Syntactic Theory

Tree-Adjoining Grammar (TAG)

Yi Zhang

Department of Computational Linguistics
Saarland University

November 10th, 2009

Outline

Introducing Auxiliary Trees

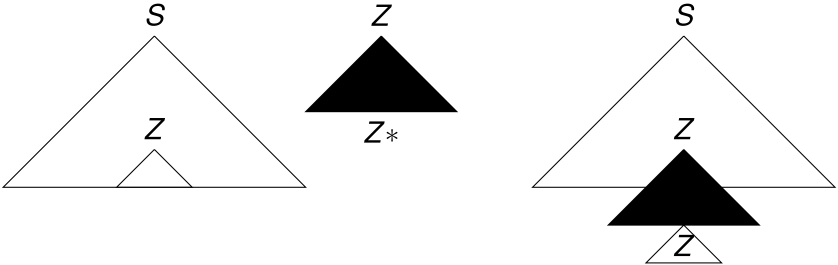
Auxiliary trees are the other type of elementary structures in TAG

- 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 β and a tree α (initial, auxiliary or derived tree) by cutting α into two parts and inserting β 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



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

Tree-Adjoining Grammar: Formal Definition

- A **Tree-Adjoining Grammar (TAG)** is a quintuple (Σ, NT, I, A, S), where
  1. Σ is a finite set of terminal symbols
  2. NT is a finite set of non-terminal symbols: $\Sigma \cap NT = \Phi$
  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

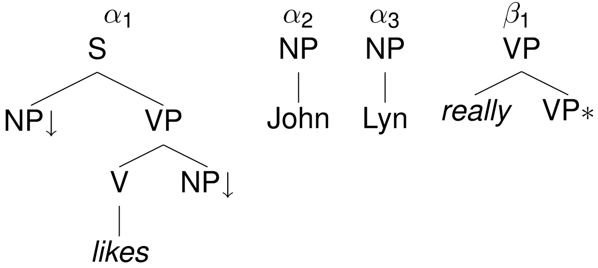
- **Derived Tree** is the result of the derivations and represents the phrase structure
- **Derivation Tree** specifies how a derived tree was constructed
  - The root is labeled by an *S*-type initial tree
  - All other nodes are labeled by initial trees in the cases of substitutions, and auxiliary 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 $\mathscr{G}$:

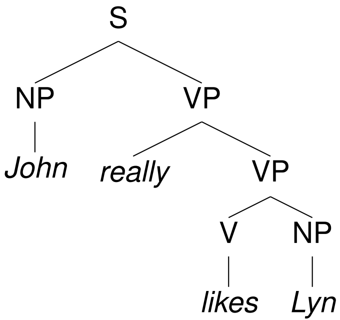
$\mathscr{G} = (\{john, lyn, really, likes\}, \{S, NP, VP, V\}, \{\alpha_1, \alpha_2, \alpha_3\}, \{\beta_1\}, \{S\})$

with the following elementary trees:

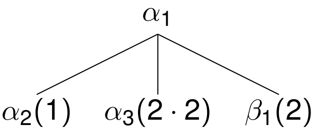


Derived Tree & Derivation Tree: Example (Cont.)

Derived Tree:



Derivation Tree:



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

Constraining Adjoining Operation

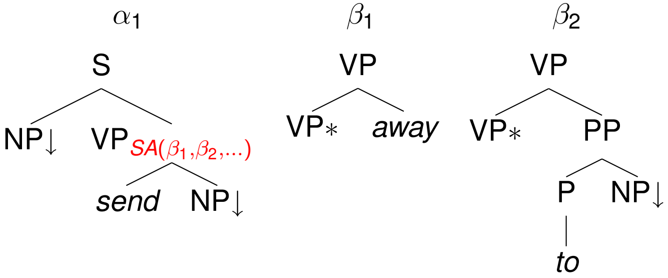
- In the TAG shown so far, an auxiliary tree β can be adjoined on any node *n*, if:
  - *n* has the identical label of the root in β
  - *n* is not annotated for substitution
- It is convenient for linguistic description to have more precision for specifying which auxiliary trees can be adjoined at a 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 $OA \doteq OA(A)$

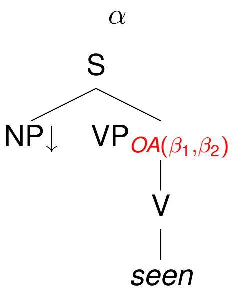
Selective Adjunction: An Example

One possible analysis of *“send”* could involve selective adjunction:



Obligatory Adjunction: An Example

For when you absolutely must have adjunction at a node:



Outline

Tree-Adjoining Grammar (TAG)

Adding Constraints to TAG

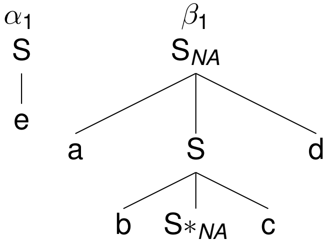
Formal Properties of TAG

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 $\{a^n b^n e c^n d^n, n \geq 1\}$ can not be generated by any CFG, but can be properly covered by TAG



Lexicalization of CFG with TAG

Theorem

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

- 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 TAG

Theorem

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

- Adjunction is sufficient to lexicalize context-free grammars
- The use of substitution enables one to lexicalize a grammar with more compact TAG

Closure of TAG under Lexicalization

Theorem

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

Other Formal Properties of TAG and TAL

- CFL $\subset$ TAL $\subset$ Indexed Languages $\subset$ 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

Joshi, A. and Schabes, Y. (1997).
Tree-adjoining grammars.