield: We must bear in mind the Community as a whole
Yield: И вы действительно свою, 20
Finite Ambiguity

Definition

A tree grammar $G$ is **finitely ambiguous** if for every $w \in Q^*$

\[
\text{Parses}(w) = \{ t \in L(G) \mid \text{yd}(t) = w \}
\]

is finite
classes $\mathcal{G}, \mathcal{G}'$ of tree grammars

**Definition**

$\mathcal{G}'$ (strongly) lexicalizes $\mathcal{G}$ if for every finitely ambiguous $G \in \mathcal{G}$ there exists an equivalent lexicalized $G' \in \mathcal{G}'$
Strong Lexicalization

classes $\mathcal{G}, \mathcal{G}'$ of tree grammars

**Definition**

$\mathcal{G}'$ (strongly) lexicalizes $\mathcal{G}$ if for every finitely ambiguous $G \in \mathcal{G}$ there exists an equivalent lexicalized $G' \in \mathcal{G}'$

**Example**

- LTGs strongly lexicalize themselves

[SCHABES, 1990]
classes \( \mathcal{G}, \mathcal{G}' \) of tree grammars

**Definition**

\( \mathcal{G}' \) (strongly) lexicalizes \( \mathcal{G} \) if for every finitely ambiguous \( G \in \mathcal{G} \) there exists an equivalent lexicalized \( G' \in \mathcal{G}' \)

**Example**

- LTGs *cannot* strongly lexicalize themselves

[Schabes, 1990]
Strong Lexicalization

classes $\mathcal{G}, \mathcal{G}'$ of tree grammars

**Definition**

$\mathcal{G}'$ (strongly) lexicalizes $\mathcal{G}$ if for every finitely ambiguous $G \in \mathcal{G}$ there exists an equivalent lexicalized $G' \in \mathcal{G}'$

**Example**

- LTGs cannot strongly lexicalize themselves  
  \[\text{[Schabes, 1990]}\]
- TSGs strongly lexicalize LTGs  
  \[\text{[Schabes, 1990]}\]
Strong Lexicalization

classes $\mathcal{G}, \mathcal{G}'$ of tree grammars

**Definition**

$\mathcal{G}'$ (strongly) lexicalizes $\mathcal{G}$ if for every finitely ambiguous $G \in \mathcal{G}$ there exists an equivalent lexicalized $G' \in \mathcal{G}'$

**Example**

- LTGs cannot strongly lexicalize themselves
  
  [Schabes, 1990]

- TSGs cannot strongly lexicalize LTGs
  
  [Schabes, 1990]
Strong Lexicalization

classes $\mathcal{G}, \mathcal{G}'$ of tree grammars

**Definition**

$\mathcal{G}'$ (strongly) lexicalizes $\mathcal{G}$ if for every finitely ambiguous $G \in \mathcal{G}$ there exists an equivalent lexicalized $G' \in \mathcal{G}'$

**Example**

- LTGs cannot strongly lexicalize themselves
  
  [Schabes, 1990]

- TSGs cannot strongly lexicalize LTGs
  
  [Schabes, 1990]

- RTGs strongly lexicalize LTGs
Strong Lexicalization

classes $\mathcal{G}, \mathcal{G}'$ of tree grammars

**Definition**

$\mathcal{G}'$ (strongly) lexicalizes $\mathcal{G}$ if for every finitely ambiguous $G \in \mathcal{G}$ there exists an equivalent lexicalized $G' \in \mathcal{G}'$

**Example**

- LTGs **cannot** strongly lexicalize themselves
  
  [Schabes, 1990]

- TSGs **cannot** strongly lexicalize LTGs
  
  [Schabes, 1990]

- RTGs **cannot** strongly lexicalize LTGs
Overview

Tree Adjoining Grammars
Tree-Adjoining Grammars

## Motivation

- mildly context-sensitive formalism
- productions express local dependencies
Motivation

- mildly context-sensitive formalism
- productions express local dependencies
- but can realize global dependencies
Tree-Adjoining Grammars

Motivation

- mildly context-sensitive formalism
- productions express local dependencies
- but can realize global dependencies

Applications

- TAG for English [XTAG RESEARCH GROUP, 2001]
- lexicalized TAG for German [KALLMEYER et al., 2010]
Definition (Joshi et al., 1969)

\[ G = (N, \Sigma, S, R) \] tree-adjoining grammar (TAG) with finite set \( R \)

