Grammar Formalisms - TAG

Yoav Goldberg

(most slides are by Julia Hockenmaier,
some slides by Anoop Sarkar)

Tree-Adjoining Grammar

(Lexicalized) Tree-Adjoining Grammar

- **TAG is a tree-rewriting formalism:**
  - TAG defines operations (**substitution**, **adjunction**) on trees.
  - The **elementary objects** in TAG are trees (not strings)

- **TAG is lexicalized:**
  - Each elementary tree is **anchored** to a lexical item (word)
  - “**Extended domain of locality**”:
    The elementary tree contains all arguments of the anchor.
  - TAG requires a linguistic theory which specifies the shape of these elementary trees.

- **TAG is mildly context-sensitive:**
  - can capture Dutch cross-serial dependencies
  - but is still efficiently parseable

AK Joshi and Y Schabes (1996) Tree Adjoining Grammars.

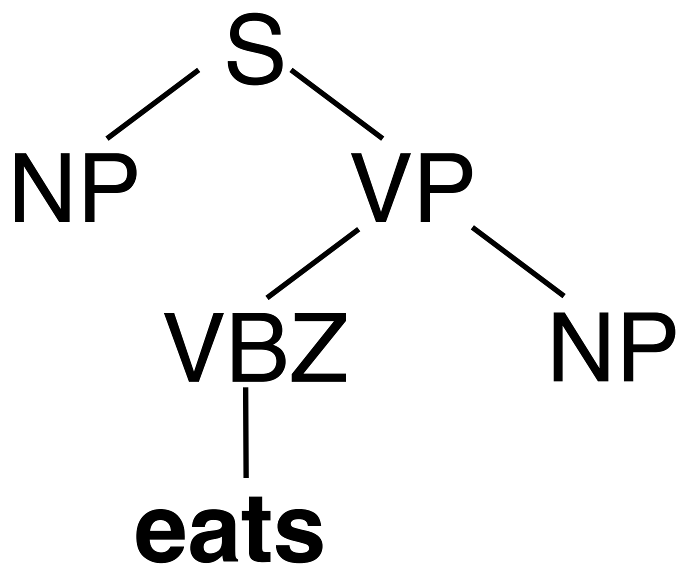
Domain of locality

- In a CFG, the domain of locality is confined to a single rule.

- Each local tree is independent.

Extended domain of locality

- We want to capture all arguments of a word in a single elementary object.
- We also want to retain certain syntactic structures (e.g. VPs).
- Our elementary objects are tree fragments:

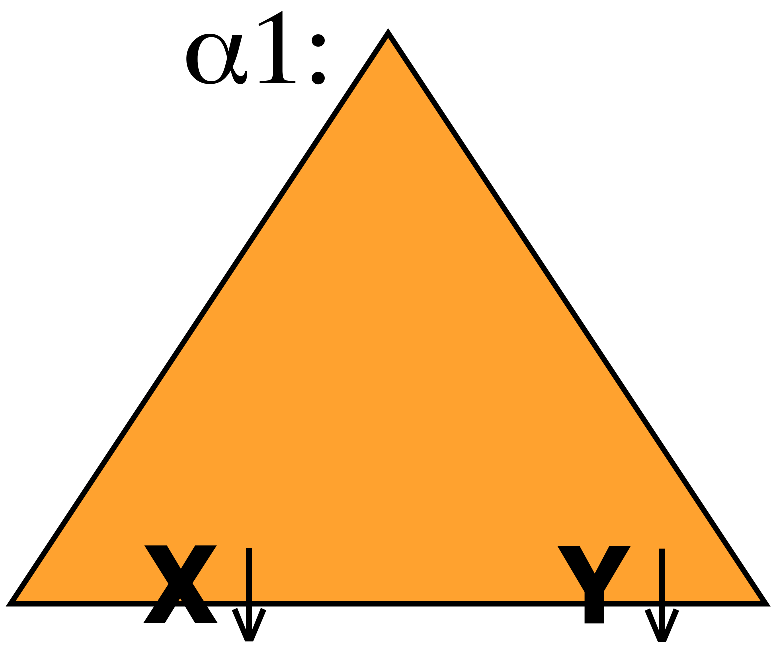


Lexicalized TAG

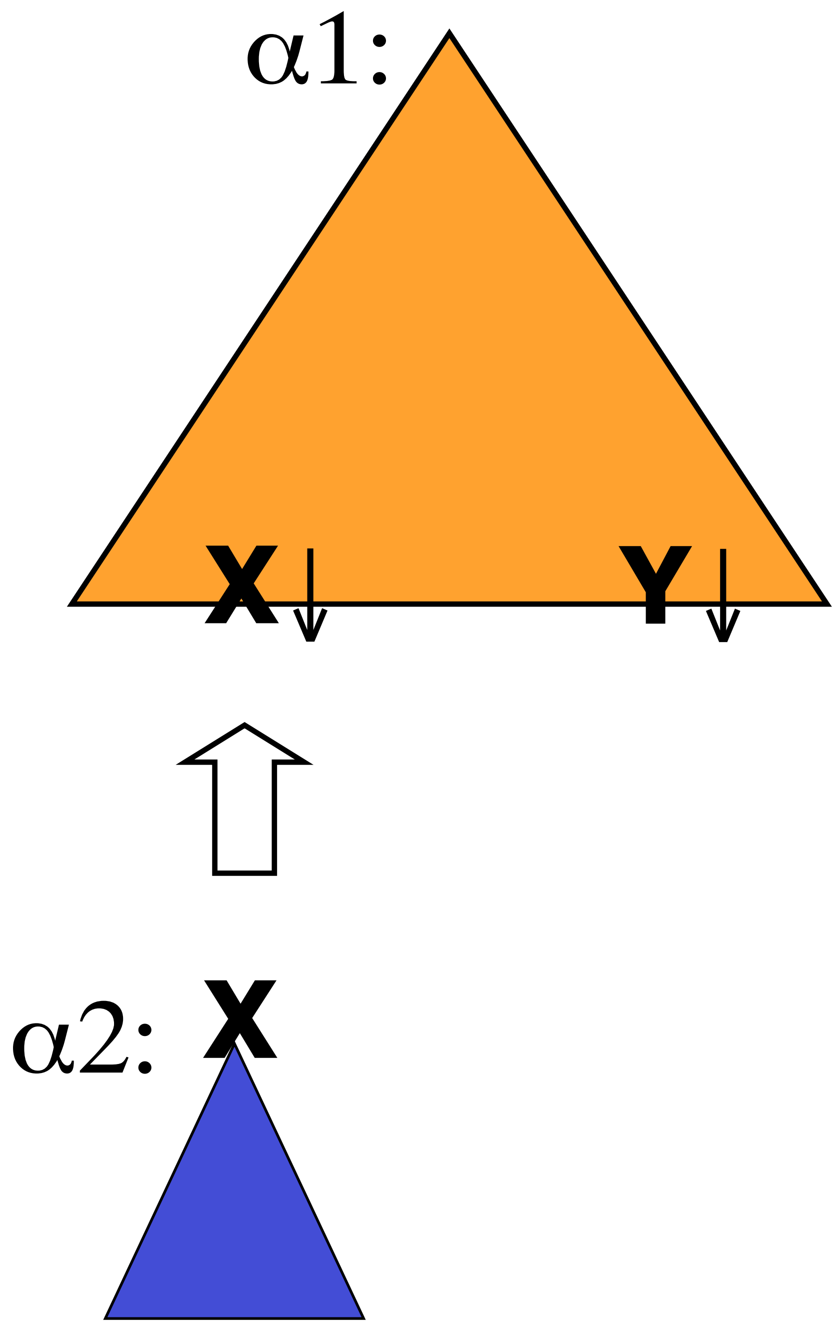
- A Lexicalized TAG (LTAG) is a TAG where each elementary tree has at least one terminal symbol as a leaf node
- A non-lexicalized TAG can always be converted to a lexicalized TAG (Joshi & Schabes, 1997)
- Lexicalization has some useful effects:
  - finite ambiguity: corresponds to our intuition about NL ambiguities,
  - statistical dependencies between words can be captured which can improve parsing accuracy

TAG substitution (arguments)

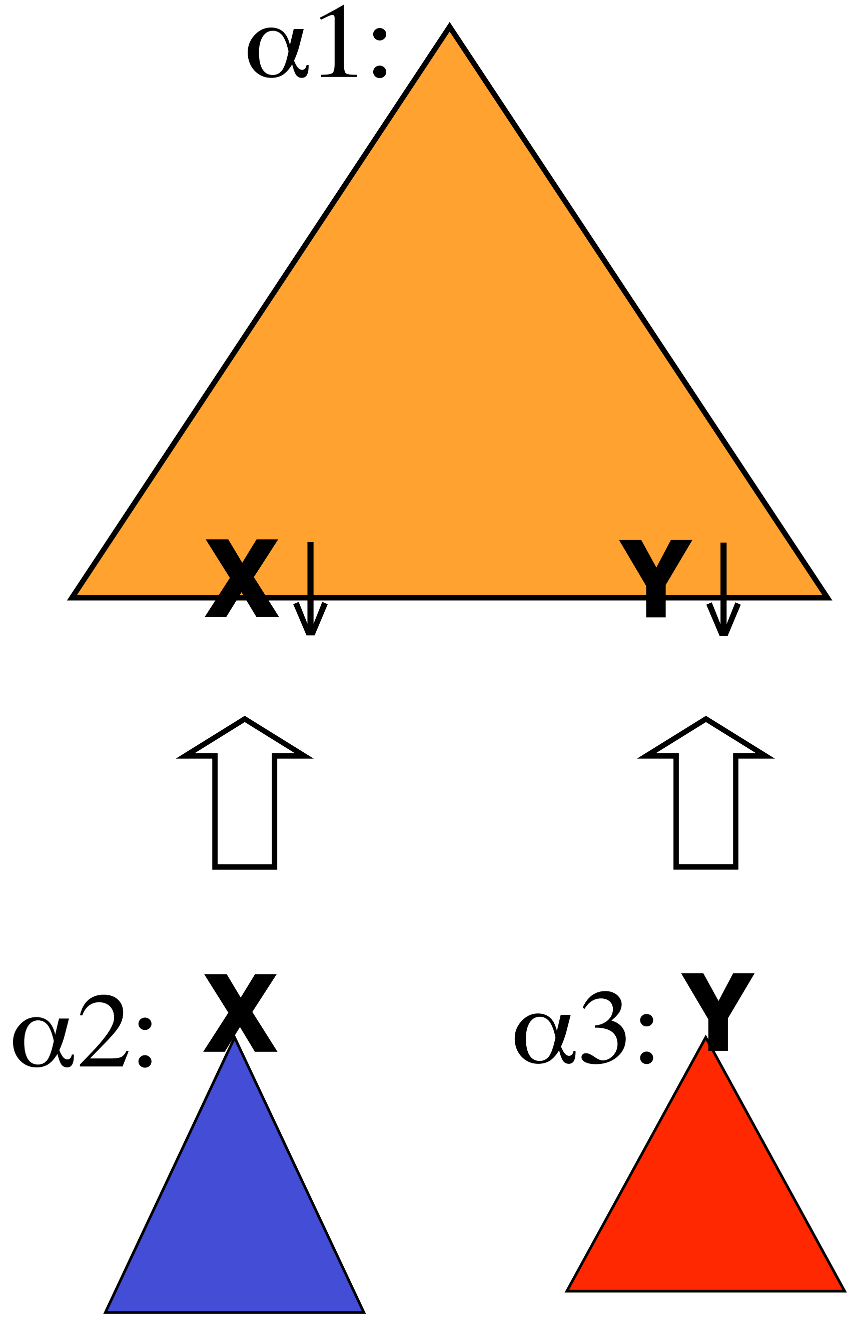
TAG substitution (arguments)



