Tree-Adjoining Grammar (TAG)

Linguistics 614

With thanks to Detmar Meurers & Laura Kallmeyer

Spring 2015
Overview

1. Introduction
   - Introduction to TAG
   - Feature-Based TAG

2. TAG for Natural Languages
   - Properties of Elementary Trees
   - Constituency and Dependency

3. Conclusion
TAG (Joshi et al. 1975, Joshi & Schabes 1997) extends CFG in the following sense:

- In a CFG, each derivation step amounts to substituting a new tree of height 1 for a leaf.
- In a TAG, we allow (finite) trees that are arbitrarily large.
Introduction to TAG (2)
Pharse Structure Trees

1. $S \rightarrow NP \ VP$
2. $VP \rightarrow ADV \ VP$
3. $VP \rightarrow V$
4. $NP \rightarrow John$
5. $ADV \rightarrow sometimes$
6. $V \rightarrow laughs$
Introduction to TAG (3)

String rewriting derivation

0 S
1 NP VP (rule #1)
2 John VP (rule #4)
3 John ADV VP (rule #2)
4 John sometimes VP (rule #5)
5 John sometimes V (rule #3)
6 John sometimes laughs (rule #6)
Introduction to TAG (4)
Substitution (Tree Substitution Grammars (TSGs))

Elementary structures are trees

- Arrow indicates where substitution takes place

Tree 1:
- S
- NP
- VP
  - V
    - laughs

Tree 2:
- NP
- VP
- V
  - John

Derived tree:
- S
- NP
- VP
  - V
    - John
    - laughs
Q: with TSGs, how would we obtain *John heartily laughs*?

- Besides substitution at leaves, we also can replace internal nodes with new trees (adjunction).
  - In an adjunction, the new tree is an auxiliary tree and has a special leaf, the foot node.
  - The trees that are added in substitution operations are called initial trees.

Auxiliary tree modifies an XP only if root & foot nodes are both XP

- Using adjunction gives **Tree Adjoining Grammar (TAG)**
(1) John sometimes laughs

Derived tree:
A **Tree Adjoining Grammar (TAG)** is a quadruple $G = \langle N, T, I, A \rangle$ such that

- $T$ and $N$ are disjoint alphabets, the terminals and nonterminals,
- $I$ is a finite set of initial trees, and
- $A$ is a finite set of auxiliary trees.
- The trees in $I \cup A$ are called **elementary** trees.
  - nodes in elementary trees are labeled with symbols from $T \cup N \cup \{\varepsilon\}$
  - all internal nodes have labels from $N$
- $G$ is **lexicalized** iff each elementary tree has at least one leaf with a terminal label.
TAG as defined above are more powerful than CFG but they cannot generate the copy language ($\{ww \mid w \in \{a, b\}^*\}$).

In order to increase the expressive power, adjunction constraints are introduced that specify for each node:

1. whether adjunction is mandatory and
2. which trees can be adjoined.
Three types of constraints are distinguished:

1. A node is said to carry a **obligatory adjunction (OA)** constraint if adjunction is obligatory at that node.

2. A node is said to carry a **null adjunction (NA)** constraint if adjunction is not obligatory and the set of adjoinable trees is empty.

3. A node is said to carry a **selective adjunction (SA)** constraint if adjunction is not obligatory and the set of adjoinable trees is not empty.
Introduction to TAG

Example: TAG for the copy language
(2) John seems to sleep

S

NP

NP
John

VP

OA

V

to

V

sleep

VP

VP*

V

seems
TAG derivations are described by derivation trees:

For each derivation in a TAG there is a corresponding derivation tree. This tree contains

- nodes for all elementary trees used in the derivation,
- and edges for all adjunctions and substitutions performed throughout the derivation.

Whenever an elementary tree $\gamma$ was attached to the node at address $p$ in the elementary tree $\gamma'$, there is an edge from $\gamma'$ to $\gamma$ labeled with $p$.

We use Gorn addresses: The root has address $\varepsilon$, and the $i$th daughter of the node with address $p$ has address $p_i$. 
The derivation tree for the derivation of (2) *John seems to sleep*:

```
sleep
   / \  
  1   2
john  seems
```
Feature-structure based TAG (FTAG) (Vijay-Shanker & Joshi, 1988): each node has a top and a bottom feature structure (except substitution nodes that have only a top). Nodes in the same elementary tree can share features (extended domain of locality).

Intuition:

- The top feature structure tells us something about what the node represents within the surrounding structure, and
- the bottom feature structure tells us something about what the tree below the node represents.