- substitution productions

Substitution production

\[ \Sigma \]

\[ \begin{array}{c}
NP \\
of \\
NP
\end{array} \]
Definition (Joshi et al., 1969)

$G = (N, \Sigma, S, R)$ tree-adjoining grammar (TAG) with finite set $R$

- substitution productions
- adjunction productions

Substitution production

Adjunction production
Tree-Adjoining Grammars

S
Tree-Adjoining Grammars

Used substitution production

Lecture III: Lexicalization
Tree-Adjoining Grammars

Lecture III: Lexicalization

Used substitution production

S

NP¹

VP

V

NP²

VP

V

NP
Tree-Adjoining Grammars

S

\[ \text{V} \]

\[ \text{NP}^1 \]

\[ \text{VP} \]

\[ \text{likes} \]

\[ \text{NP}^2 \]

Used substitution production

\[ \text{V} \]

\[ \text{likes} \]
Tree-Adjoining Grammars

```
S → NP VP
V → "likes" NP
```

Used substitution production

```
NP → N
```
Tree-Adjoining Grammars

S

NP

VP

V

likes

N

candies

Used substitution production

N

candies
Used adjunction production
Tree-Adjoining Grammars

Used substitution production

ADJ
red

NP
VP
S
V
likes
NP
ADJ
N
red
candies
Tree-Adjoining Grammars

Definition (generated tree language)

\[ L(G) = \bigcup_{A \in S} \{ t \in T_\Sigma \mid A \Rightarrow^*_G t \} \]
Tree-Adjoining Grammars

Productions

\[
S \\
  \downarrow \\
  c
\]

\[
S \\
  \downarrow \\
  S
\]

\[
S \\
  \downarrow \\
  a \\
  S
\]

\[
S \\
  \downarrow \\
  S^* \\
  a
\]

\[
S \\
  \downarrow \\
  b \\
  S
\]

\[
S \\
  \downarrow \\
  S^* \\
  b
\]

String language \( L(G) \) = \{wcw | w ∈ \{a, b\}∗\}
Tree-Adjoining Grammars

Productions

\[
\begin{align*}
S & \rightarrow aS \\
S & \rightarrow cS \\
S & \rightarrow bS \\
S & \rightarrow S^* \\
S^* & \rightarrow aS^* \\
S^* & \rightarrow bS^*
\end{align*}
\]

String language \( yd(L(G)) = \{wcw | w \in \{a, b\}^*\} \)
Tree-Adjoining Grammars

Productions

```
S  
a

b

S

S

S

S

S

S

a

b

c
```

String language $y_d(L(G)) = \{wcw | w \in \{a, b\}^*\}$
Tree-Adjoining Grammars

String language

\[ yd(L(G)) = \{ wcw \mid w \in \{a, b\}^* \} \]
Tree-Adjoining Grammars

Theorem (Schabes, 1990)

TAGs can strongly lexicalize LTGs and themselves
Tree-Adjoining Grammars

Theorem (Schabes, 1990 and Kuhlmann, Satta, 2012)

TAGs can strongly lexicalize LTGs and themselves
but not themselves
Tree-Adjoining Grammars

Theorem (Schabes, 1990 and Kuhlmann, Satta, 2012)

TAGs can strongly lexicalize LTGs and themselves
but not themselves

Widespread myth

Kallmeyer: Parsing beyond context-free grammars
Springer, 2010

Joshi, Schabes
Tree-adjoining grammars

Joshi, Schabes
Tree-adjoining grammars and lexicalized grammars
In Tree Automata and Languages, North-Holland, 1992
Context-free Tree Grammar

Definition (ROUNDS, 1969)

\[(N, \Sigma, S, P)\] context-free tree grammar (CFTG)

- alphabet \(N\)
- alphabet \(\Sigma\)
- \(S \subseteq N\)
- \(P\) is a finite set of \(A(x_1, \ldots, x_k) \rightarrow r\)
  - \(A \in N\)
  - \(r \in C_{N \cup \Sigma}(\{x_1, \ldots, x_k\})\)
Context-free Tree Grammar

Example

CFTG \( (N, \Sigma, \{S\}, P) \) with
- \( N = \{S, A\} \)
- \( \Sigma = \{\alpha, \beta, \sigma\} \)

Productions

\[
\begin{align*}
S & \rightarrow A(\alpha, \alpha) \mid A(\beta, \beta) \mid \sigma(\alpha, \beta) \\
A(x_1, x_2) & \rightarrow A(\sigma(x_1, S), \sigma(x_2, S)) \\
A(x_1, x_2) & \rightarrow \sigma(x_1, x_2)
\end{align*}
\]
Context-free Tree Grammar