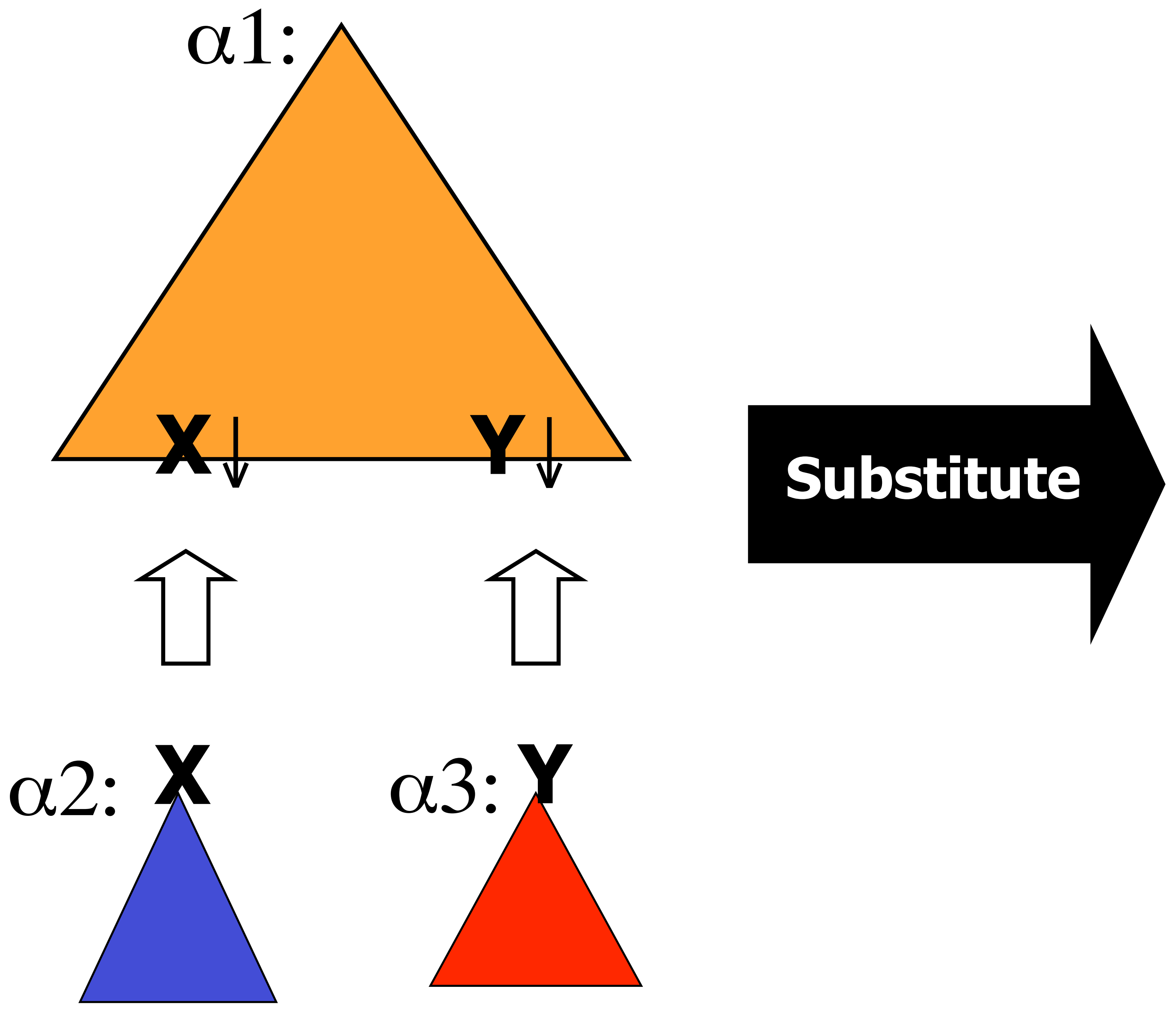
TAG substitution (arguments)



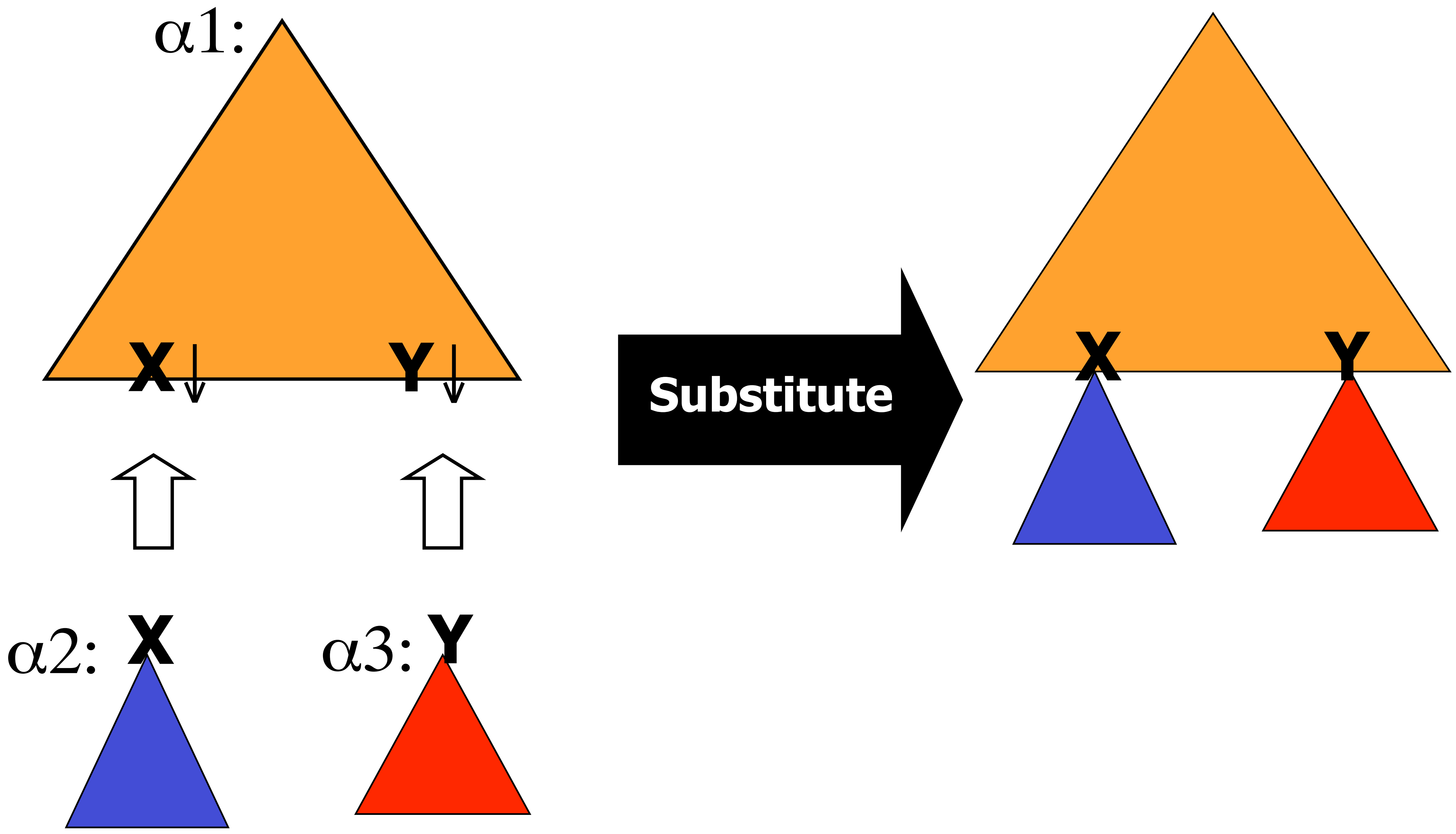
TAG substitution (arguments)



