Syntactic Theory
Tree-Adjoining Grammar (TAG)

Yi Zhang

Department of Computational Linguistics
Saarland University

November 10th, 2009
Outline

Tree-Adjoining Grammar (TAG)

Adding Constraints to TAG

Formal Properties of TAG

Linguistic Relevance of TAG

Variants of TAG
**Introducing Auxiliary Trees**

**Auxiliary trees** are the other type of elementary structures in TAG

- interior nodes labeled by non-terminal symbols
- frontier nodes labeled by terminal and non-terminal symbols
- non-terminal nodes on the frontier of the auxiliary tree are marked for substitution except for one node, called the **foot node** (and conventionally noted with (∗))
**Adjoining Operation**

**Adjoining** (or **adjunction**) builds a new tree from an auxiliary tree $\beta$ and a tree $\alpha$ (initial, auxiliary or derived tree) by cutting $\alpha$ into two parts and inserting $\beta$ in between:

- The node of the root of the auxiliary tree is identified with the node $Z$.
- The node of the foot of the auxiliary tree is identified with the root of the excised tree.

![Diagram showing the adjoining operation](image)
Finer Details of the Operations

- $Z$ must not be a substitution node (non-terminal node on the tree frontier)
- the sub-tree dominated by $Z$ is excised, leaving a copy of $Z$ behind
- When a node is marked for substitution, only trees derived from initial trees can be substituted for it
A Tree-Adjoining Grammar (TAG) is a quintuple $(\Sigma, NT, I, A, S)$, where

1. $\Sigma$ is a finite set of terminal symbols
2. $NT$ is a finite set of non-terminal symbols: $\Sigma \cap NT = \emptyset$
3. $S$ is a distinguished non-terminal symbol: $S \in NT$
4. $I$ is a finite set of initial trees
5. $A$ is a finite set of auxiliary trees
Derived Tree & Derivation Tree in TAG

- Derived Tree is the result of the derivations and represents the phrase structure.
- Derivation Tree specifies how a derived tree was constructed.
  - The root is labeled by an S-type initial tree.
  - All other nodes are labeled by initial trees in the cases of substitutions, and auxiliary trees in the cases of adjoining.
  - A tree address is associated with each node (except for the root) to denote the node in the parent tree to which the derivation operation has been performed.
Derived Tree & Derivation Tree: Example

For TAG $\mathcal{G}$:

$\mathcal{G} = (\{john, lyn, really, likes\}, \{S, NP, VP, V\}, \{\alpha_1, \alpha_2, \alpha_3\}, \{\beta_1\}, \{S\})$

with the following elementary trees:

```
\begin{center}
\begin{tikzpicture}
  \tikzstyle{level 1}=[sibling distance=3.5cm]
  \tikzstyle{level 2}=[sibling distance=2cm]
  \tikzstyle{level 3}=[sibling distance=1.5cm]

  \node (S) {$S$}
      child {node (NP) {$NP$} child {node (V) {$V$} child {node (likes) {$likes$}}}}
      child {node (VP) {$VP$}
        child {node (John) {$John$}}
        child {node (Lyn) {$Lyn$}}
        child {node (really) {$really$}}
      }
      child {node (alpha_1) {$\alpha_1$}}

  \end{tikzpicture}
\end{center}
```
Derived Tree:  
```
S  
  NP  VP
  |    |
John really VP
    |VP   |
    V NP
    |      |
    likes Lyn
```  

Derivation Tree:  
```
α1
  α2(1) α3(2 · 2) β1(2)
```
Addresses in Derivation Trees

- root node has address 0
- $k$ is the address of the $k^{th}$ child of the root node
- $p \cdot q$ is the address of the $q^{th}$ child of the node at address $p$
Outline

Tree-Adjoining Grammar (TAG)

Adding Constraints to TAG

Formal Properties of TAG

Linguistic Relevance of TAG

Variants of TAG
In the TAG shown so far, an auxiliary tree $\beta$ can be adjoined on any node $n$, if:

- $n$ has the identical label of the root in $\beta$
- $n$ is not annotated for substitution

It is convenient for linguistic description to have more precision for specifying which auxiliary trees can be adjoined at a given node.
Adjoining Constraints

- **Selective Adjunction** ($SA(T)$): only members of a set $T \subseteq A$ can be adjoined on the given node, but the adjunction is not mandatory.

- **Null Adjunction** ($NA$): any adjunction is disallowed for the given node ($NA = SA(\Phi)$).

- **Obligatory Adjunction** ($OA(T)$): an auxiliary tree member of the set $T \subseteq A$ must be adjoined on the given node.
  - for short $OA = OA(A)$.
One possible analysis of “send” could involve selective adjunction:

\[
\begin{align*}
S & \quad \alpha_1 \\
NP & \quad VP & \quad \beta_1 \\
\text{send} & \quad \text{VP} & \quad \text{VP}^* \\
\text{NP} & \quad \text{VP} & \quad \text{PP} \\
\end{align*}
\]
Obligatory Adjunction: An Example

For when you absolutely must have adjunction at a node:

```
α
S
NP↓VP OA(β₁, β₂) V seen

β₁
VP
Aux has

β₂
VP
Aux is
```
Outline

Tree-Adjoining Grammar (TAG)

Adding Constraints to TAG

Formal Properties of TAG

Linguistic Relevance of TAG

Variants of TAG
Mildly Context Sensitiveness

- Any CFG can be easily converted into an equivalent TAG that generates the same set of trees
- Languages like \( \{a^n b^n e c^n d^n, \ n \geq 1\} \) cannot be generated by any CFG, but can be properly covered by TAG
Lexicalization of CFG with TAG

Theorem

If $G = (\Sigma, NT, P, S)$ is a finitely ambiguous CFG which does not generate the empty string, then there is a lexicalized TAG $G_{\text{lex}} = (\Sigma, NT, I, A, S)$ generating the same string and tree language as $G$.

- Adjunction is sufficient to lexicalize context-free grammars
- The use of substitution enables one to lexicalize a grammar with more compact TAG
Lexicalization of CFG with $\text{TAG}$

**Theorem**

If $G = (\Sigma, NT, \mathcal{P}, S)$ is a finitely ambiguous CFG which does not generate the empty string, then there is a lexicalized $\text{TAG}$ $G_{\text{lex}} = (\Sigma, NT, I, A, S)$ generating the same string and tree language as $G$.

- Adjunction is sufficient to lexicalize context-free grammars
- The use of substitution enables one to lexicalize a grammar with more compact $\text{TAG}$
Closure of TAG under Lexicalization

Theorem
If $G$ is a finitely ambiguous TAG that uses substitution and adjunction as combining operation, s.t. $\lambda \notin L(G)$, then there exists a lexicalized TAG $G_{\text{lex}}$ which generates the same string and tree language as $G$.
Other Formal Properties of TAG and TAL

- $\text{CFL} \subseteq \text{TAL} \subseteq \text{Indexed Languages} \subseteq \text{CSL}$
- TAL is characterized by embedded push-down automaton (EPDA)
- TAL can be parsed in polynomial time ($O(n^6)$ in worst case)
- TAG, HG, LIG and CCG are weakly equivalent
References I

Tree-adjoining grammars.