### Example

CFTG \((N, \Sigma, \{S\}, P)\) with
- \(N = \{S, A\}\)
- \(\Sigma = \{\alpha, \beta, \sigma\}\)

### Productions

\[
\begin{align*}
S & \rightarrow A(\alpha, \alpha) \mid A(\beta, \beta) \mid \sigma(\alpha, \beta) \\
A(x_1, x_2) & \rightarrow A(\sigma(x_1, S), \sigma(x_2, S)) \\
A(x_1, x_2) & \rightarrow \sigma(x_1, x_2)
\end{align*}
\]
Context-free Tree Grammar

Example

CFTG \((N, \Sigma, \{S\}, P)\) with
- \(N = \{S, A\}\)
- \(\Sigma = \{\alpha, \beta, \sigma\}\)

Production:
\[
S \rightarrow A(\alpha, \alpha) \mid A(\beta, \beta) \mid \sigma(\alpha, \beta)
\]
\[
A(x_1, x_2) \rightarrow A(\sigma(x_1, S), \sigma(x_2, S))
\]

Lecture III: Lexicalization
Context-free Tree Grammar

Productions

Derivation:
Context-free Tree Grammar

Productions

\[ S \to A \alpha \alpha \]
\[ A \alpha \alpha \to \sigma \]
\[ A \beta \beta \to \sigma \]
\[ A \alpha \beta \to \sigma \]
\[ A x_1 x_2 \to \sigma \]

Derivation:

\[ S \to A \]
\[ A \alpha \alpha \to \sigma \]
\[ \sigma \to \sigma \]

Lecture III: Lexicalization
Context-free Tree Grammar

Productions

\[
\begin{align*}
S & \rightarrow A\alpha\alpha \mid A\beta\beta \mid A\sigma \\
A & \rightarrow x_1\alpha x_2\beta \\
S & \rightarrow \sigma S
\end{align*}
\]

Derivation:

\[
S \Rightarrow_G A \alpha \alpha \Rightarrow_G A \beta \beta \Rightarrow_G A \sigma \Rightarrow_G \sigma S \Rightarrow_G \sigma S
\]
Context-free Tree Grammar

Productions

\[ S \rightarrow A \alpha \alpha \mid A \beta \beta \mid \sigma \]

\[ A \sigma x_1 x_2 \rightarrow \sigma S x_2 S \mid \sigma x_1 x_2 \]

Derivation:

\[ S \Rightarrow_G A \alpha \alpha \Rightarrow_G A \sigma \sigma \Rightarrow_G A \alpha S \alpha S \Rightarrow_G A \beta \beta S \]

Lecture III: Lexicalization
Context-free Tree Grammar

Productions

\[
S \rightarrow A \quad \quad A \rightarrow A \quad \quad A \rightarrow \sigma \quad \quad A \rightarrow \alpha \beta \quad \quad S \rightarrow \sigma \quad \quad S \rightarrow \alpha x_1 x_2 S \quad \quad S \rightarrow \sigma x_1 x_2 S
\]

Derivation:

[Diagram showing derivations of the productions]
Context-free Tree Grammar

Productions

\[ S \rightarrow A \alpha \alpha \bigg| A \beta \beta \bigg| \sigma \]

\[ A \xrightarrow{\sigma} \]

\[ S \xrightarrow{\sigma} \]

Derivation:

\[ S \xrightarrow{G} \]

\[ A \alpha \alpha \xrightarrow{G} \]

\[ A \beta \beta \xrightarrow{G} \]

\[ S \alpha \alpha \xrightarrow{G} \]

\[ \sigma \xrightarrow{G} \]

\[ A \alpha \alpha \xrightarrow{G} \]

\[ A \beta \beta \xrightarrow{G} \]

\[ S \alpha \alpha \xrightarrow{G} \]

\[ A \alpha \alpha \xrightarrow{G^*} \]

\[ A \alpha \alpha \xrightarrow{G} \]

\[ \sigma \xrightarrow{G} \]

\[ \beta \beta \xrightarrow{G} \]

\[ \alpha \beta \xrightarrow{G} \]
Context-free Tree Grammar

Definition (generated language)

\[ L(G) = \bigcup_{A \in S} \{ t \in T_\Sigma \mid A \Rightarrow^*_G t \} \]
Theorem (JOSHI et al., 1975 and MÖNNICH, 1997)