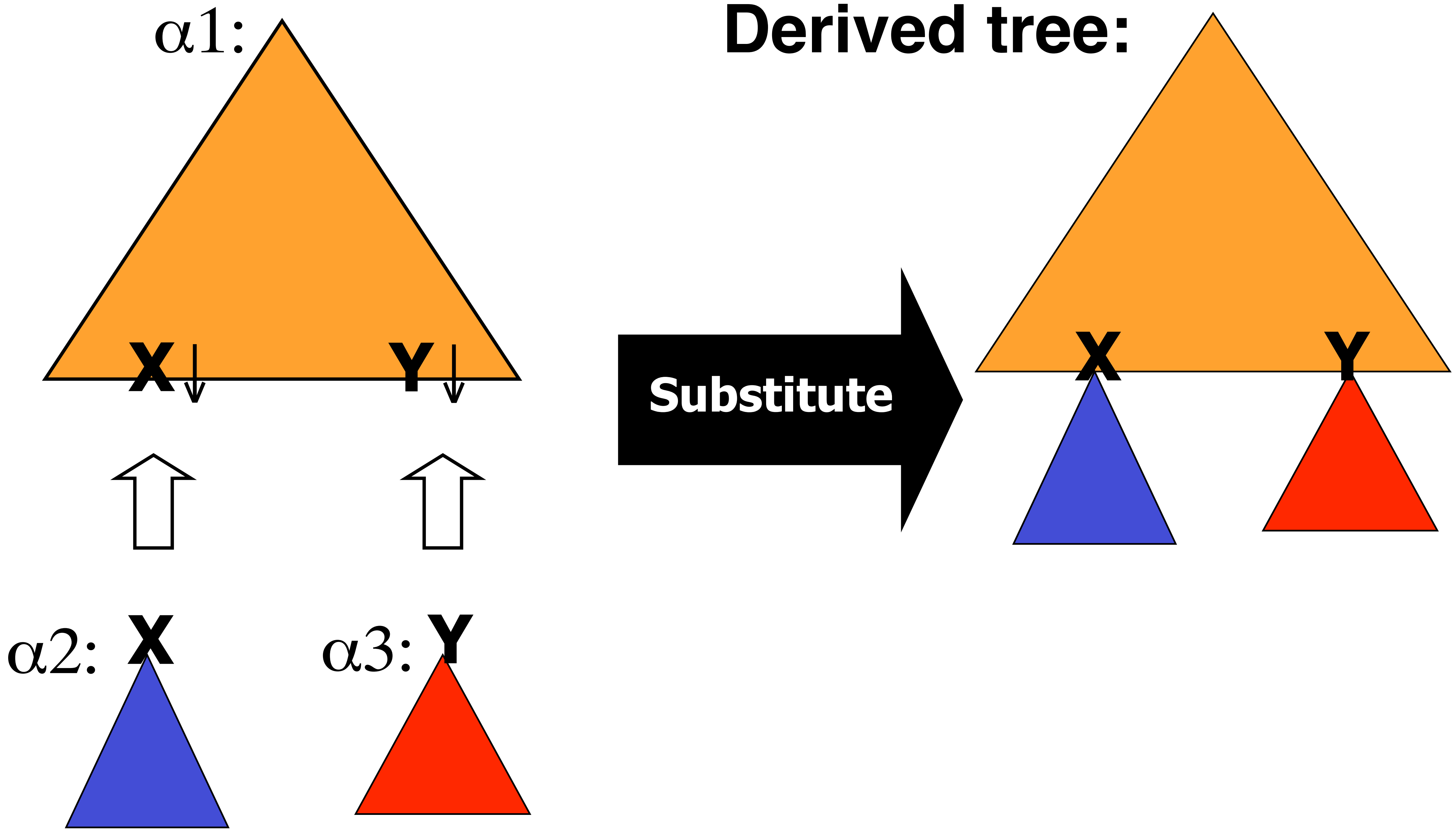
TAG substitution (arguments)



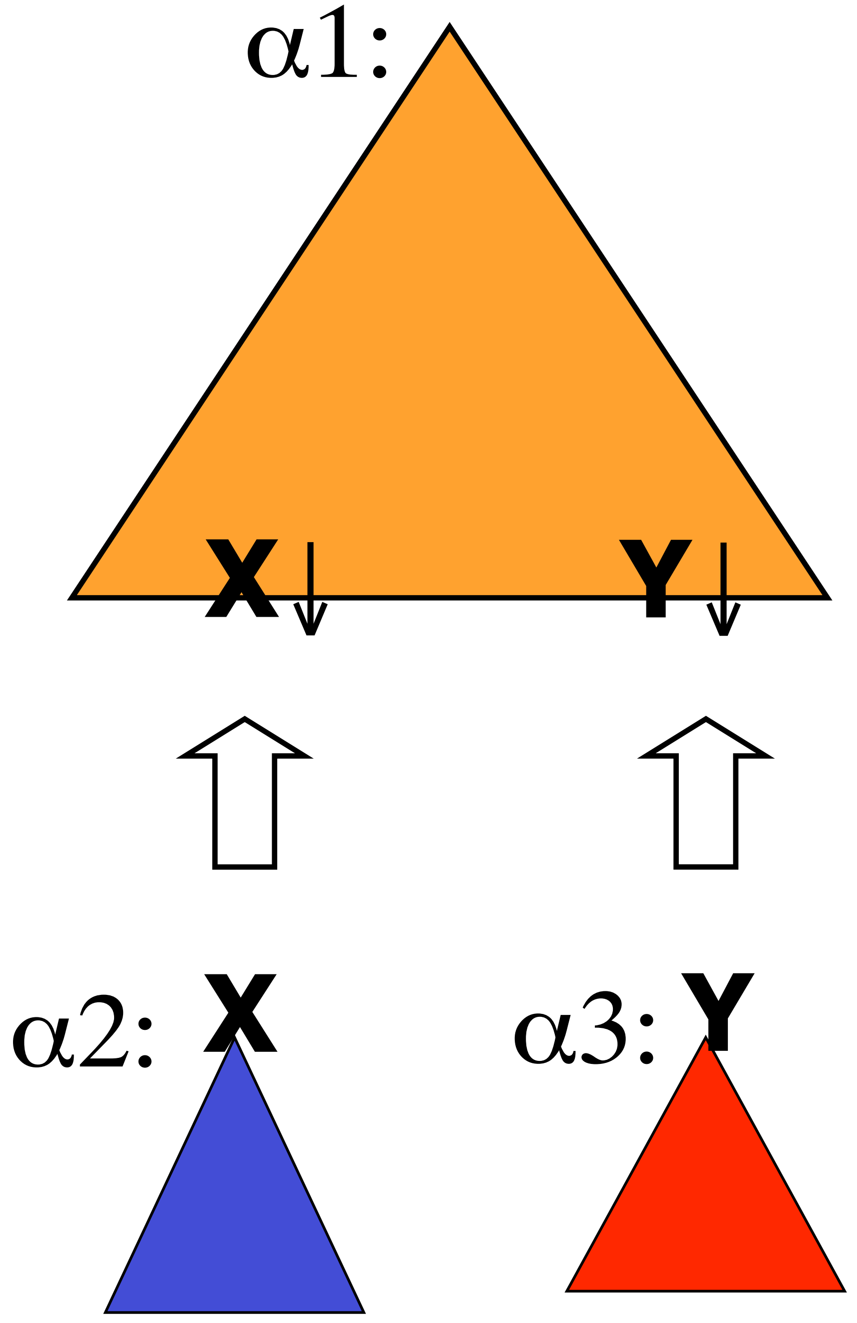
TAG substitution (arguments)



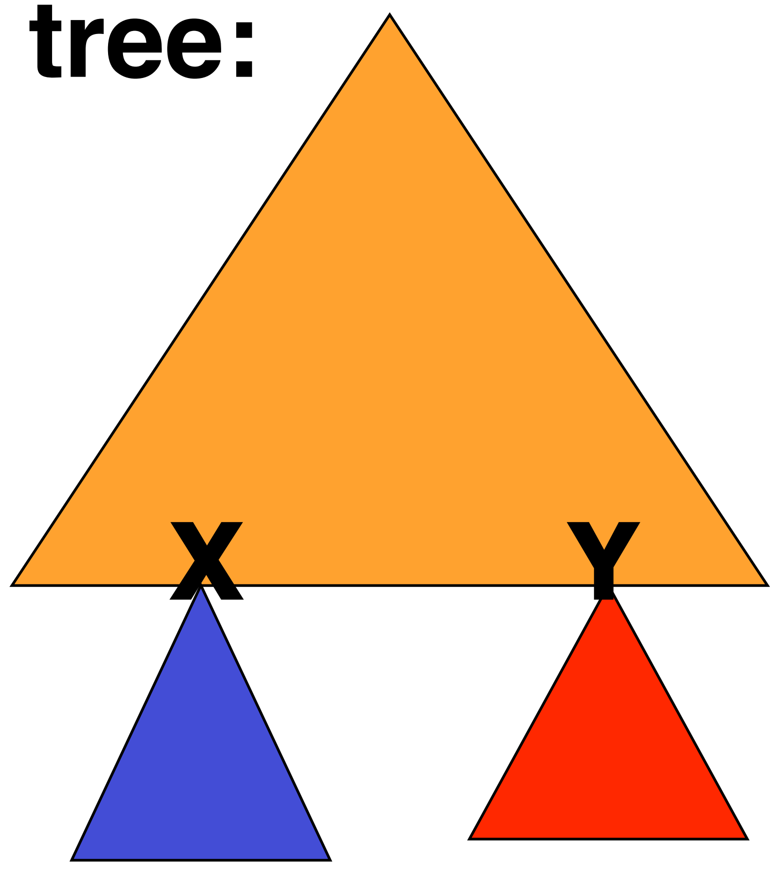
TAG substitution (arguments)



TAG substitution (arguments)