In the final derived tree, both must be the same.
Feature-Based TAG

Example:

```
[CAT  S]
[CAT  S]
```

```
[CAT  NP]
[A GR  1]
```

```
[CAT  VP]
[AGR  1]
[PERS  3]
[NUM  sing]
```

```
[CAT  V]
[CAT  V]
```

sings
Example:

```
[CAT   S]
[CAT   S]
[CAT   NP]
[AGR   1]
[CAT   VP]
[AGR   1]
[MODE  ind]
[CAT   VP]
[MODE  ger]
[CAT   V]
[CAT   V]
```

singing
FTAG (4)

Unification during derivation:

- **Substitution**: the top of the root of the new initial tree unifies with the top of the substitution node.
- **Adjunction**: the top of the root of the new auxiliary tree unifies with the top of the adjunction site, and the bottom of the foot of the new tree unifies with the bottom of the adjunction site.
- In the final derived tree, top and bottom unify for all nodes.
Example:

```
[ CAT  S ]
[ CAT  S ]

[ CAT  NP ]
[ AGR   1 ]

[ CAT  NP ]
[ AGR ]
[ PERS  3 ]
[ NUM   sing ]

 john

[ CAT  VP ]
[ CAT  V ]
[ CAT  V ]

| sings |
```
FTAG (6)

Example:

```
[CAT  VP]
[CAT  VP]
[AGR  2]
[MODE  ind]
```

```
[CAT  V]
[CAT  V]
[AGR  2]
[PERS  3]
[NUM  sing]
```

```
[AGR  1]
```

```
[CAT  S]
[CAT  S]
```

```
[CAT  VP]
```

```
[CAT  VP]
```

```
[CAT  VP]
```

```
[CAT  V]
```

```
[CAT  V]
```

```
is
```

```
singing
```
FTAG (7)

- In FTAG, there are no explicit adjunction constraints.
- Instead, adjunction constraints are expressed via feature unification requirements.
- Important: LTAG feature structures are restricted; there is only a finite set of possible feature structures.
- Therefore, the following can be shown:

  For each FTAG there exists a weakly equivalent TAG with adjunction constraints and vice versa. The two TAGs generate even the same sets of trees, only with different node labels.
Properties of Elementary Trees

Elementary trees (1)

Important features of LTAG (Lexicalized TAG):

- Grammar is **lexicalized**
- Recursive parts are put into separate elementary trees that can be adjoined (**Factoring of recursion, FR**)
- Elementary trees can be arbitrarily large, in particular (because of FR) they can contain elements that are far apart in the final derived tree (**Extended domain of locality**)

**LTAG game:** [http://www.ltaggame.com](http://www.ltaggame.com)
(3) a. \( \text{who}_i \) did John tell Sam that Bill likes \( t_i \)

b. \( \text{who}_i \) did John tell Sam that Mary said that Bill likes \( t_i \)
Properties of Elementary Trees

Elementary trees (3)

- Elementary trees are extended projections of lexical items.
  - Recursion is factored away $\Rightarrow$ finite set of elementary trees.
  - The elementary tree of a lexical predicate contains slots for all arguments of the predicate, for nothing more.

- Besides lexical predicates, there are functional elements (complementizers, determiners, auxiliaries, negation) whose treatment in LTAG is less clear. They can be
  - either in separate elementary trees (XTAG, 2001)
  - or in the elementary tree of the lexical item they are associated with (Frank, 2002).
Elementary trees (4)

Example

(4) John gives a book to Mary

```
  S
   NP↓
     VP
      NP↓
        V
          NP↓
            gives
        P
          NP↓
            to
```
Elementary trees (5)

Example:

(5) John expected Mary to make a comment

*expected* selects for a subject NP and an infinitival sentence:

![Tree diagram]

The sentential object is realised as a foot node in order to allow extractions:

(6) whom does John expect to come?
to make a comment: make and comment in the same elementary tree since they form a light verb construction:

```
S
  NP↓
    VP
      V
      NP
        to make
        N
        comment

      NP
        Det
        a
        NP*
```
Elementary trees (7)

Example with modifiers:

(7) the good student participated in every course during the semester

```
         N
        / \  
       AP   N*
          / \
         A   
           /
          good

        NP
       /  
      Det   N
         /  
        the  student
```
Properties of Elementary Trees

Elementary trees (8)

\[
\begin{align*}
S & \rightarrow NP \rightarrow VP \\
S & \rightarrow VP \\
VP & \rightarrow V PP \\
NP & \rightarrow VP* PP \\
VP* & \rightarrow PP \\
PP & \rightarrow P NP \\
NP & \rightarrow in \\
VP & \rightarrow during
\end{align*}
\]
Constraints on larger structures (constraints on “unbounded dependencies”) need not be stipulated but follow from the possibilities of adjunction in the elementary trees.

