Syntactic Theory

Lecture 4 (15.11.2010)

PD Dr.Valia Kordoni

Email: kordoni@coli.uni-sb.de

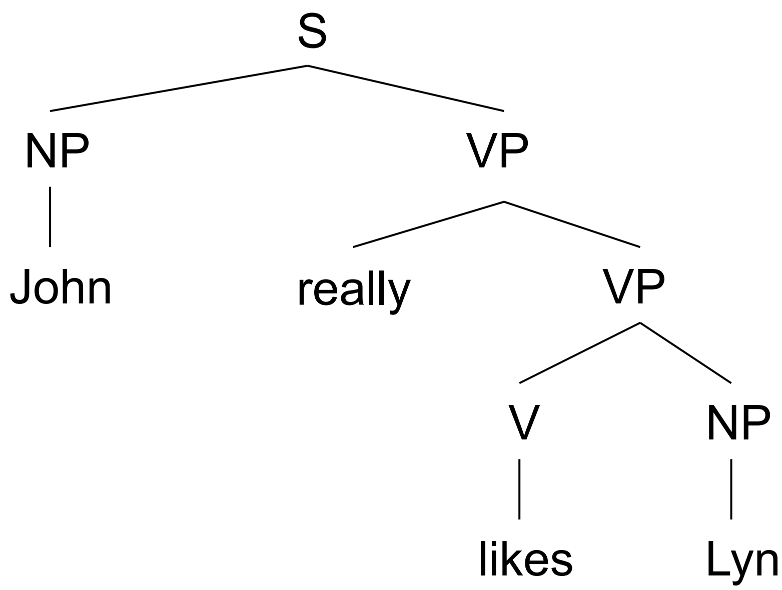
http://www.coli.uni-saarland.de/courses/syntactic-theory/2010/

Tree-Adjoining Grammar (TAG)

TAG

- Pseudo-extension of CFGs
  - Abandon the context-free grammar formalism
  - Keep the idea of deriving complete trees in a sequence of rewriting steps —but in TAG we rewrite *trees*, not strings
- Highly lexicalized (LTAG):
  - Every tree is associated with exactly one lexical item
  - Every lexical item is associate with a set of trees

Phrase Structure Trees



(1) a. S → NP VP

b. VP → really VP

c. VP → V NP

d. V → likes

e. NP → John

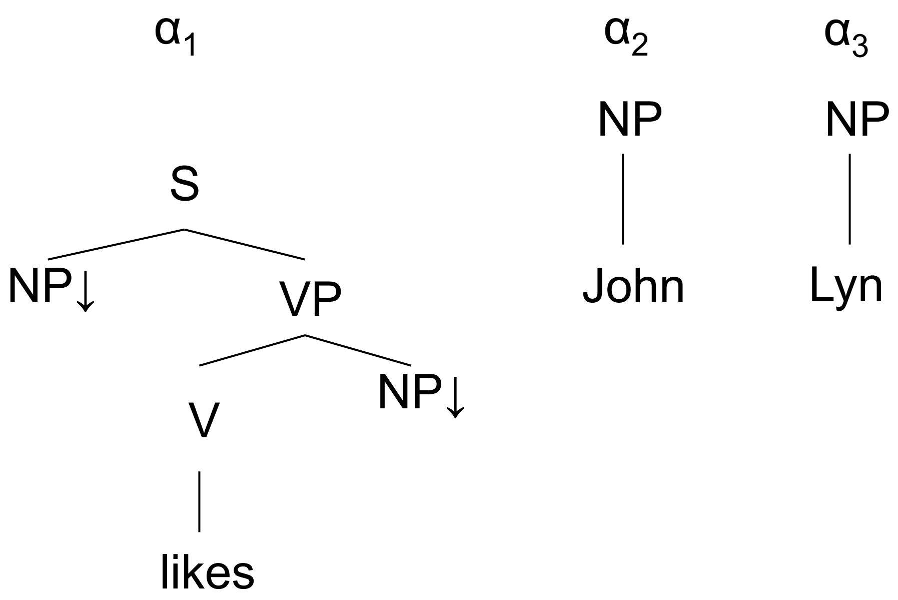
f. NP → Lyn

String rewriting derivation

1. S → **NP** VP (1a)
2. → John **VP** (1e)
3. → John really **VP** (1b)
4. → John really **V** NP (1c)
5. → John really likes **NP** (1d)
6. → John really likes Lyn (1f)

Tree Substitution Grammars

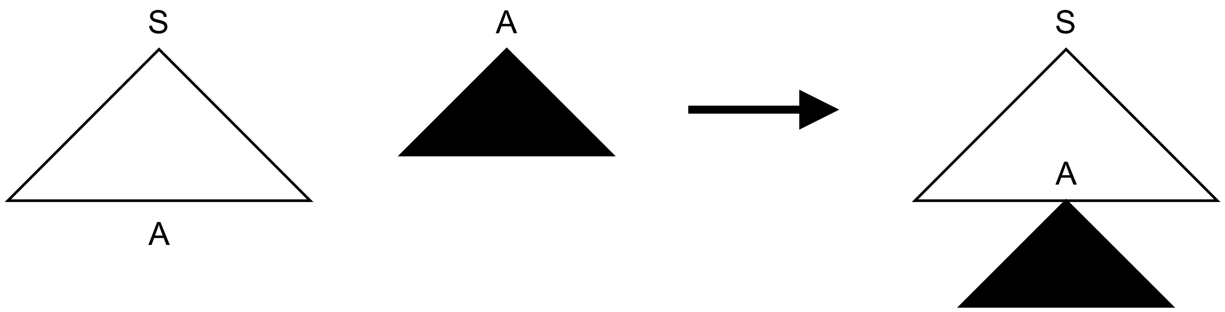
- Elementary structures are trees
- A down arrow (↓) indicates where a substitution takes place



Substitution operation

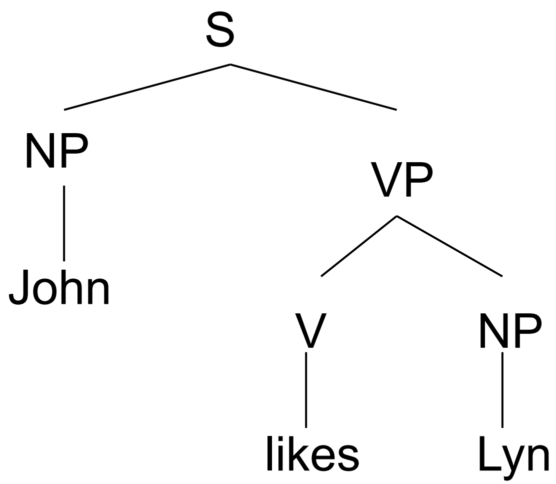
The substitution operation allows us to insert elementary trees into other elementary trees

- Where there is a (non-terminal) node marked for substitution (↓) on the **frontier**, an elementary tree **rooted** in the same category can be substituted there



Final Tree

So, we end up with the following **derived tree**



Notes:

- order of substitutions is irrelevant
- This tree is **completed** = there are no substitution nodes left on the frontier

Elementary trees

Let's step back a little and look at the building blocks of TAG. Our basic elements are **elementary trees**, which come in two guises:

- **initial trees**, which have:
  - root node
  - interior nodes labeled by non-terminal symbols
  - frontier nodes of terminal and non-terminal symbols; substitution nodes are marked by the down arrow (↓)

➔ TSGs only use initial trees

Elementary trees (cont.)