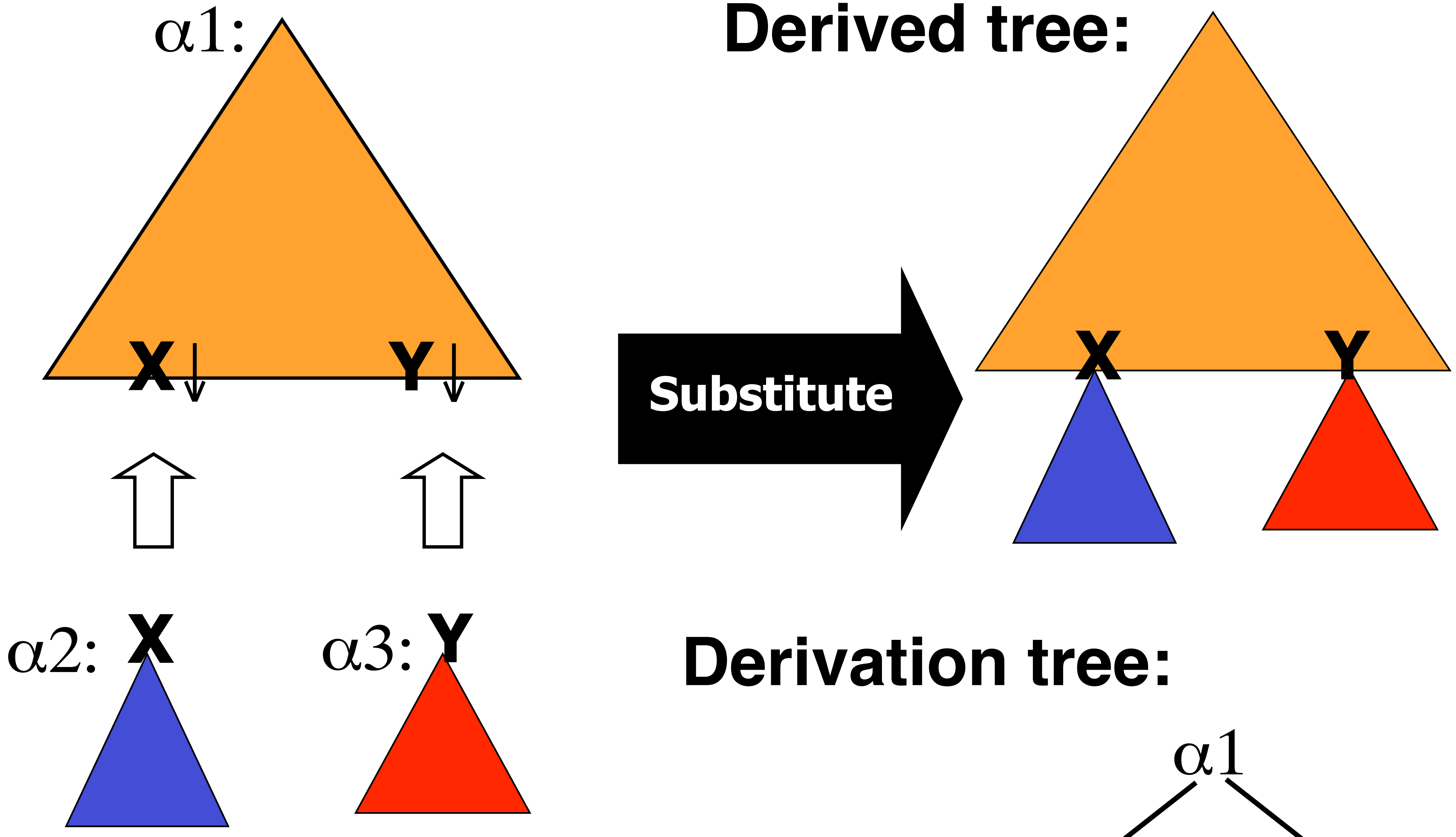
Derived tree:



Derivation tree:

TAG substitution (arguments)

Derived tree:



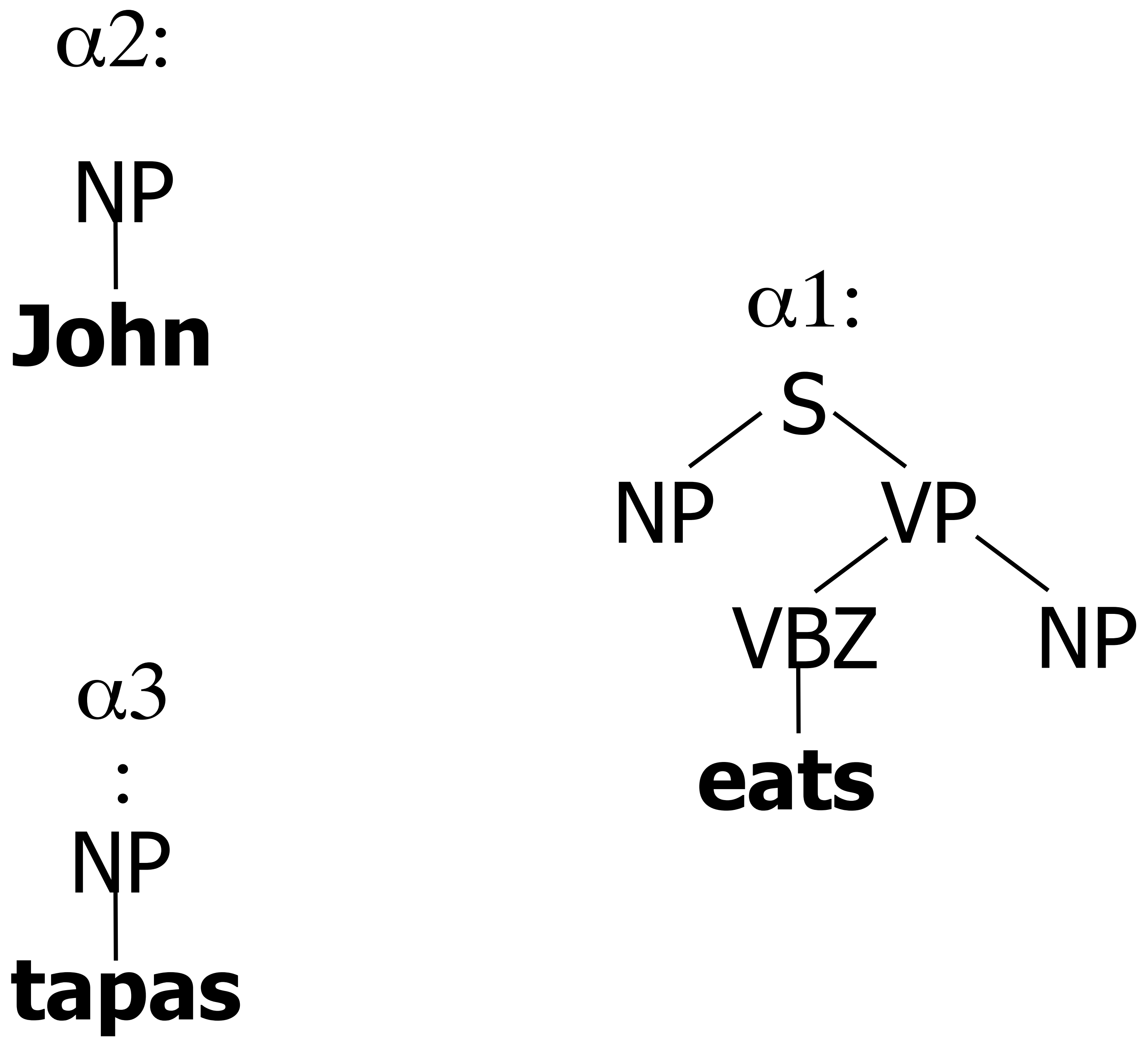
Derivation tree:

$$\begin{tikzcd}
 & \alpha 1 \arrow[dl, dash] \arrow[dr, dash] & \\
\alpha 2 & & \alpha 3
\end{tikzcd}$$

Tree-Substitution Grammar

- **TAG without adjunction**
  = Tree-substitution grammar.
  - elementary objects = trees.
  - recursive operation: substitution
- **Substitution alone does not give us anything beyond context-free grammar.**

A small TSG lexicon

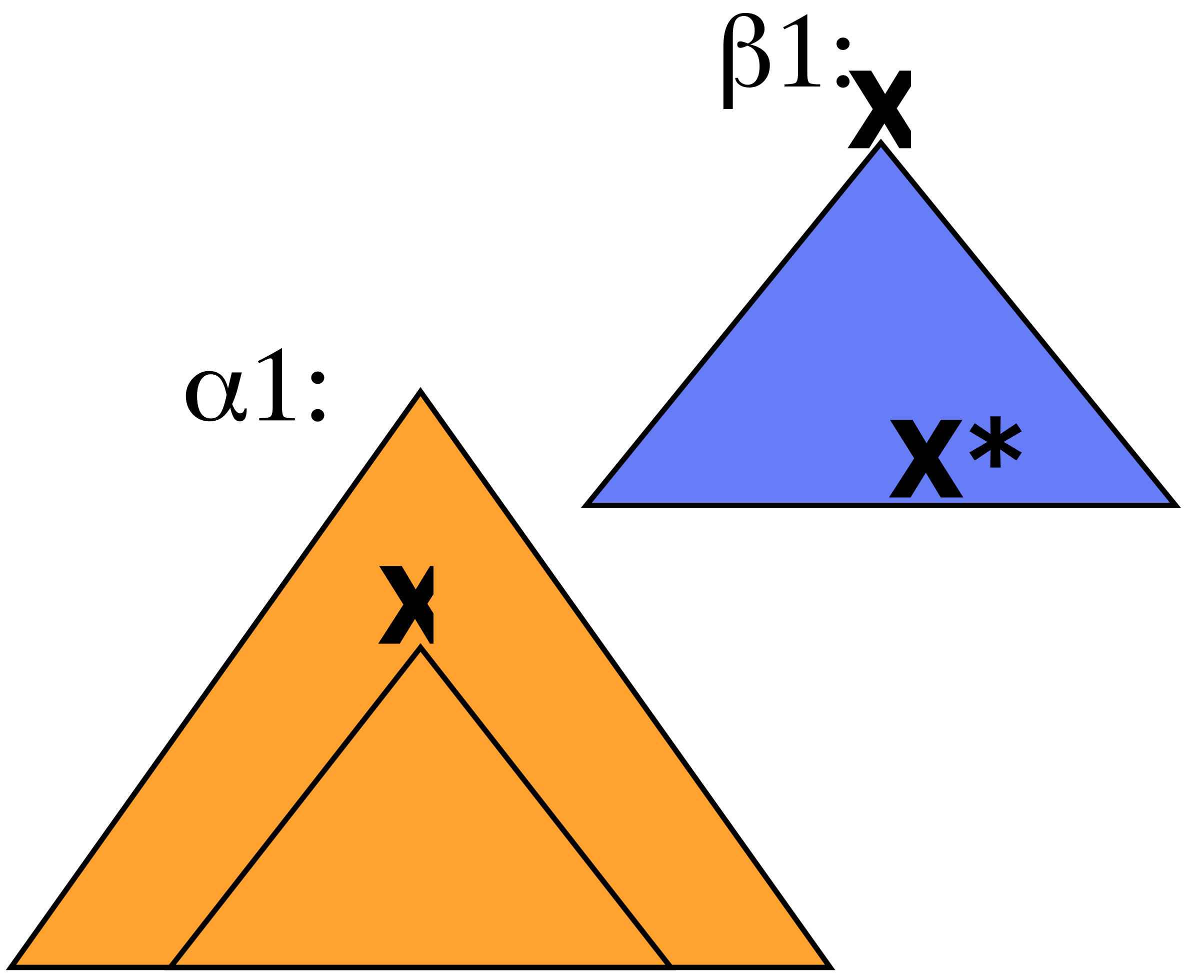


What about adjuncts?

