11-722 Grammar Formalisms

Tree-Adjoining Grammars

Alon Lavie

February 21, 2006

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
Long Distance Dependencies: WH Questions

Fig. 8.4. Elementary Trees for a TAG, G1.

Fig. 8.5. A Derived Tree for the TAG, G2.
Tree Adjoining Languages (TALs)

- The *tree-set* $T(G)$ of a TAG is the set of S-rooted derived trees that have a yield consisting of all terminals.

- The language of a TAG $G$:
  \[
  L(G) = \{ w \in \Sigma^* \mid w \text{ is the yield of some } t \in T(G) \}
  \]
Formal Properties of TALs

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, all closure properties of CFLs hold for TALs)
- 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
- The lexical item with which a structure is associated is called the **anchor** of the structure and must be overt (i.e. not the empty string)
- The composition operation(s) must combine a finite set of structures into a finite number of structures
- A 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 tree set
- A *strong* lexicalization produces a grammar that accepts the same language and generates the same tree set as the original grammar
Motivation for Lexicalized Grammars

- Lexicalized grammars are finitely ambiguous (no sentence of finite length can be analyzed in an infinite number of ways)

- The finite ambiguity of lexicalized grammars guarantees that the recognition problem is decidable

- Lexicalization leads naturally 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
     - Reduces the number of structures to be considered during parsing
     - Constrains the compositions considered during parsing (anchors must appear in the same order in the combined structures as in the input string)
     - Although the worst-case complexity of parsing is not improved (since the whole grammar could be selected), practical performance can be improved
Lexicalization of CFGs

- Only finitely ambiguous CFGs can be lexicalized (no left recursion)
- A lexicalized CFG would have at least one terminal symbol (the anchor) on the right side of every rule
- Greibach Normal Form is a weak lexicalization of CFGs
- A Tree-Substitution Grammar is a grammar over trees where substitution is the only composition operator
- A CFG is equivalent to a TSG where all trees have depth 1
- Substitution alone is insufficient to lexicalize CFGs
- **Theorem:** For any finitely ambiguous CFG $G$ that does not generate the empty string, there is a lexicalized TAG $G_{lex}$ that generates the same language and tree set as $G$. Furthermore, the only necessary composition operation for $G_{lex}$ is adjunction. (Proof: pp. 19-21)
- Use of substitution allows for the creation of more compact TAGs
Some Variants of TAGs

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

- Nodes in the trees are associated with feature-structures
- NT nodes in initial trees that are candidates for adjunction are associated with two f-structures (top and bottom), expressing constraints that must be satisfied during the adjunction
- Other nodes have a single f-structure
- f-structures can be co-indexed among nodes (meaning: they are the same!)
- Substitution and Adjunction involve unification of the feature structures of the appropriate nodes
- The success/failure of unification constrains the applicability of the operations - no need for restricting adjunction anymore
- At the end of the derivation: top and bottom f-structures of each node must unify
- Only finite-valued features are required
- XTAG grammar for English: lexicalized feature-based TAG
Feature Structures and Unification

Feature Unification

\[
\begin{align*}
\begin{bmatrix} f : + \end{bmatrix} \cup \begin{bmatrix} f : + \\ g : + \end{bmatrix} &= \begin{bmatrix} f : + \\ g : + \end{bmatrix} \\
\begin{bmatrix} f : + \end{bmatrix} \cup \begin{bmatrix} g : + \end{bmatrix} &= \begin{bmatrix} f : + \\ g : + \end{bmatrix} \\
\begin{bmatrix} f : + \\ g : + \end{bmatrix} \cup \begin{bmatrix} f : + \\ g : + \end{bmatrix} &= \begin{bmatrix} f : + \\ g : + \end{bmatrix} \\
\begin{bmatrix} f : - \end{bmatrix} \cup \begin{bmatrix} f : + \\ g : + \end{bmatrix} &= \text{fails to unify}
\end{align*}
\]
Feature Structures and Unification

\[
\begin{align*}
\text{AGR} & \begin{cases} 
\text{PERS: 1} \\
\text{NUM: sing}
\end{cases} & \text{AGR} & \begin{cases} 
\text{PERS: 1||3} \\
\text{NUM: sing}
\end{cases} \\
\end{align*}
\]

\[
\begin{align*}
\text{AGR} & \begin{cases} 
\text{PERS: 1} \\
\text{NUM: sing}
\end{cases}
\end{align*}
\]
Substitution and Unification

