Tree-Adjoining Grammar (TAG)

One of several formalisms that are actually more powerful than CFG

Note: CFG with features isn't any more powerful than vanilla CFG. (Why? what do we mean by "more powerful"?)

Read the Transparencies

- This lecture used transparencies.
- But here are some very brief notes to remind you what we covered.
- The transparencies are now online, too.

What CFG and TAG Share

- Build a tree from a bunch of tree fragments
  - technically, build a derived tree from elementary trees
- The collection of allowed fragments **is** the grammar!
- The fragments might correspond to context-free rules, but they might be bigger as in TAG
- Semantics associated with every fragment

Substitution

- Substitution: Stick an appropriate *initial tree* fragment at the bottom of a tree (to expand a childless nonterminal node)
- Fills semantic slots of other words
- Get templates and idioms for free: "NP called NP up," "NP kicked the bucket"

Adjunction

- Adjunction: Stick an appropriate *auxiliary tree* fragment into the middle of a tree
- Splits a node into two parts and sticks some material between them
- Good for adding optional modifiers
- Good for long-distance dependencies
- Why insert into the middle?
  - Because the insertion doesn't affect the specified semantic relations among nodes in the original fragment

Features

- Still need features with TAGs
- Every elementary tree has some features
- If a node can be split, it must specify which features will get associated with the top half vs. bottom half
- Have to do unification or checking to figure out the values of the features after all the substitution and adjunction is done. This tells us how to do morphology legally.