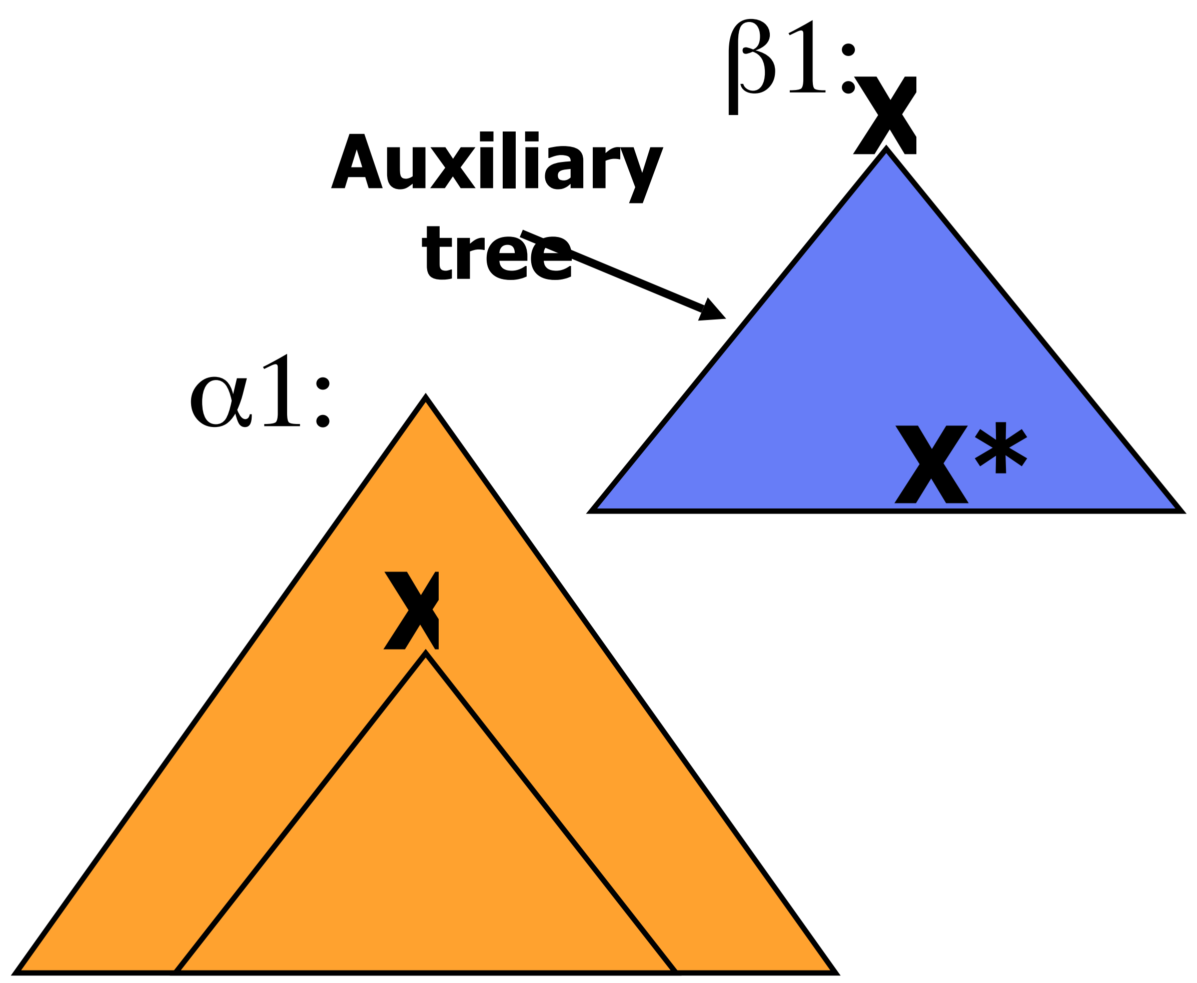
- **Can we generate**

  John always eats tapas.
  John always eats tapas when he’s in Spain.
  John always eats tapas for dinner when he’s in Spain.