Initial Trees and Substitution
Adjunction and Unification

Auxiliary Trees and Adjunction
Case Assignment Using Features

\[
\begin{align*}
S_r & \downarrow \\
NP_0 & \downarrow \begin{cases}
\text{case} : <1> \\
\text{agr} : <2>
\end{cases} \\
VP & \downarrow \\
V & \begin{cases}
\text{assign\_case} : <1> \\
\text{agr} : <2>
\end{cases} \\
NP_1 & \uparrow \\
\end{align*}
\]
Case Assignment Using Features: Example

\[
\begin{align*}
S_f & \\
& \quad \downarrow NP_0 \quad \downarrow \text{VP} \\
& \quad \quad \downarrow \text{NP} \quad \downarrow \text{VP} \\
& \quad \quad \quad \downarrow \text{V} \quad \downarrow \text{NP} \\
& \quad \quad \quad \quad \downarrow \text{NP} \\
& \quad \quad \quad \quad \quad \text{admired} \\
\end{align*}
\]
Case and Agreement Example: “saw”

```
S
  [ assign-case : <3> 
    agr : <4> ]

NP
  [ case : <3> 
    agr : <4> ]

VP
  [ assign-case : <3> [ assign-case : <1> 
    agr : <2> ] ]

V

NP
  [ case : acc ]

saw
```
Case and Agreement Example: “she” and “her”

NP[ ]
[case : <1>]
[agr : <2>]

NP[ ]
[case : <1>]
[agr : <2>]

N[case : <1> nom/acc]
[agr : <2>]
[case : nom]
[agr : pers : 3]
[num: sing]

N[case : <1> nom/acc]
[agr : <2>]
[case : acc]
[agr : pers : 3]
[num: sing]

she

her
Case and Agreement Example: “saw her”

\[
\begin{align*}
S_r & [ ] \\
\text{assign-case} & : <3> [ ] \\
\text{agr} & : <4> [ ] \\

\text{NP}_l & [ ] \\
\text{case} & : <3> \\
\text{agr} & : <4> \\

\text{VP} & [ ] \\
\text{assign-case} & : <3> \\
\text{agr} & : <4> \\
\text{assign-case} & : <5> [ ] \\
\text{agr} & : <6> [ ] \\

\text{NP} & [ ] \\
\text{case} & : <1> \\
\text{agr} & : <2> \\

\text{V} & [ ] \\
\text{assign-case} & : <5> \\
\text{agr} & : <6> \\

\text{NP} & [ ] \\
\text{case} & : \text{nom/acc} \\
\text{agr} & : <2> [ ] \\
\text{case} & : \text{acc} \\
\text{agr} & : \text{pers: 3} \\
\text{num: sing} & : \text{ sing} \\
\end{align*}
\]
Case and Agreement Example: “she saw her”

```
S, [ ]
  assign-case : <3>[ ]
  agr : <4>[ ]

NP[case : <5>]
  agr : <6>[ ]
  case : <5> nom/acc

VP[assign-case : <7>]
  agr : <8>[ ]

  assign-case : <7>[ ]

V[assign-case : <7>]
  agr : <8>[ ]
  case : <1>

  assign-case : <1>

NP[case : acc]
  agr : <2>[ ]
  case : <1>

  assign-case : <1> nom/acc

N[case : <1> nom/acc]
  agr : <2>[ ]
  case : acc
```

```
she

saw

her
```
Case and Agreement Example: “she saw her” Unified