- **auxiliary trees**, which have
  - root node
  - interior nodes labeled by non-terminal symbols
  - frontier nodes similar to as in initial trees, but with a designated (*) **foot node** = identical label to the root node

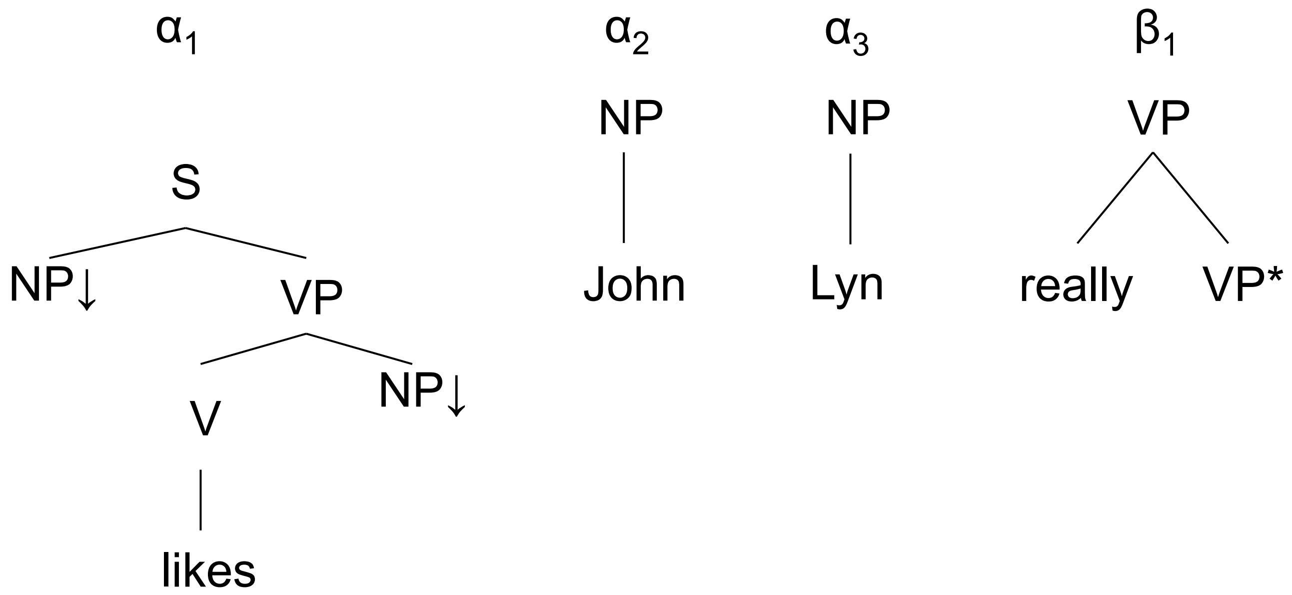
  ➔ TAGs need auxiliary trees for adjunction

➔ In LTAG, at least one frontier node must be a terminal symbol (lexical item)

Lexicalization

Lexicalization is the process of associating at least one terminal element with every elementary tree.

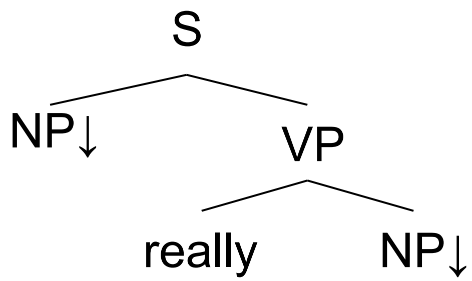
Adjunction is necessary if we want to lexicalize the grammars in a linguistically meaningful way, i.e., substitution isn't enough.



The need for adjunction

With the elementary trees above and using only substitution, there is no way to generate *John really likes Lyn*.

We would need an elementary tree along the following, unappealing lines:



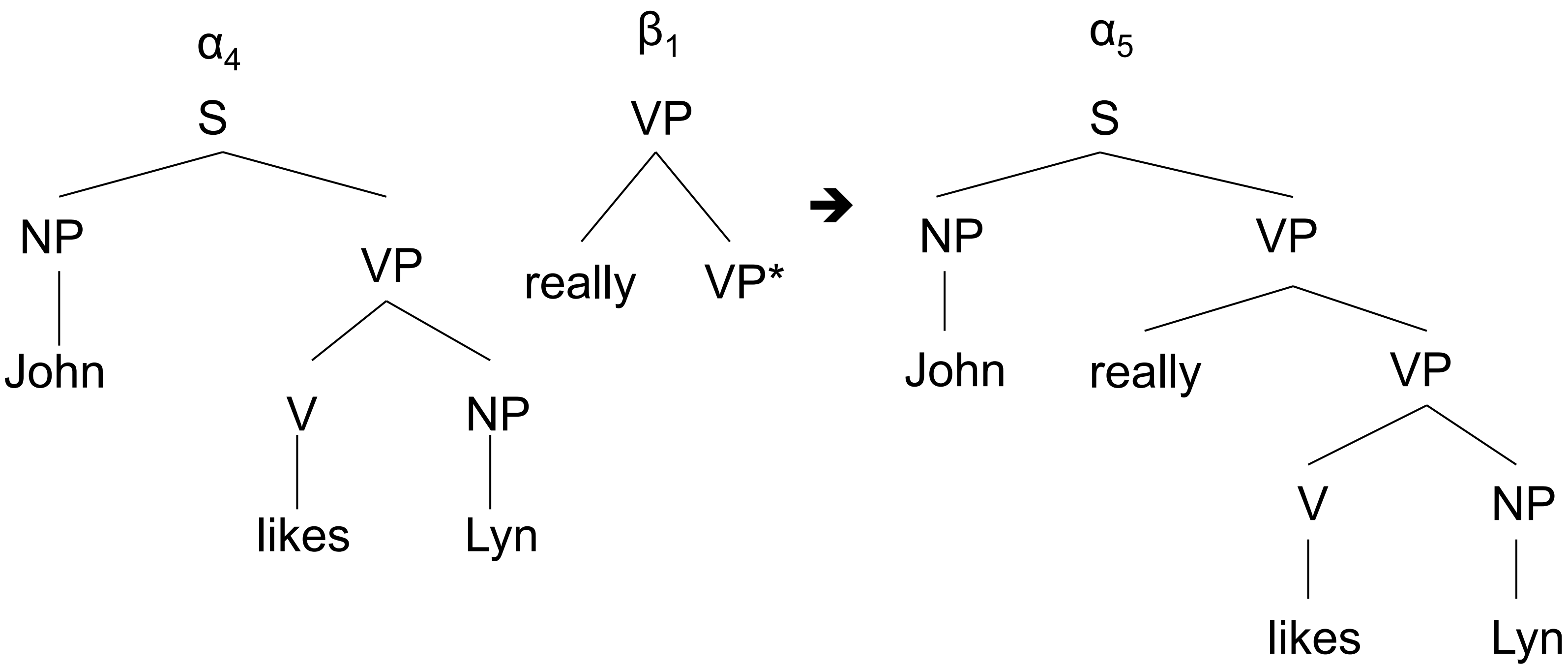
Adjunction

So, we introduce the **adjunction** operation, which is where auxiliary trees come in.