Fundamental LTAG hypothesis: Every syntactic dependency is expressed locally within a single elementary tree.

Non-local dependency corollary: Non-local dependencies always reduce to local ones once recursive structure is factored away.

What do the elementary trees look like for the following sentence?

(8) which book did Harvey say Cecile had read
Properties of Elementary Trees

Elementary trees (10)

Tree families

- In the lexicon, the trees are organized in tree families.
- Each family contains a base tree and trees derived from the base tree using transformations.
- Important: These transformations operate only on a finite set, i.e., on structures of bounded size.

Tree families group together trees belonging to the same subcategorization frame.
Properties of Elementary Trees

Elementary trees (11)
Tree family example

The trees for the different forms of *buy* in (9) belong to one tree family.

(9) a. John bought a book
   b. What does John buy?
   c. Who bought a book?
   d. A book was bought by John
   e. The man who bought the book this morning was from Tübingen.

*buy* in (10) has a different tree family.

(10) John bought Mary a book
The derived tree gives the constituent structure.
The derivation tree records the history of how the elementary trees are put together.
⇒ the edges in the derivation tree represent predicate-argument dependencies where a substitution-edge has a downward direction, an adjunction edge an upward direction;
the derivation tree is close to a semantic dependency graph.
⇒ compute semantics on derivation tree
Constituency and Dependency (2)
Ditransitive verb

(11) John buys Bill a book

Elementary trees:

Derivation tree:

1 22 23
John Bill a_book
Constituency and Dependency (3)

Sentential Complement

(12) Bill hopes that John wins

\[
\begin{align*}
\text{NP} & \quad \text{Bill} \\
\text{NP} & \quad \text{hopes} \\
\text{S} & \quad \text{that} \\
\text{NP} & \quad \text{wins} \\
\text{VP} & \quad \text{hopes} \\
\text{S} & \quad \text{Bill} \\
\end{align*}
\]
Constituency and Dependency (4)

Raising to Object

(13) John expects [ Bill to win ]

\[
S \\
NP \downarrow \quad VP \\
\downarrow \quad V \\
\downarrow \quad S^* \\
\text{expects} \\
\text{to win}
\]

\[
S \\
NP \downarrow \quad VP \\
\downarrow \quad S^* \\
\text{to win}
\]
Constituency and Dependency (5)

Object-control Equivalence

(14) John persuades Bill [ PRO to leave ]

```
S
  NP
    V
      persuades
    NP
      S*
        PRO
        to leave
```

```
to leave
  \epsilon
  persuades
  1\ 22
  John  Bill
```
Constituency and Dependency (6)

Subject raising

(15) John seems to like Bill

\[
\begin{array}{c}
S \\
\downarrow \quad \downarrow \\
NP \\
\downarrow \quad VP \\
VP \\
\downarrow \\
V \\
seems \\
\end{array}
\quad
\begin{array}{c}
S \\
\downarrow \quad \downarrow \\
NP \\
\downarrow \quad VP \\
VP \\
\downarrow \\
V \\
to \\
like \\
\end{array}
\]

\[
\begin{array}{c}
to \_like \\
\end{array}
\quad
\begin{array}{c}
1 \\
John \\
\end{array}
\quad
\begin{array}{c}
2 \\
seems \\
\end{array}
\quad
\begin{array}{c}
22 \\
Bill \\
\end{array}
\]
(16) which book did Harvey say Cecile had read

had_read

1
which_book

2
did_say

21
Cecile

21
Harvey
The derivation tree is not always the semantic dependency structure:

(17) roasted red pepper

⇒ proposal of alternative derivation with multiple adjunctions of modifier trees at the same node.
On the other hand, multiple adjunctions are not always desired:

(18) John seems to be likely to win the race

This is the correct dependency structure.
Another problematic case:

(19) John claims Bill is likely to win

```
1      ε       2
Bill  claims  is_likely
       1
       John

to_win
```
Conclusion

- TAG extend CFGs by introducing *adjunction*, in addition to *substitution*. TAG are only slightly more powerful than CFG.

- Elementary trees of lexical predicates encapsulate subcategorization frames: For each subcategorized argument, there is a non-terminal leaf (either a substitution node or a foot node).

- Recursion is factored away: only slots for subcategorized arguments are provided. Modifiers are added by adjunction.

- Extended domain of locality: Syntactic dependencies are defined locally, within single elementary trees. Unbounded dependencies arise from adjunction between an argument and its lexical head.