\[
S_r \left[ \begin{array}{l}
\text{assign-case} : <1> \text{ nom} \\
\text{agr} : <2> \left[ \begin{array}{l}
\text{pers} : 3 \\
\text{num} : \text{ sing}\end{array} \right]
\end{array} \right]
\]

\[
\begin{array}{c}
NP \left[ \begin{array}{l}
\text{case} : <1> \\
\text{agr} : <2>
\end{array} \right] \\
VP \left[ \begin{array}{l}
\text{assign-case} : <1> \\
\text{agr} : <2>
\end{array} \right]
\end{array}
\]

\[
\begin{array}{c}
N \left[ \begin{array}{l}
\text{case} : <1> \\
\text{agr} : <2>
\end{array} \right] \\
V \left[ \begin{array}{l}
\text{assign-case} : <1> \\
\text{agr} : <2>
\end{array} \right] \\
NP \left[ \begin{array}{l}
\text{agr} : <3> \left[ \begin{array}{l}
\text{pers} : 3 \\
\text{num} : \text{ sing}\end{array} \right] \\
\text{case} : <4> \text{ acc}
\end{array} \right]
\end{array}
\]

\[
\begin{array}{c}
she \\
saw \\
her
\end{array}
\]
Obligatory Adjunction Example: “tries to leave”
Obligatory Adjunction Example: “tries to leave” Unified

```
S_r [tense : +]
   /   \
  /     \
NP     VP
   /   \
  /     \
N      V
  /   \
John  tries
   /   \
  /     \
NP     VP_d
   /   \
  /     \
ε      to
   /   \
  /     \
V      VP
     /   \
    /     \
 leave
```

Grammar Formalisms
Obligatory Adjunction Example: “thinks to leave”

\[ \beta \text{thinks} \]

\[ S_f \hspace{1cm} \alpha \text{leave} \]

\[ \text{John} \hspace{1cm} \text{thinks} \]

\[ S_r \hspace{1cm} \text{to} \hspace{1cm} \text{VP} \]

\[ \text{John} \hspace{1cm} \text{thinks} \]

\[ \text{to} \hspace{1cm} \text{VP} \]

\[ \text{leave} \]
Obligatory Adjunction Example: Past Participle

NP [case: nom
  agr: [num: sing
    pers: 3]]

he

VP [tense: <1>
  mode: <2>
  agr: <3>
  assign-case: <4>]

he

VP [tense: <1>
  mode: <2>
  agr: <3>
  assign-case: <4>]

V [tense: <1>
  mode: <2>
  agr: <3>
  assign-case: <4>]

V [tense: <1>
  mode: <2>
  num: sing
  pers: 3
  assign-case: nom]

has
Obligatory Adjunction Example: Past Participle
Obligatory Adjunction Example: Past Participle
Obligatory Adjunction Example: Past Participle
Obligatory Adjunction Example: Past Participle

\[
S, \quad [ ]
\]
\[
tense : <5> \quad [ ]
mode : <6> \quad [ ]
\]
\[
NP, \quad \text{case} : <7> \quad \text{nom}
ag
case : <8> \quad \text{num} : \text{sing}
pers : 3
\]
\[
VP, \quad \text{agr} : <8>
assign-case : <7>
tense : <5>
mode : <6>
tense : <1>
mode : <2>
agr : <3>
assign-case : <4>
\]
\[
he,
\]
\[
V, \quad \text{tense} : <1> \quad [ ]
mode : <2> \quad [ ]
agr : <3> \quad [ ]
assign-case : <4> \quad [ ]
\]
\[
NP, \quad \text{assign-case} : <9>
tense : <11>
mode : <10>
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
\[
\]
Obligatory Adjunction Example: Past Participle
Obligatory Adjunction Example: All Features