- We can now insert one tree into another, provided that the nodes match up
- That is, an auxiliary tree can modify an XP iff its root and foot nodes are both labeled XP

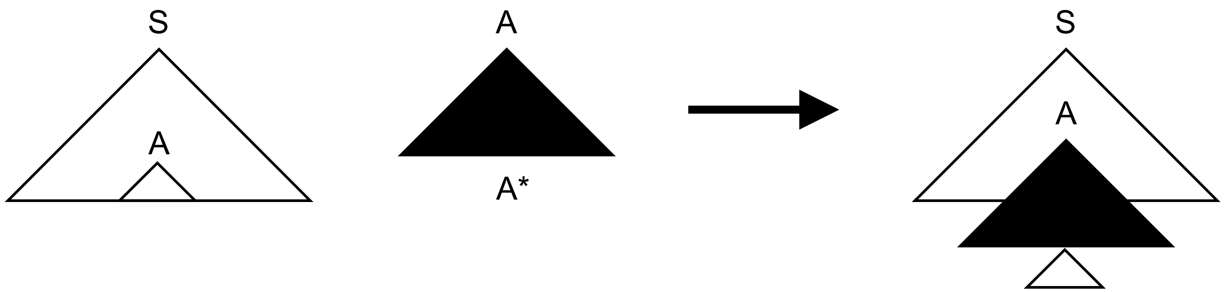
Using adjunction and substitution gives us true **Tree Adjoining Grammars (TAGs)**

Adjunction example



Adjunction operation

- An auxiliary tree is inserted into an initial tree (or derived tree) by cutting the initial/derived tree into two parts, above and below a node (A)
  - The node of the root of the auxiliary tree is identified with the node A
  - The node of the foot of the auxiliary tree is identified with the root of the excised tree



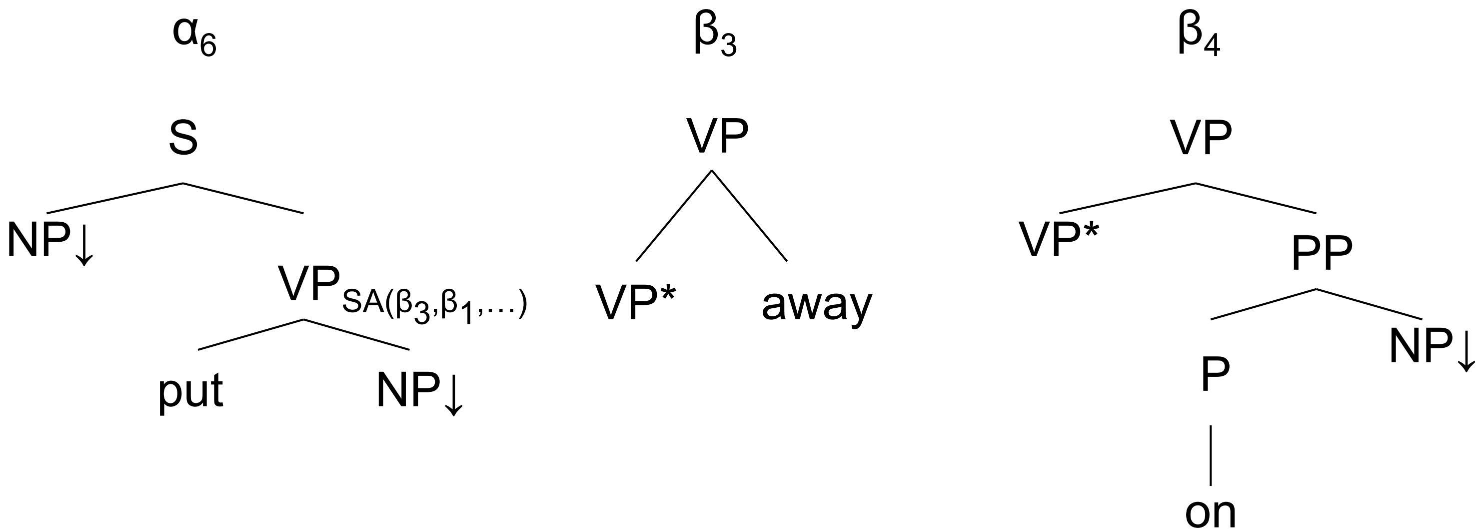
Adjunction (Adjoining) Constraints

Adjunction sometimes needs to be constrained even more than by ensuring category identity

- **Selective Adjunction** (SA(T)): only members of T, a set of auxiliary trees, may adjoin at this node
- **Null Adjunction** (NA): no adjunction is allowed at this node
- **Obligatory Adjunction** (OA(T)): a member of T must adjoin at this node

Selective Adjunction

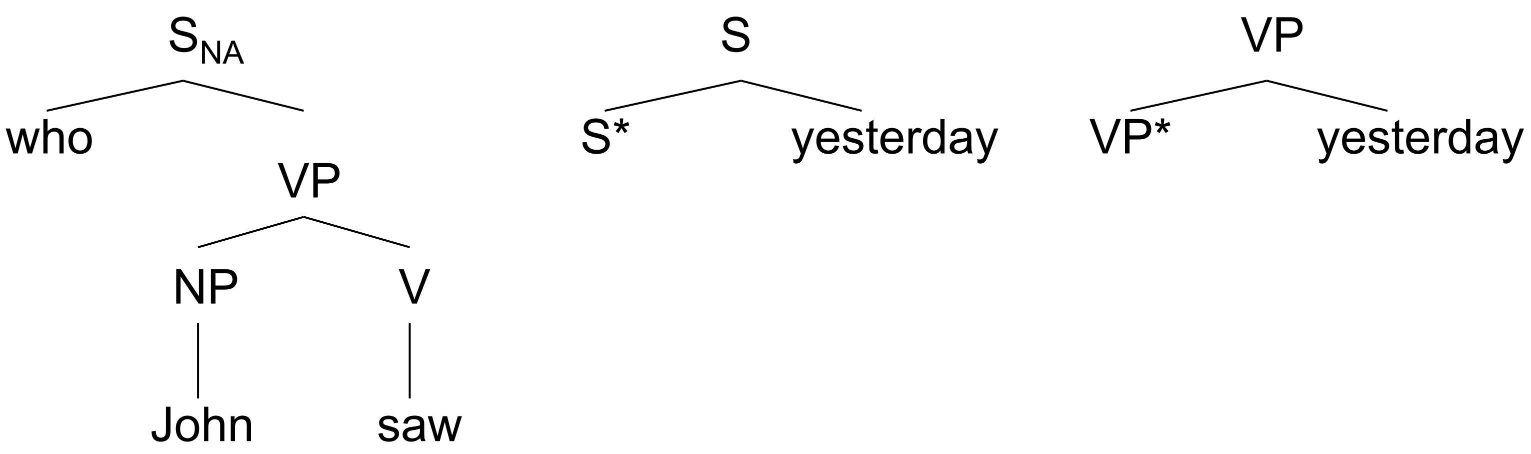
One possible analysis of put could involve selective adjunction:



➔ We might want a way to say that *locative* VP modifiers can adjoin here → we'll come back later to using features to redefine adjunction constraints