For every (non-strict) TAG there is an equivalent CFTG

Adjunction production

```
      N
   /   \
ADJ    N
```

Corresponding CFTG production

```
N
x_1 \rightarrow ADJ
      N
   /   \
x_1
```
## Context-free Tree Grammar

### Definition

CFTG \((N, \Sigma, S, P)\) **start-separated** if

- it has a single start nonterminal
- the start nonterminal \(A \in S\) does *not* occur in the right-hand sides of \(P\)

\(|S| = 1\)
Context-free Tree Grammar

Definition

CFTG \((N, \Sigma, S, P)\) start-separated if
- it has a single start nonterminal \(|S| = 1\)
- the start nonterminal \(A \in S\) does not occur
  in the right-hand sides of \(P\)

Theorem

For every CFTG there is an equivalent start-separated CFTG
Definition

CFTG \((N, \Sigma, S, P)\) start-separated if

- it has a single start nonterminal \(|S| = 1\)
- the start nonterminal \(A \in S\) does not occur in the right-hand sides of \(P\)

Theorem

For every CFTG there is an equivalent start-separated CFTG

Proof.

- add new start nonterminal \(A'\) \(S' = \{A'\}\)
- add new production \(A' \rightarrow A\) for every \(A \in S\)
**Definition**

CFTG **growing** if non-initial productions contain \( \geq 3 \) non-variables

**Example**

\[
\begin{array}{c}
A \\
\downarrow \quad \downarrow \\
\sigma \quad \sigma \\
S \quad S \\
\downarrow \quad \downarrow \\
x_1 \quad x_1 \\
x_2 \quad x_2
\end{array}
\]

- **growing**
- **not growing**
Definition

CFTG \textit{growing} if non-initial productions contain $\geq 3$ non-variables

Example

\begin{align*}
A & \rightarrow \sigma \\
S & \rightarrow x_1 \sigma S x_2 \\
\end{align*}

Theorem (STAMER, OTTO, 2007)

For every CFTG there is an equivalent growing CFTG
Context-free Tree Grammar

growing CFTG

CFTG
Context-free Tree Grammar

Example

\[ S \rightarrow A \quad \left| \quad A \rightarrow \sigma \alpha \quad \left| \quad \beta \quad \beta \right. \right. \]

\[ S \quad \left| \quad A \rightarrow \sigma \quad \left( x_1 \quad x_2 \right) \quad \left| \quad A \rightarrow \sigma \quad \left( x_1 \quad S \quad x_2 \quad S \right) \right. \right. \]
Example

\[
S \rightarrow A \quad | 
\quad A \quad | \quad \sigma
\]

\[
A \quad x_1 \quad x_2 \quad \rightarrow \quad \sigma
\]

Eliminate last production:

\[
S \rightarrow A/\sigma \quad | 
\quad A/\sigma \quad | \quad \sigma
\]

\[
A \quad x_1 \quad x_2 \quad \rightarrow \quad \sigma
\]
Context-free Tree Grammar

CFTG \((N, \Sigma, S, P)\)

<table>
<thead>
<tr>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production (\ell \rightarrow r \in P)</td>
</tr>
<tr>
<td>- monadic if (r) contains (\leq 1) nonterminals</td>
</tr>
<tr>
<td>- terminal if (r) contains 0 nonterminals</td>
</tr>
</tbody>
</table>
CFTG \((N, \Sigma, S, P)\)

<table>
<thead>
<tr>
<th>Definition</th>
<th>Production</th>
<th>( \ell \rightarrow r \in P )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>monadic</strong> if ( r ) contains ( \leq 1 ) nonterminals</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>terminal</strong> if ( r ) contains 0 nonterminals</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>lexicalized</strong> if ( r ) contains ( \geq 1 ) lexical items</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>doubly lexicalized</strong> if ( r ) contains ( \geq 2 ) lexical items</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Theorem (Engelfriet, ~, 2012)

For every CFTG with finite ambiguity there is an equivalent CFTG such that
- all (non-initial) monadic productions are lexicalized
- all (non-initial) terminal productions are doubly lexicalized
Theorem (ENGELFRIET, ~, 2012)

For every CFTG with finite ambiguity there is an equivalent CFTG such that

- all (non-initial) monadic productions are lexicalized
- all (non-initial) terminal productions are doubly lexicalized

Proof.

