Tree-Adjoining Grammars

References:
- Aravind Joshi & Yves Schabes, "Tree-Adjoining Grammars"
- Anthony Kroch & Aravind Joshi, "The Linguistic Relevance of TAG"
- Vijay-Shanker & Joshi, "Feature Structure Based TAGs" (COLING-88)
- XTAG Online Tutorial: http://www.cis.upenn.edu/xtag/tutorial.html

February 5, 2004

Alon Lavie

Tree-Adjoining Grammars

II-722 Grammar Formalisms
In a TAG, a derived tree does not contain enough information to determine how it was constructed. A derivation tree is necessary to uniquely specify how a derived tree was constructed. Derivation trees do not contain enough information to determine how they were constructed. A derivation tree is necessary to uniquely specify how a derived tree was constructed. Every node (except the root) is also associated with the address of the node in the parent tree where the composition operation took place. Each tree adjoined to their parent tree are linked by a solid line. Trees substituted into their parent tree are linked by a dashed line. Trees adjoined to their parent tree are linked by a solid line. Trees substituted into their parent tree are linked by a dashed line.
Derivation Example

Initial Trees

Auxiliary Tree
Derivation Example

\[ \alpha_n x_0 V_n x_1 \] 

\[ \alpha_N x_N [pasta] (2) \] 

\[ \beta_v x_A R [quickly] (2) \]

\[ \alpha_N x_N [Maya] (1) \]

Diagram:

```
Tree:   derived-tree-130118
```

```
S
  NP
    N
      pasta
    VP
      VP
        NP
          V
            eats
        VP
          NP
            N
              Maya
          VP
            AD
              quickly
        NP
```

```
Derived Tree
```

```
Derived Tree Example
```
Derivation Example
Derivation Example: Non-compositional
Long Distance Dependencies: WH Questions
Tree Adjoining Languages (TALs)

The tree-set $\mathcal{L}(G)$ of a TAG $G$ is the set of S-rooted derived trees that have a yield consisting of all terminals $\{ w \mid * \in T \in w \}$.

The language of a TAG $G$ is the set of all yields of trees in $\mathcal{L}(G)$.
Tree-adjoining languages (TALs) have the following properties:

- CFLs are strictly included in TALs (CFL ⊂ TAL)
- TALs are closed under union, concatenation, and Kleene-star (in fact, TALs are closed under union, concatenation, and Kleene-star)
- There is a pumping lemma for TALs
- A variant of the push-down automaton called an embedded pushdown automaton (EPDA) corresponds exactly to TALs
Lexicalized Grammars

A grammar is lexicalized if it consists of:

- A finite set of finite structures each associated with a lexical item
- Operation(s) for composing the structures
- An anchor of the structure and must be over (i.e., not the empty string)

The composition operation(s) must combine a finite set of structures into a finite number of structures.

Lexicalized TAG is a TAG in which at least one terminal symbol (the anchor) appears on the frontier of every elementary tree.

A weak lexicalization produces a grammar that accepts the same language as the original but does not generate the same set of trees.

A strong lexicalization produces a grammar that accepts the same language and generates the same treeset as the original grammar.

A lexicalized TAG is a TAG in which at least one terminal symbol appears on the frontier of every elementary tree.
Motivation for Lexicalized Grammars

Lexicalized grammars are definitely ambiguous (no sentence of finite length can be analyzed in an infinite number of ways). Although the worst-case complexity of parsing is not improved, input string input string

- Finite ambiguity guarantees that the recognition problem is decidable
- Lexicalization naturally leads to a two-step parsing strategy:
  1. Select the set of structures associated with items in the input string
  2. Parse the input string with respect to this set of structures

The finite ambiguity of lexicalized grammars guarantees that the length of sentences can be analyzed in an infinite number of ways (no sentence of finite length can be analyzed in an infinite number of ways). Lexicalized grammars are inherently ambiguous (no sentence of finite length can be analyzed in an infinite number of ways).
Use of substitution allows for the creation of more compact TAGs.

Composition operation for $G_{lex}$ is adjunction. (Proof: pp. 19-21)

Furthermore, the only necessary same language and tree set as $G$. Furthermore, the only necessary the empty string, there is a lexicalized TAG $G_{lex}$ that generates the

Theorem: For any finitely ambiguous CFG $G$ that does not generate

Substitution alone is insufficient to lexicalize CFGs

A CFG is equivalent to a TSG where all trees have depth 1

substitution is the only composition operator

A Tree-Substitution Grammar is a Grammar over trees where

Creibach Normal Form is a weak lexicalization of CFGs

anchor (on the right side of every rule

A lexically Tagged CGF would have at least one terminal symbol (the

Only finitely ambiguous CFGs can be lexicalized (no left recursion)
Some Variants of TAGS

- Multi-component TAGS
- Probabilistic TAGS
- Synchronous TAGS
- Feature Structure Based TAGS
Feature Structure Based Lexicalized TAGS

- Feature Structure Based Lexicalized TAGS
- Only finite-valued features are required
- Unity
  - At the end of the derivation: top and bottom $f$-structures of each node must unify
  - Operations - no need for restructuring adjunction anymore
  - The success/failure of unification constrains the applicability of the operations
  - The appropriate nodes
  - Substitution and Adjunction involve unification of the feature structures of the appropriate nodes
  - $f$-structures can be co-indexed among nodes (meaning: they are the same)
  - Other nodes have a single $f$-structure
  - Substitution during the adjunction
  - Two $f$-structures (top and bottom), expressing constraints that must be satisfied during the adjunction
  - Nodes in initial trees that are candidates for adjunction are associated with NT nodes in the trees are associated with feature-structures

XTAG grammar for English: lexicalized feature-based TAG

Grammar Formalisms
Feature Structures and Unification

Feature Unification

Feature Structures and Unification
Feature Structures and Unification

\[
\begin{align*}
& \quad \text{AGR} \quad \text{PERS: 1} \\
& \quad \text{NUM: sing} \\
\hline
& \quad \text{AGR} \quad \text{PERS: 1||3} \\
& \quad \text{NUM: sing} \\
\hline
& = \\
\hline
& \quad \text{AGR} \quad \text{PERS: 1} \\
& \quad \text{NUM: sing}
\end{align*}
\]
Substitution and Unification

Initial Trees and Substitution

\[
\left[ + : \delta \right] = \left[ + : \delta \right] \triangledown \left[ + : J \right]
\]
Auxiliary Trees and Adjunction
Case Assignment Using Features: Example
Case and Agreement Example: "say"
Case and Agreement Example: "she" and "her"
Case and Agreement Example: "say her"
Case and Agreement Example: "she saw her"
Case and Agreement Example: "she saw her" Unified

```
her

NP
  case: 4
  acc
  num: sing
  pers: 3

VP
  assig-case: 1
  case: 1

S
  num: sing
  pers: 3
```

Case and Agreement Example: "she saw her" Unified
Obliqetary Adjunction Example: "tries to leave"
Obligatory Adjunction Example: "tries to leave" Unificed
Obligatory Adjunction Example: "thinks to leave"
Obligatory Adjunction Example: Past Participle
Obligatory Adjunction Example: Past Participle
Obligatory Adjunction Example: Past Participle
Obligatory Adjunction Example: Past Participle
Obligatory Adjunction Example: Past Participle
Obligatory Adjunction Example: Past Participle
Obligatory Adjunction Example: All Features