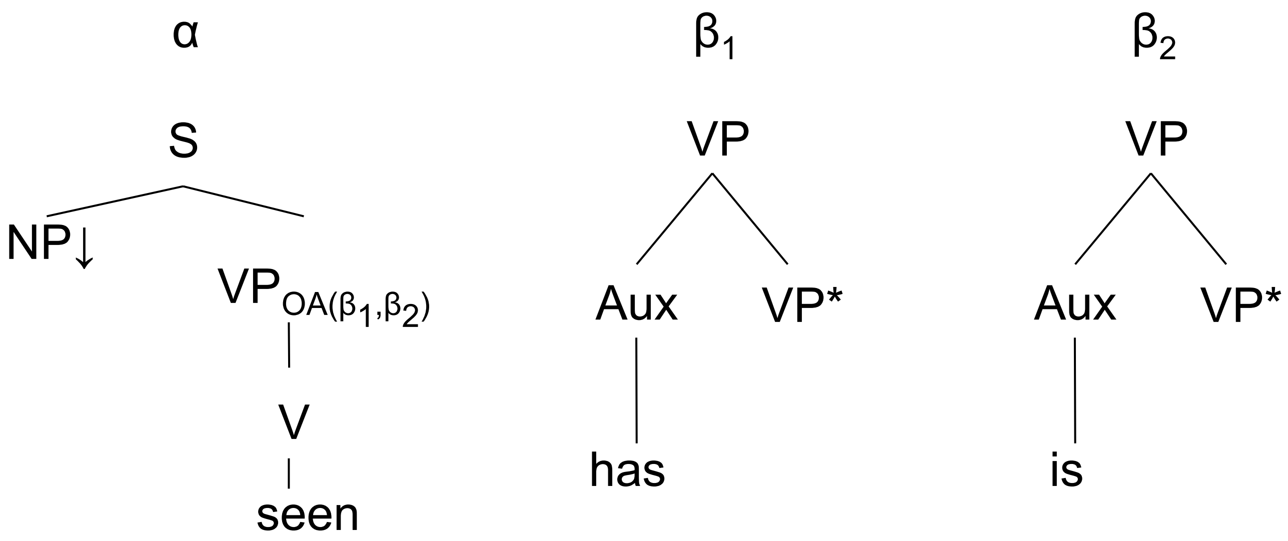
Null Adjunction

For when you absolutely cannot have an adjunct modifying a phrase



Obligatory Adjunction

For when you absolutely must have adjunction at a node:



This is often used to handle complement structures where the complement and the mother are the same category

Derived Trees and Derivation Trees

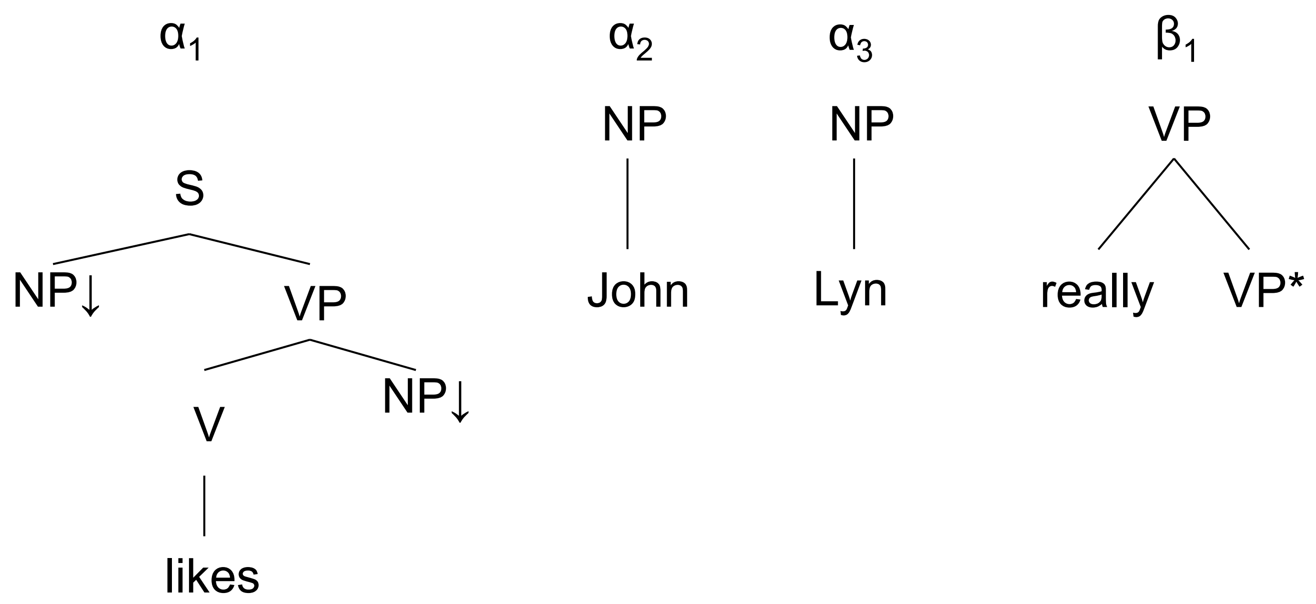
TAG distinguishes between **derived trees** and **derivation trees**. As a shorthand, think of them like so:

- Derived trees look like context-free/phrase structure trees
- Derivation trees look like dependency trees

That is, TAG provides us a way of having both kinds of representations

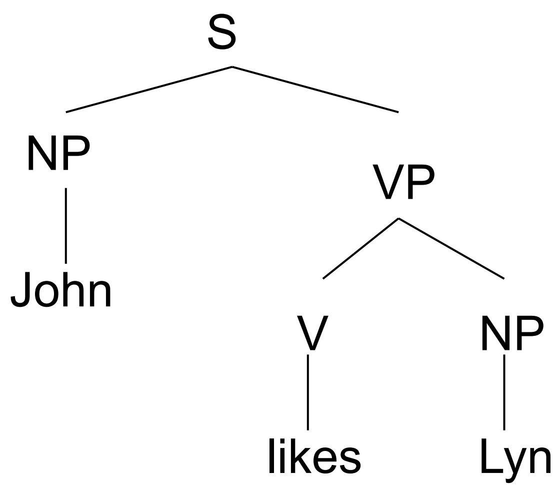
Example Lexicon

Recall the following lexical entries:



Derived Tree

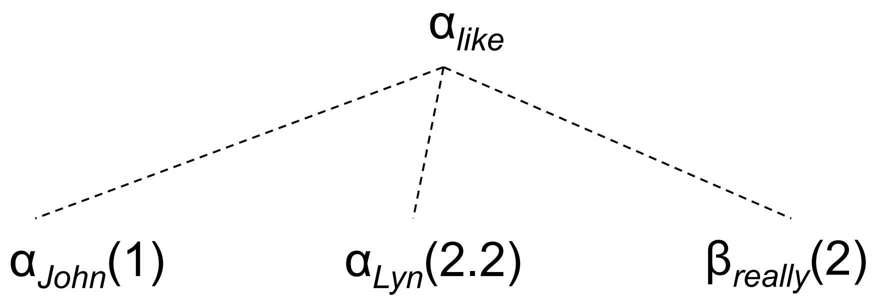
The derived tree is obtained by gluing all the tree pieces together until there's a normal-looking PS tree:



But this tells us nothing about how the tree was derived.