- similar to removal of \(\varepsilon\)-productions [HOPCROFT et al., 2001]
- compute closure under non-lexicalized productions
Context-free Tree Grammar

fully normalized CFTG

growing CFTG

CFTG
Theorem (ENGELFRIET, ~, 2012)

Every CFTG with finite ambiguity can be strongly lexicalized
Lexicalization

1. guess lexical item in non-lexicalized production

Input

\[
A \xrightarrow{x_1 \ x_2} A_{\sigma} \xrightarrow{x_1 S \ x_2 S} S
\]

Output

\[
\langle A, \alpha \rangle \xrightarrow{x_1 \ x_2} A_{\sigma} \xrightarrow{x_1 S \ x_2 S} S
\]
Lexicalization

1. guess lexical item in non-lexicalized production
2. transport guessed lexical item

Input

Output
Lexicalization

1. guess lexical item in non-lexicalized production
2. transport guessed lexical item
3. potentially guess again

Input

\[ \langle A, \alpha \rangle \quad x_1 \quad x_2 \quad x_3 \rightarrow \quad \langle A, \alpha \rangle \quad \sigma \quad \sigma \quad x_3 \quad x_1 \quad S \quad x_2 \quad S \]

Output

\[ \langle A, \alpha \rangle \quad x_1 \quad x_2 \quad x_3 \rightarrow \quad \langle A, \alpha \rangle \quad \sigma \quad \sigma \quad x_3 \quad x_1 \quad \langle S, \beta \rangle \quad x_2 \quad S \]

Lecture III: Lexicalization
Lexicalization

1. guess lexical item in non-lexicalized production
2. transport guessed lexical item
3. potentially guess again
4. cancel in terminal production

Input

\[ S \rightarrow \alpha \sigma \alpha \]

Output

\[ \langle S, \alpha \rangle \rightarrow \sigma \]

\[ x_1 \]

\[ x_1 \alpha \]
Lexicalization

lexicalized CFTG

fully normalized CFTG

growing CFTG

CFTG
Lexicalization

After lexicalization (with $\delta, \delta' \in \{\alpha, \beta\}$)

$$S \rightarrow \frac{A/\sigma}{\alpha} \frac{A/\sigma}{\beta} \frac{\sigma}{\alpha} \frac{\sigma}{\beta}$$

$$A \xrightarrow{\sigma} \frac{A/\sigma}{\sigma} \xrightarrow{S} \frac{S}{x_1} \frac{S}{x_2}$$

$$A \xrightarrow{\sigma} \frac{A/\sigma}{\sigma} \xrightarrow{\langle S, \delta \rangle} \frac{\langle S, \delta \rangle}{x_1} \frac{\langle S, \delta \rangle}{x_2} \frac{\langle S, \delta' \rangle}{x_3}$$
CFTG(\(k\)): CFTG with nonterminals of rank \(\leq k\)

**Theorem (ENGELFRIET et al. 1980)**

*CFTG*(\(k\)) induces infinite hierarchy of string languages
CFTG($k$): CFTG with nonterminals of rank $\leq k$

- CFTG($k+1$)
- CFTG($k$)
- \ldots
- CFTG(2)
- CFTG(1) includes TAG
- CFTG(0) regular tree grammar
CFTG\(k\): CFTG with nonterminals of rank \(\leq k\)

CFTG\((k + 1)\) strongly lexicalized by CFTG\(k\)

Corollary

\[\text{CFTG}(k + 1) \text{ strongly lexicalize } \text{CFTG}(k)\]

Corollary

\[\text{CFTG}(2) \text{ strongly lexicalize TAGs}\]

CFTG\(k\) includes TAG

CFTG\(1\) includes TAG

CFTG\(0\) includes TAG

regular tree grammar
CFTG\((k)\): CFTG with nonterminals of rank \(\leq k\)

- CFTG\((k + 1)\) strongly lexicalized by CFTG\((k)\)
- CFTG\((2)\) strongly lexicalize TAGs
- CFTG\((1)\) includes TAG
- CFTG\((0)\) regular tree grammar

Open problem
Rank increase necessary?
Selected references

**JOSHI, LEVY, TAKAHASHI**: *Tree Adjunct Grammars*  

**KUHLMANN, SATTA**  
*Tree-adjoining Grammars are not Closed under Strong Lexicalization.*  

**ROUNDS**: *Context-free Grammars on Trees*  
Proc. 1st STOC 1969

**SCHABES**: *Mathematical and Computational Aspects of Lexicalized Grammars*  