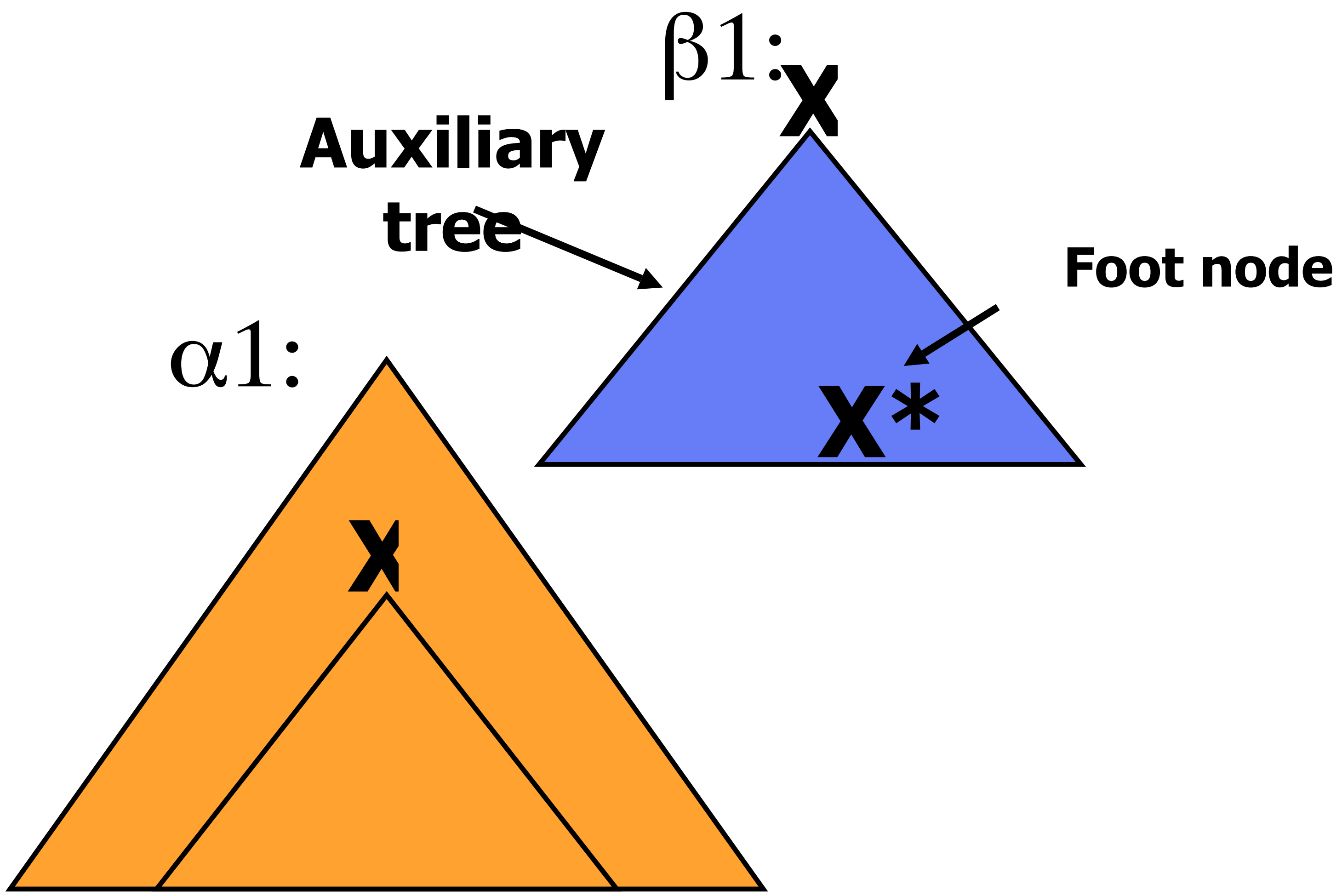
TAG adjunction



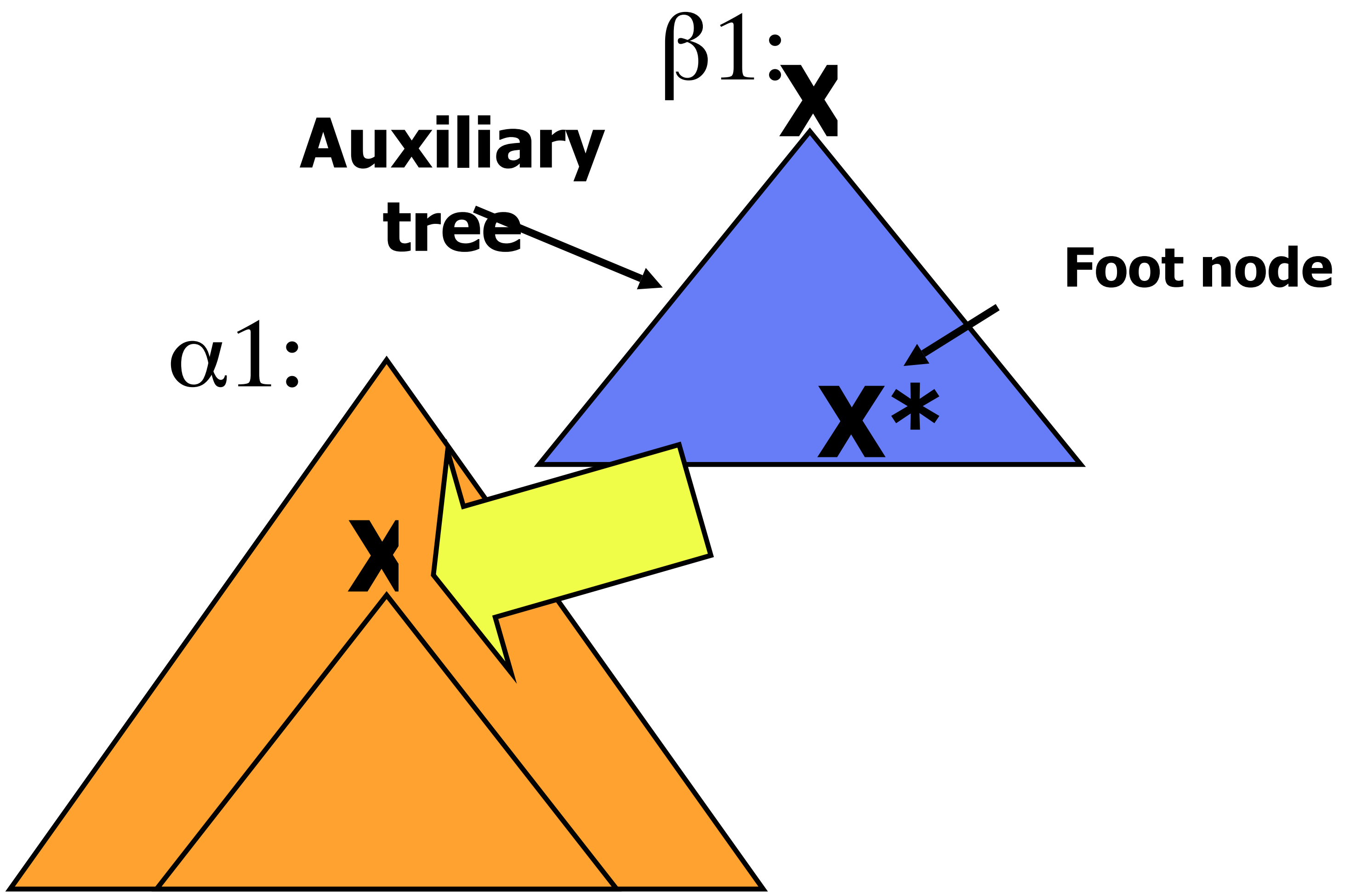
TAG adjunction



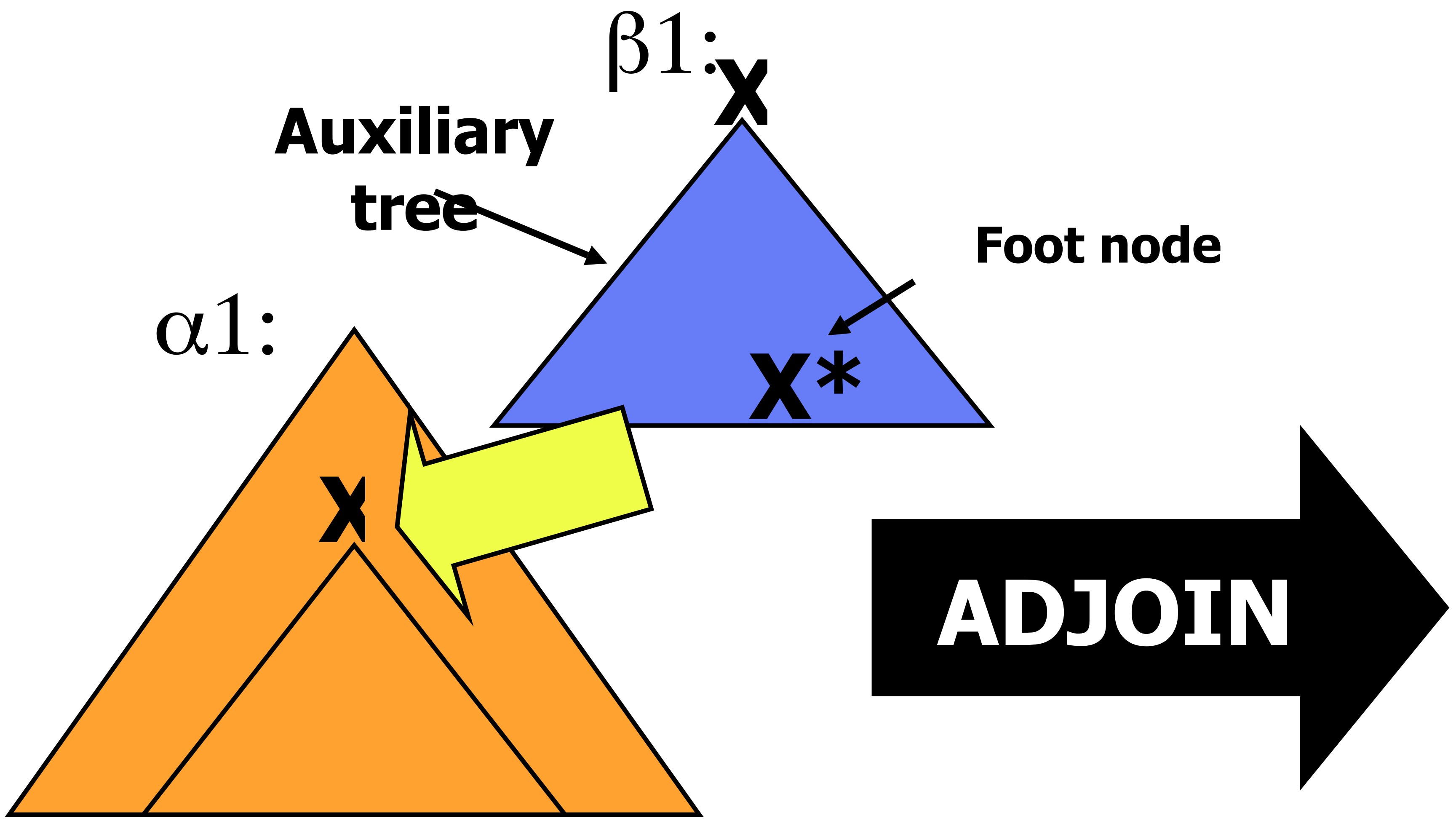
TAG adjunction



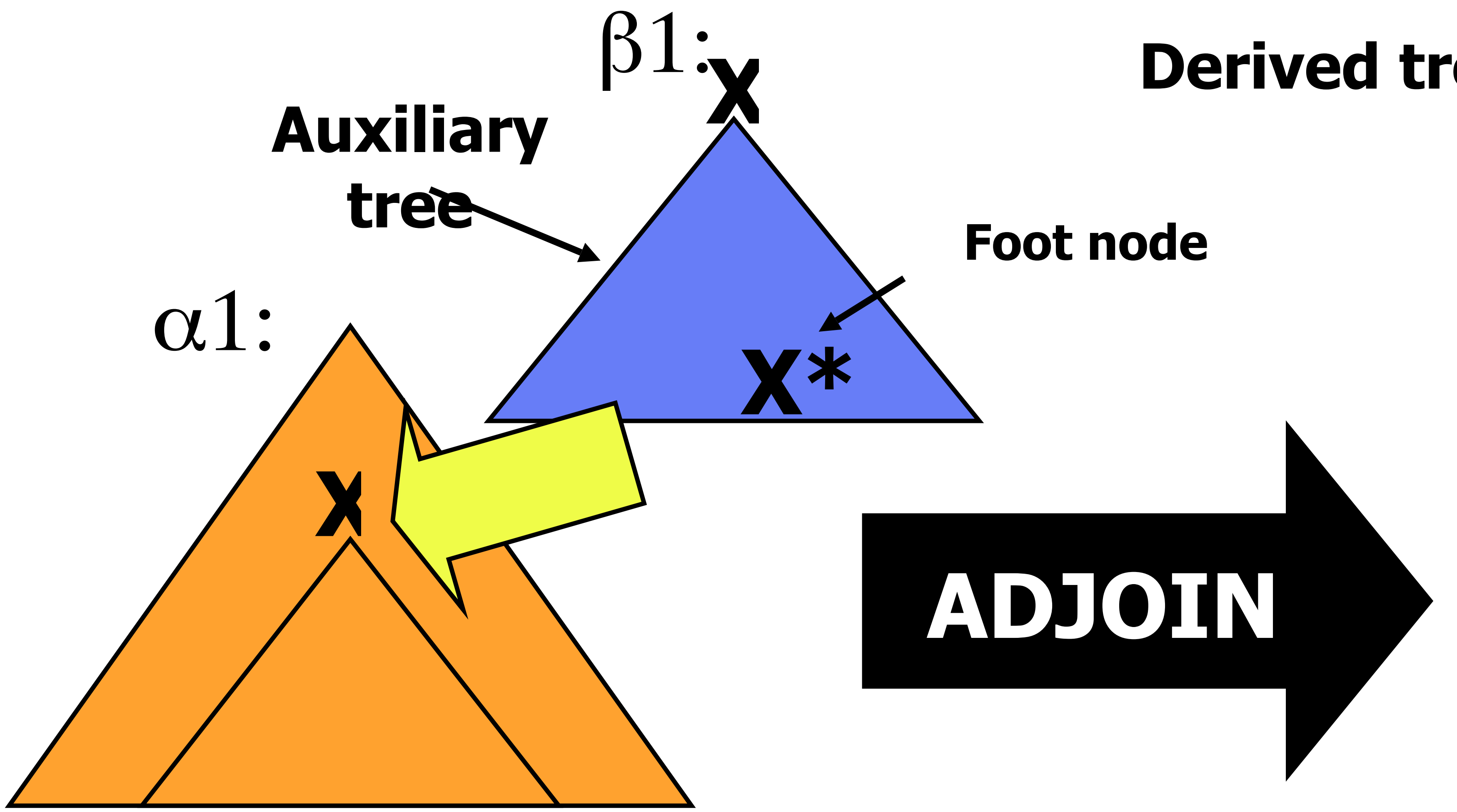
TAG adjunction



TAG adjunction