Derivation Trees

The derivation tree records a history of the derviation and in the process captures the dependency relations among words in the sentence



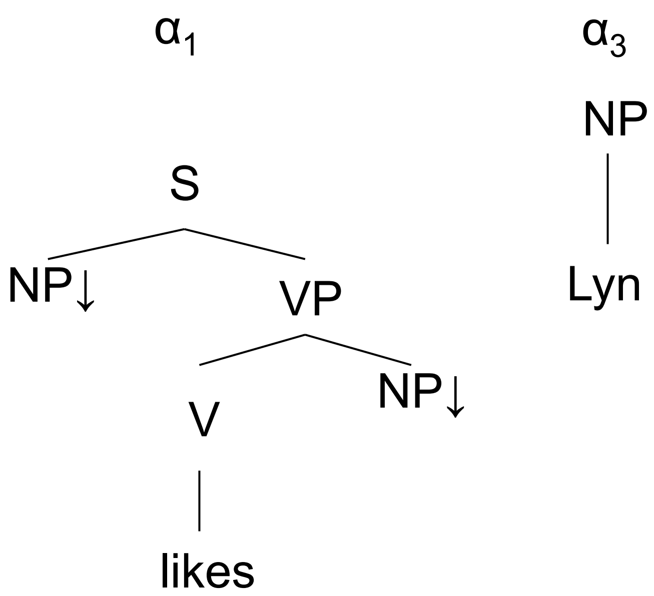
How to come up with a derivation tree

Each node in the derivation tree records the address of the node in the parent tree to which the adjunction/substitution was performed

- 0 is the root node address
- k is the address of the k^{th} child of the root node
- $p.q$ is the address of the q^{th} child of the node at address p (sort of like the q^{th} child of the p^{th} child)

Derivation tree address

Lyn gets the annotation 2.2 because VP is the second daughter of S, and NP is the second daughter of VP



Locality

TAG has a different notion of **locality** than in other formalisms

- On the one hand, an initial tree (e.g., lexical entry) can be of arbitrary size, so the domain of locality is increased.

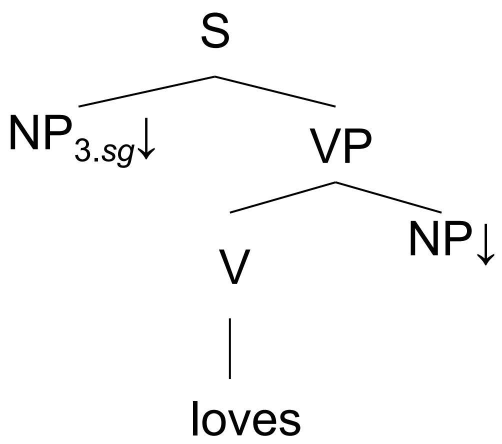
  ➔ Extended domain of locality (EDL)

- On the other hand, small initial trees can have multiple adjunctions inserted within them, so what are normally considered non-local phenomena are treated locally

  ➔ Factoring recursion from the domain of dependencies (FRD)

Domain of locality: agreement

The lexical entry for a verb like *loves* will contain a tree like the following:



With this extended domain of locality, we can easily state agreement between the subject and the verb in a lexical entry

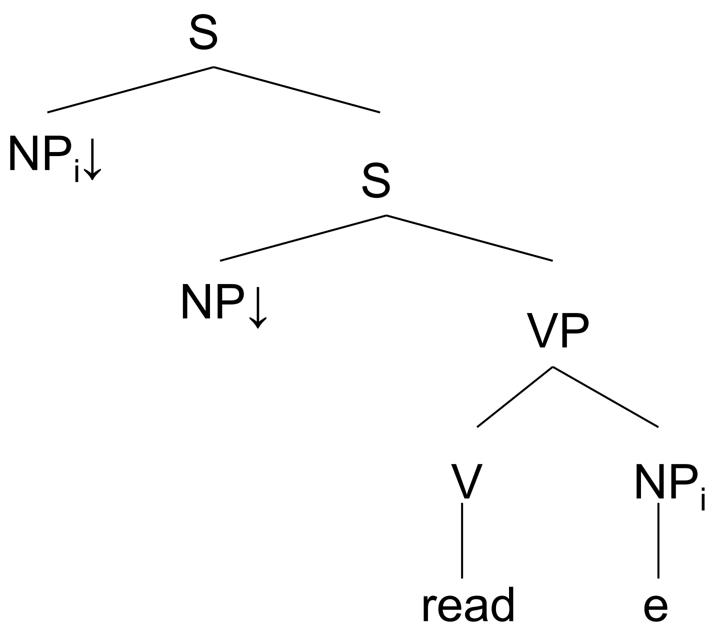
CFG notion of agreement

Compare the corresponding CFG rules; agreement has to be transfered between at least three different rules:

- $\text{S} \rightarrow \text{NP}_{3.sg}\ \text{VP}_{3.sg}$
- $\text{VP}_{3.sg} \rightarrow \text{V}_{3.sg}\ \text{NP}$
- $\text{V}_{3.sg} \rightarrow \text{loves}$

Factoring recursion from domain: Extraction

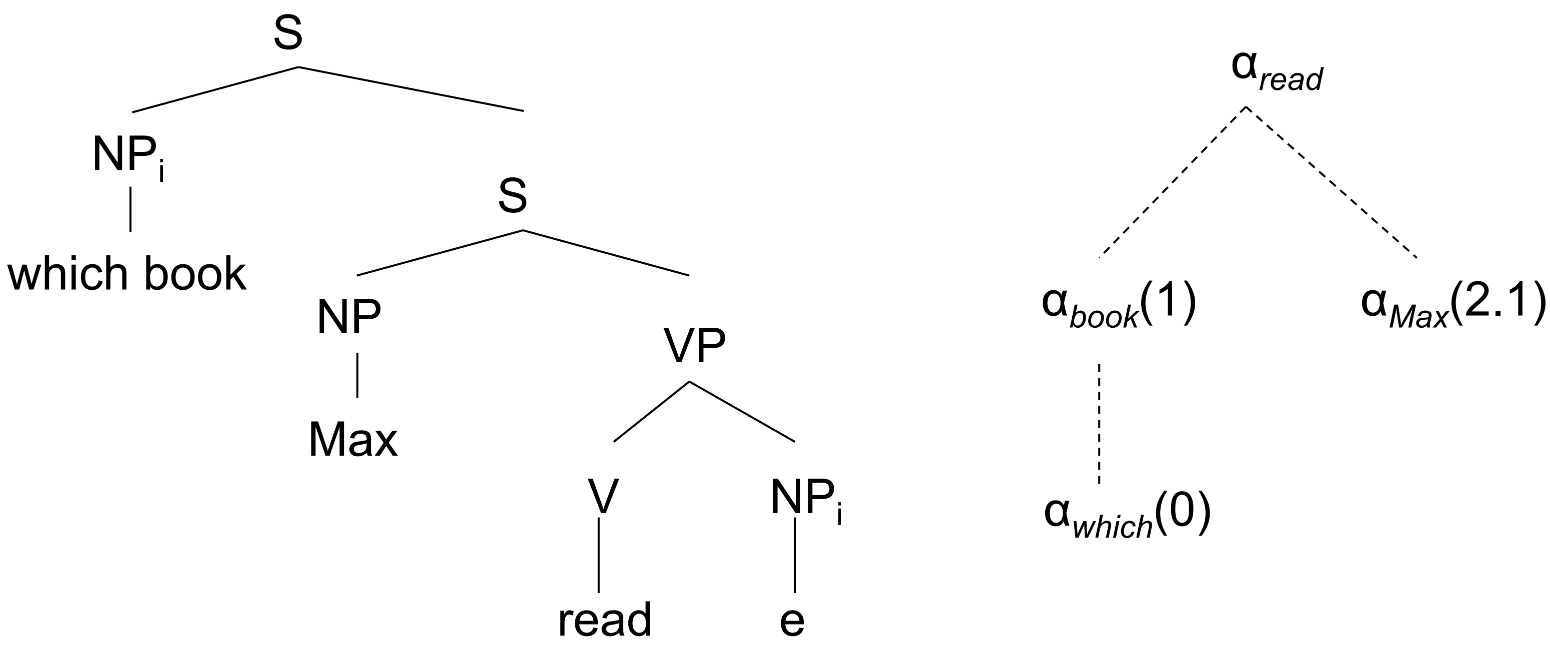
Another advantage of TAG's domain of locality is how extraction phenomena can be captured in a lexical entry



This will license a clause like *Which book Max read*

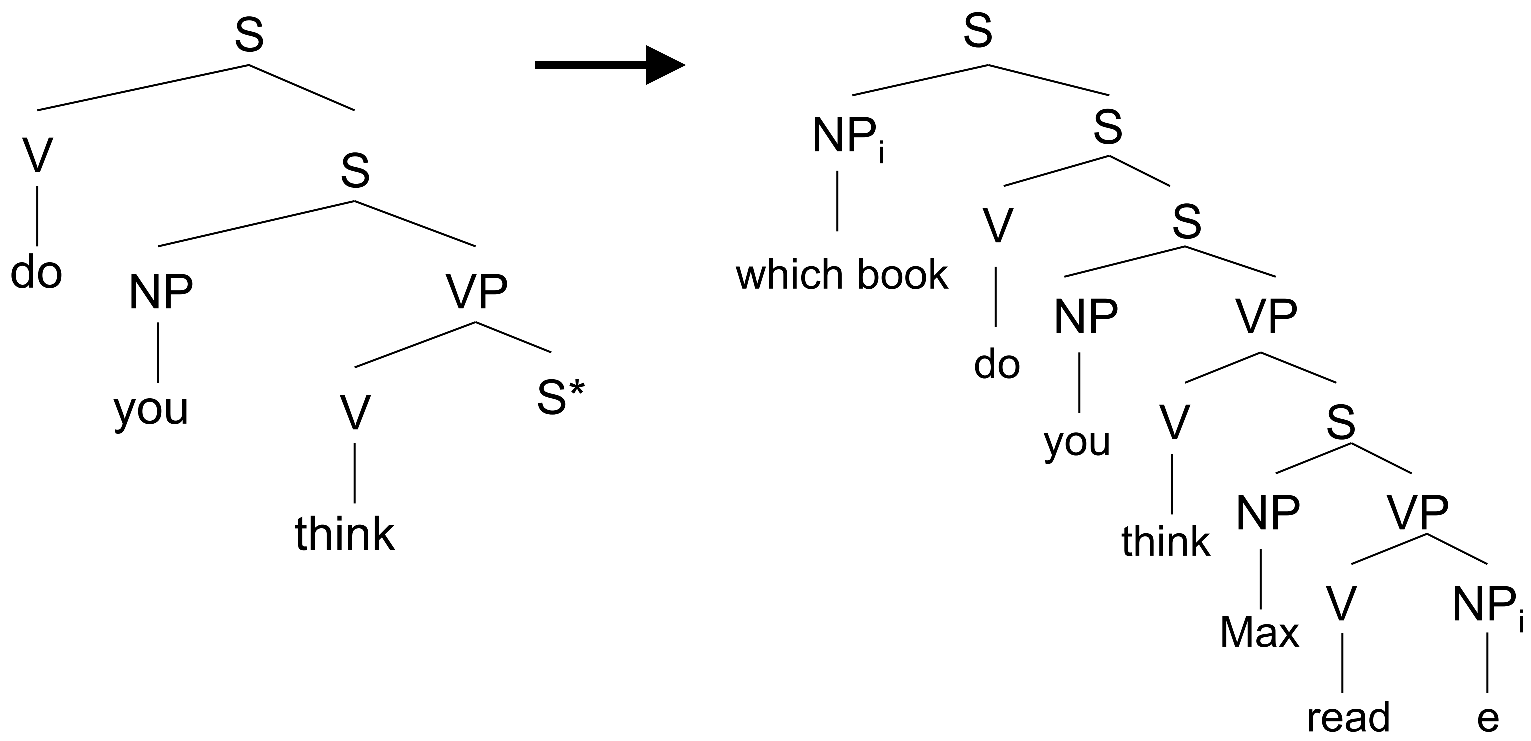
Example trees for extraction

The derived and derivation trees for *Which book Max read*:

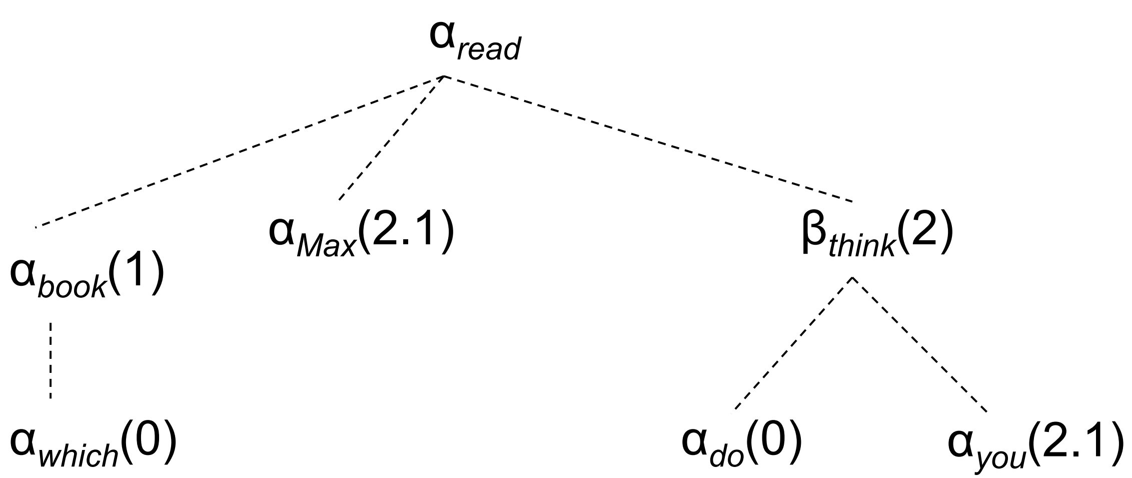


Extraction: strengths

One of the strengths of this method is that we can adjoin a phrase like *do you think*, and we still maintain the appropriate dependency relations:



The derivation tree



Note how the derivation/dependency tree maintains the same relations, simply adding another branch.

➔ That is, even though the derived tree is much higher, the dependency relations are the same.

UNIVERSITÄT DES SAARLANDES

Extraction: weaknesses

Some extraction phenomena are not as easy to handle in TAG, such as the following:

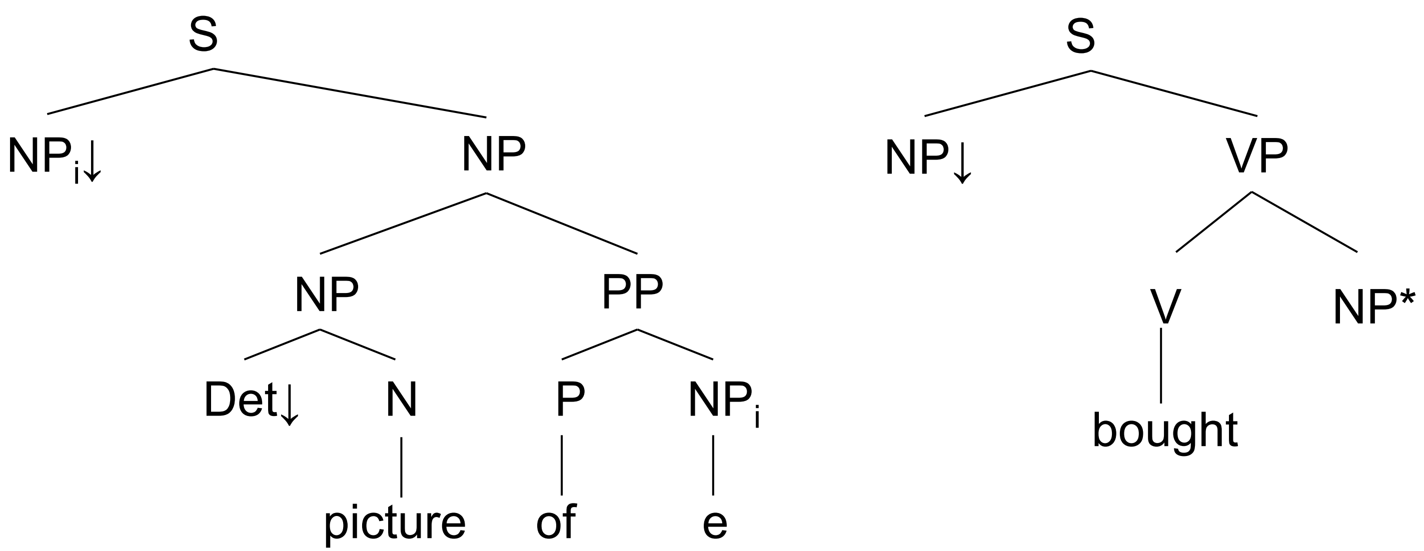
(2) This building, John bought a picture of.

What's wrong with this?

- The normal TAG view of extraction depends on adjunction, which is defined as involving a tree with identical root and foot nodes
- But *picture* is an NP, and we need to add a sentence in-between

Extraction example: picture

Lexical entry for *picture* (note again how more than one word can be in an initial tree) and potential entry for *bought*:



Problems with *picture* phrases

- Adjunction of this entry for *bought* into the *picture* tree is needed to get *This building*, *John bought a picture of*, but it is impossible
- TAG has to be extended to multi-component TAG (MCTAG), which we won't cover.

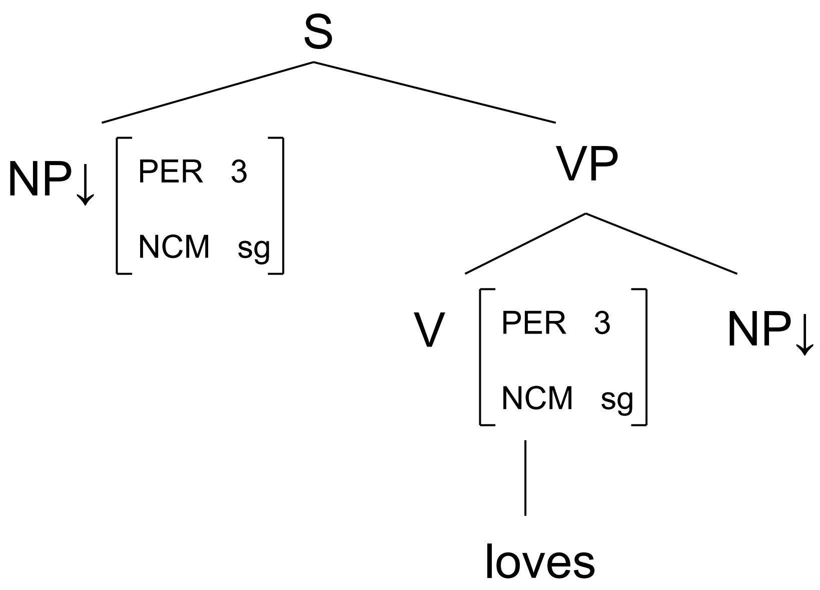
Using features in TAG

We have alluded to using **features** before, but we have not properly introduced them

- Features can be added to nodes in a tree
- In order for a tree to be substituted or adjoined, it must match the features of the node it is attaching to.
- In this way, we can reconstruct the ideas of obligatory, null, and selective adjunction

Feature example

A simple way of using features is simply as we've seen before, to enforce agreement and the like:



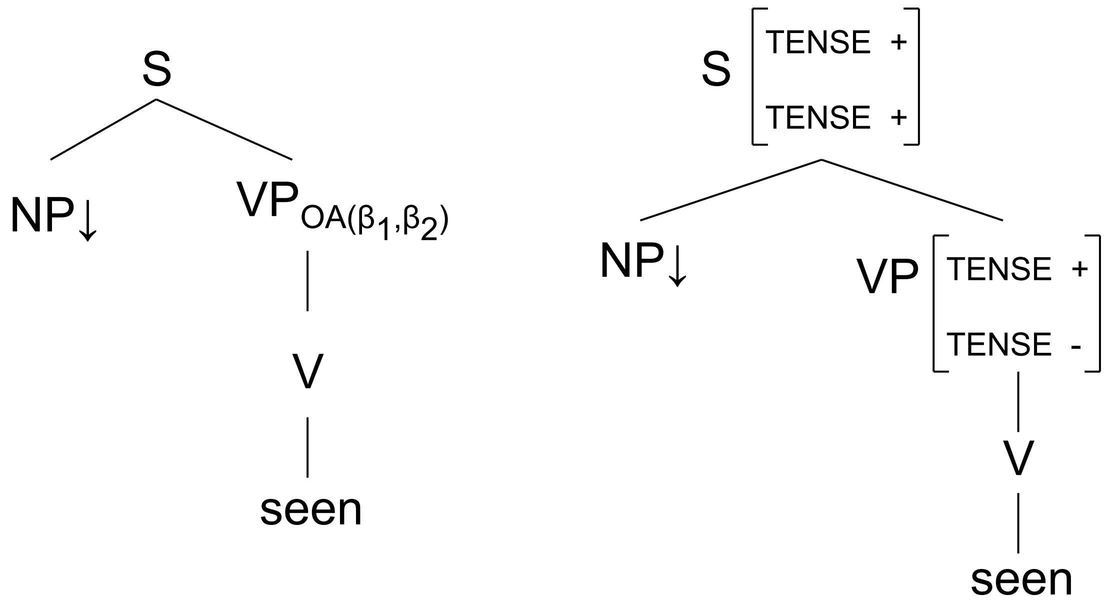
Top and bottom feature structures

To reconstruct the three kinds of adjunction, we need to define **top** and **bottom** feature structures

- top = tree above this node has these features, i.e., behaves like this
- bottom = tree below this node has these features

Feature structure example

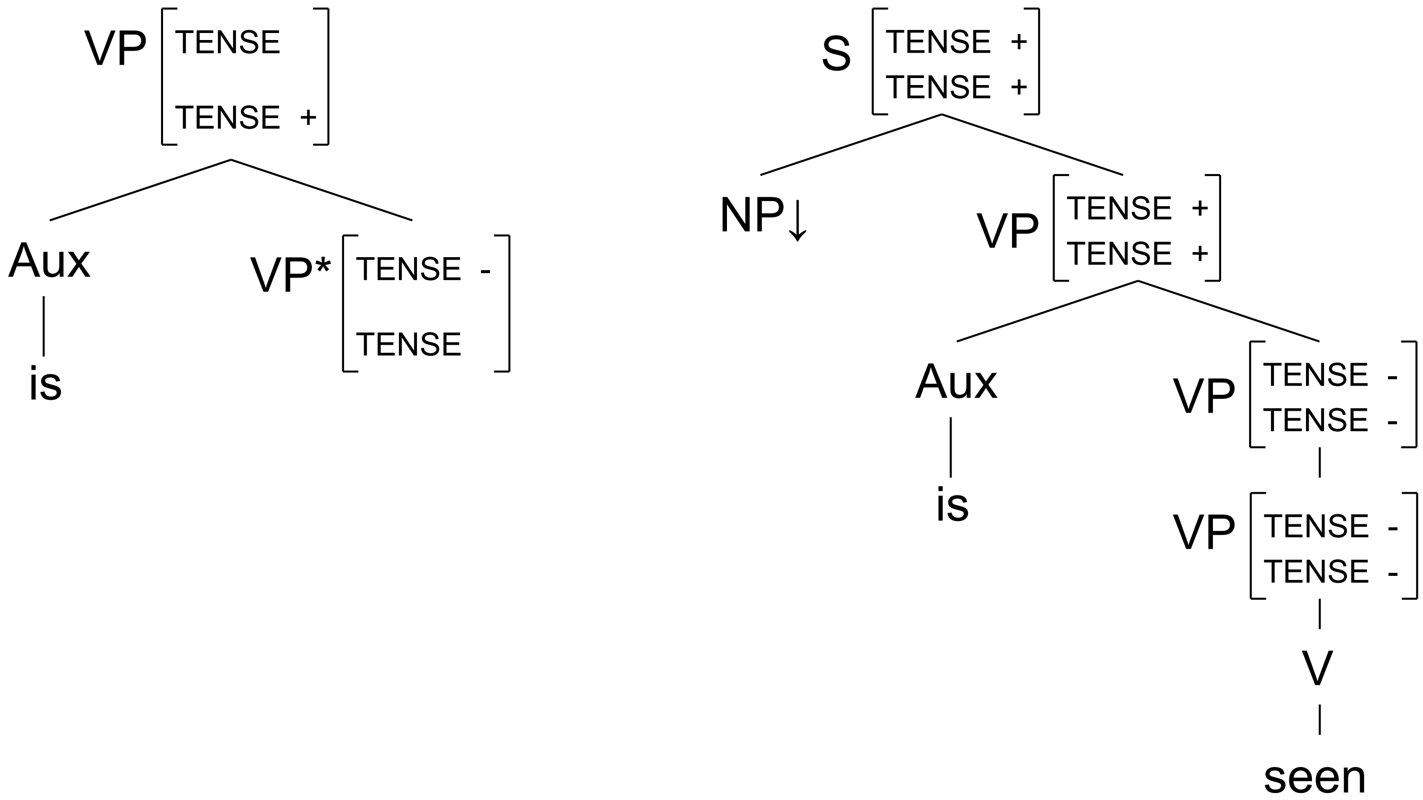
OA system — Feature system



- Above *seen's* VP node, the tree is tensed; below, it is not.
- These features do not **unify**, so the tree is not legal without adjunction

Feature structure example (cont.)

is looks for a non-tensed verb in order to make a tensed clause



Linguistic analysis

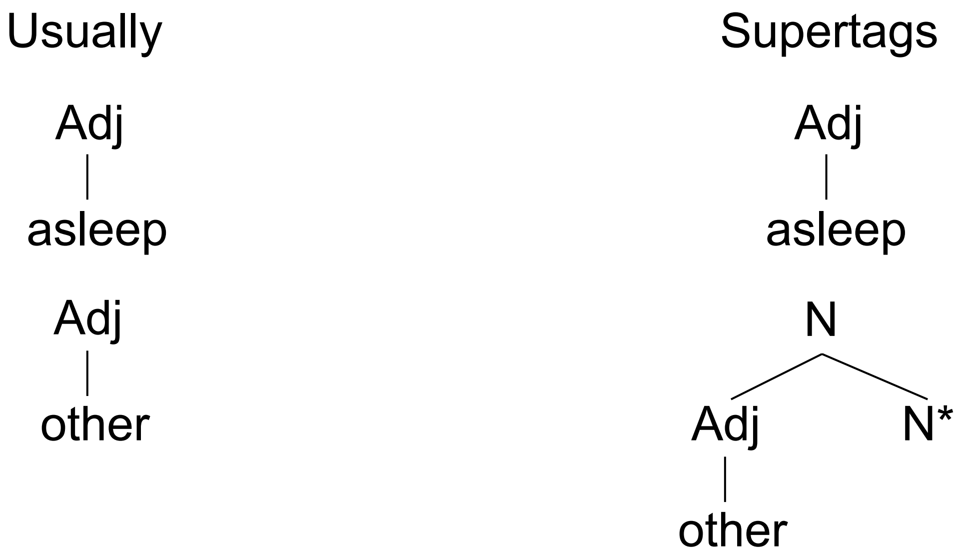
Mostly, we have just been looking at the formal description of TAGs; we need to further restrict these trees to make them match language phenomena. Some possible constraints:

- An elementary tree is the maximal syntactic projection of a lexical item
- Auxiliary trees are only used for modifiers, functional categories, predicates with verbal complements, and raising predicates
- An elementary tree is associated with a semantic meaning

We can also group elementary trees into **tree families** in order to be able to capture linguistic generalizations (right now, each lexical tree has to be individually stipulated)

Supertags

You can view a lexical entry's initial tree as a *supertag*, i.e., a part-of-speech tag with more syntactic information than usual



We can now capture distinctions between adjectives without having to specify new categories

Parsing with TAGs

The TAG formalism presents some problems for parsing (more details in the Joshi and Schabes (1997) paper if you're interested):

- Adjunction is a complicated operation because it can wrap strings around other strings

  John loves Mary can become *John probably loves Mary completely*

- Thus, more memory is required to parse a string and more operations are needed for chart parsing

Parsing with TAGs: EPDAs

Instead of regular **pushdown automata (PDAs)**, we need **embedded pushdown automata (EPDAs)** to store the parse information

- Pushdown automaton: puts items on a stack

  S → NP VP finds an NP and has VP S on a stack, meaning that once a VP is found, then an S has been completed

- Embedded pushdown automaton: puts stacks of items on a stack

  Can have a stack of NPs on the stack which will then match Dutch verbs appropriately

Parsing with TAGs: Tree traversal

How one traverses a tree in parsing a TAG grammar is important

- Cannot simply use bottom-up tree traversal → have to go in a left-to-right manner
- This left-to-right manner allows one to find adjoining nodes

References

Abeillé, Anne and Owen Rambow (2000). Tree Adjoining Grammar: An Overview. In Anne Abeillé and Owen Rambow (eds.), *Tree Adjoining Grammars: Formalisms, Linguistic Analyses and Processing*, Stanford, CA: CSLI Publications, pp. 1–68.

Joshi, Aravind K. and Yves Schabes (1997). Tree Adjoining Grammars. In A. Saloma and G. Rosenberg (eds.), *Handbook of Formal Languages and Automata*, Heidelberg: Springer-Verlag.