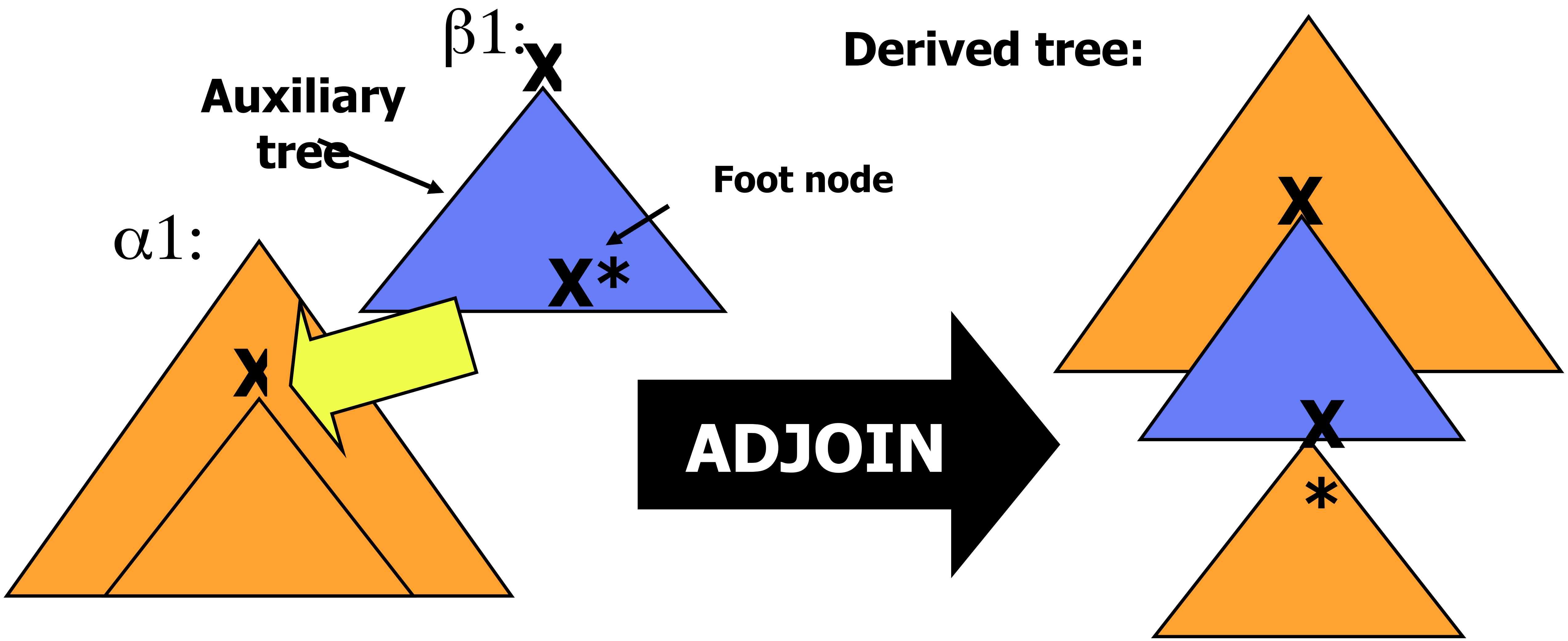


TAG adjunction

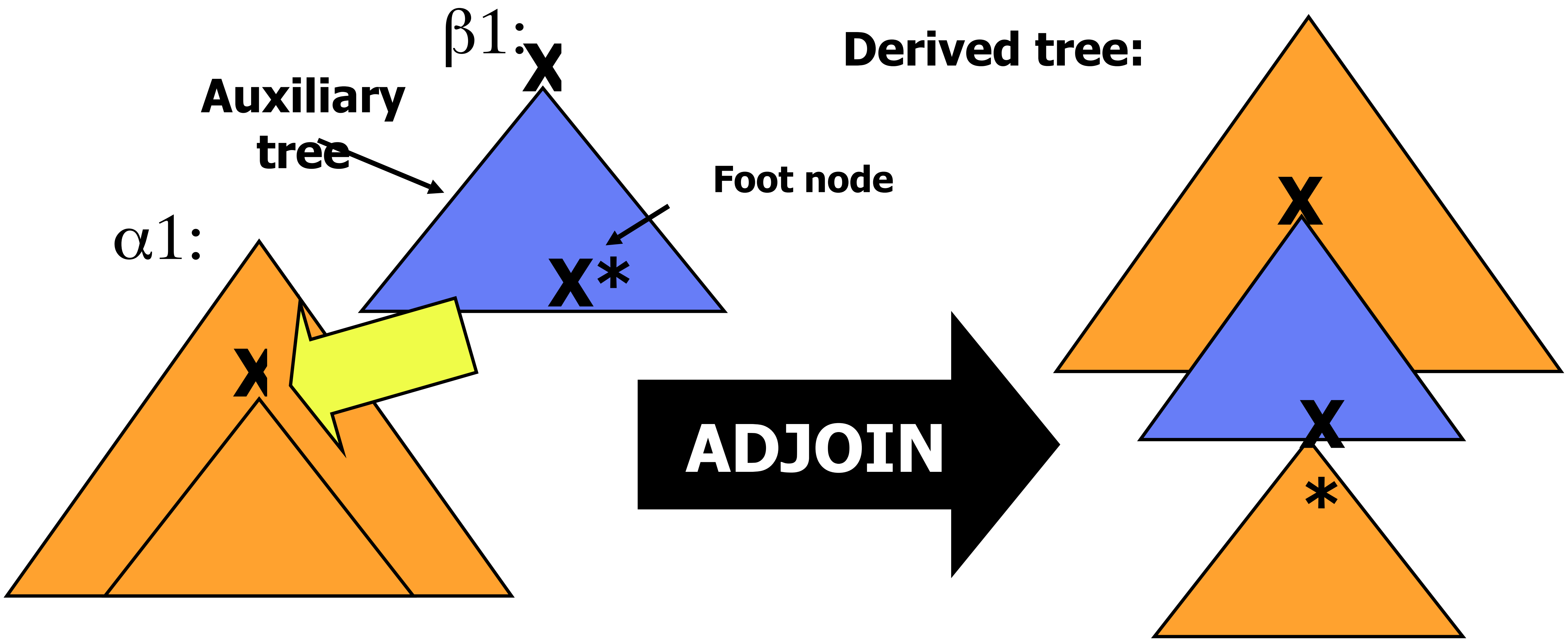


Derived tree:

TAG adjunction

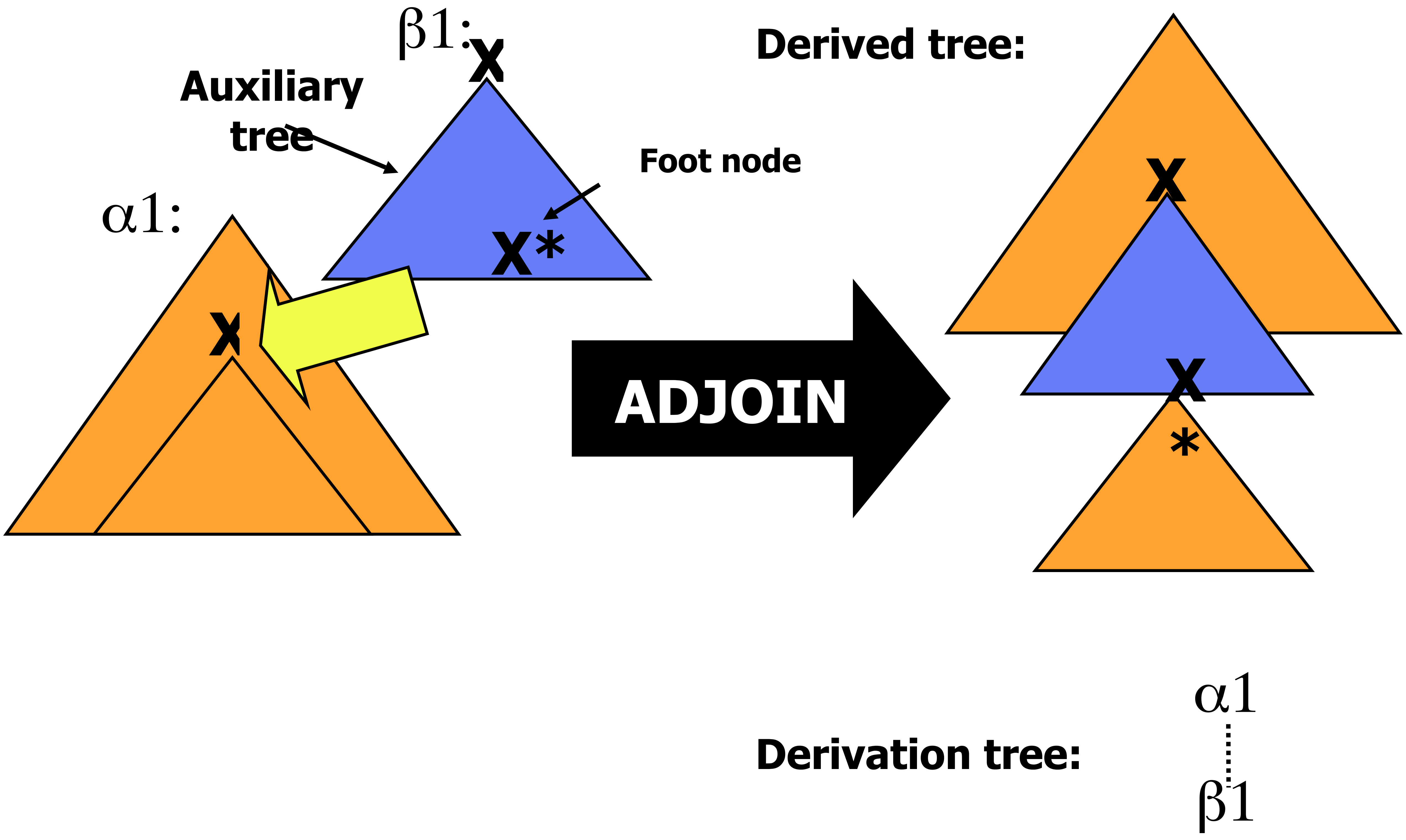


TAG adjunction



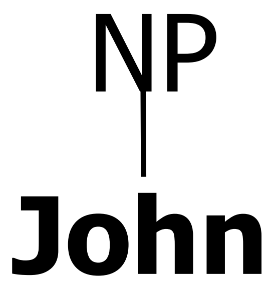
Derivation tree:

TAG adjunction

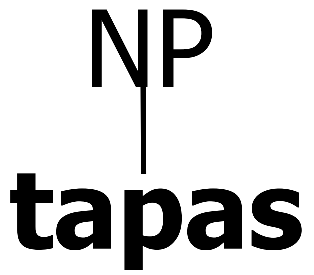


A small TAG lexicon

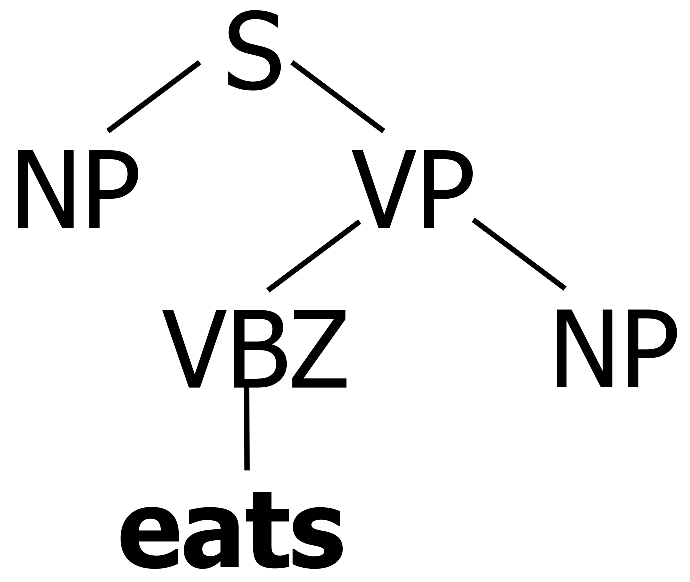
α2:



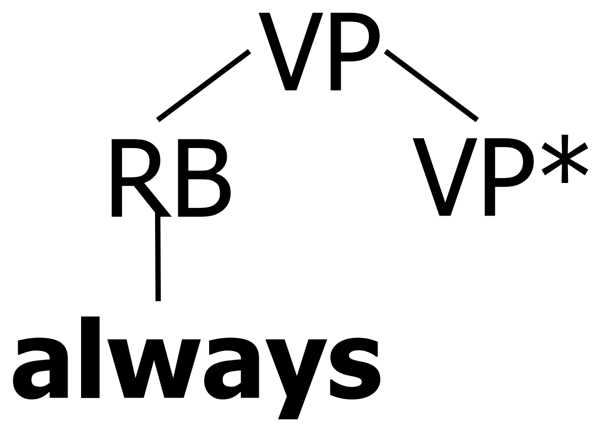
α3
:



α1:

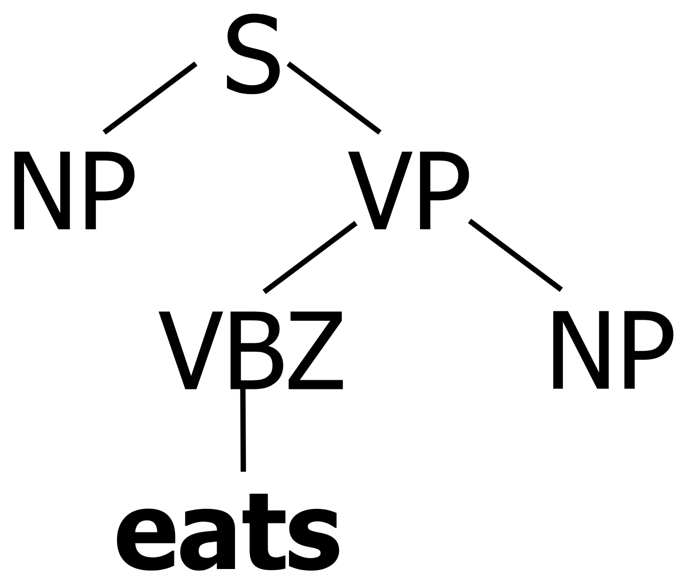


β1:

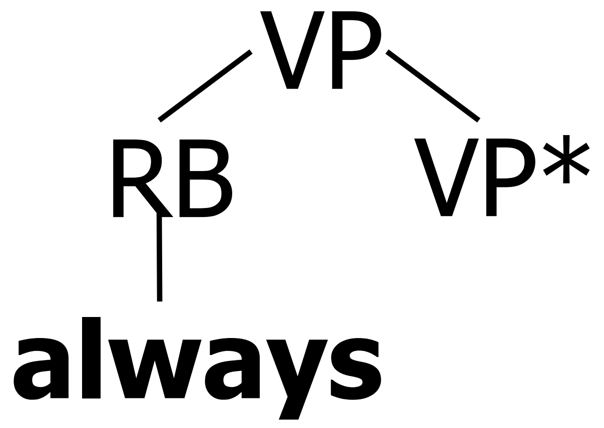


A TAG derivation

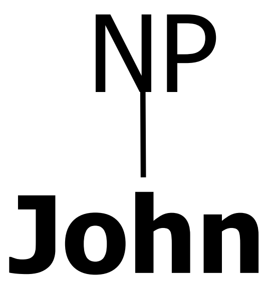
$\alpha 1$:



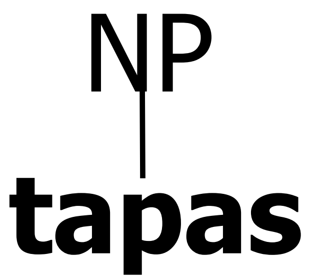
$\beta 1$:



$\alpha 2$:

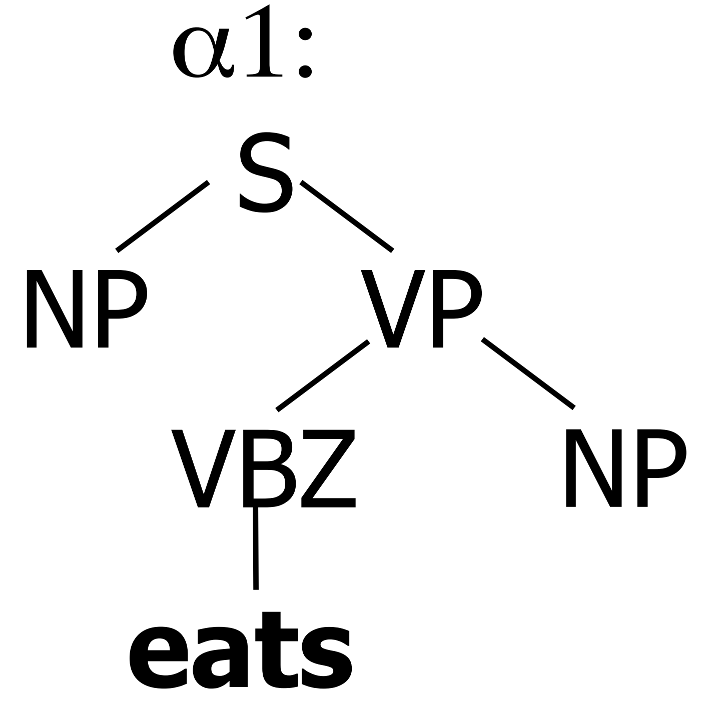


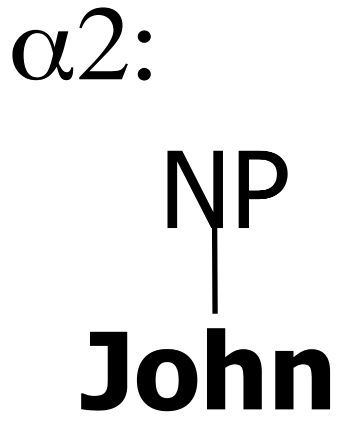
$\alpha 3$:

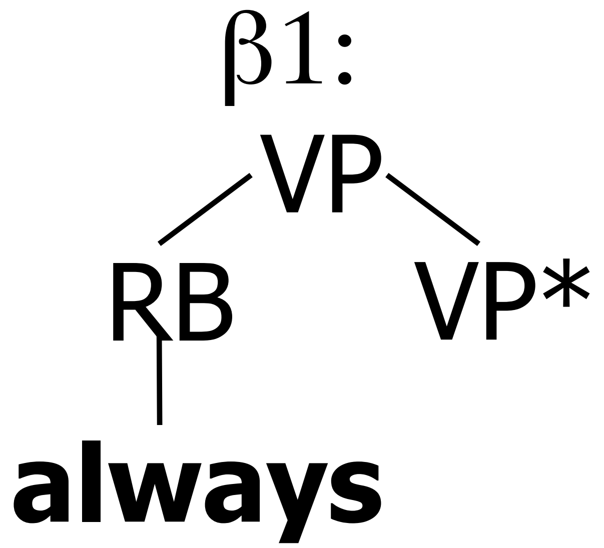


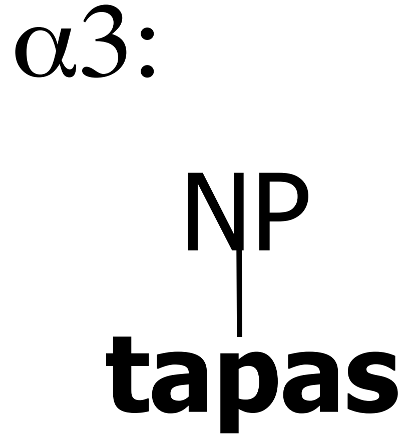
A TAG derivation

α1



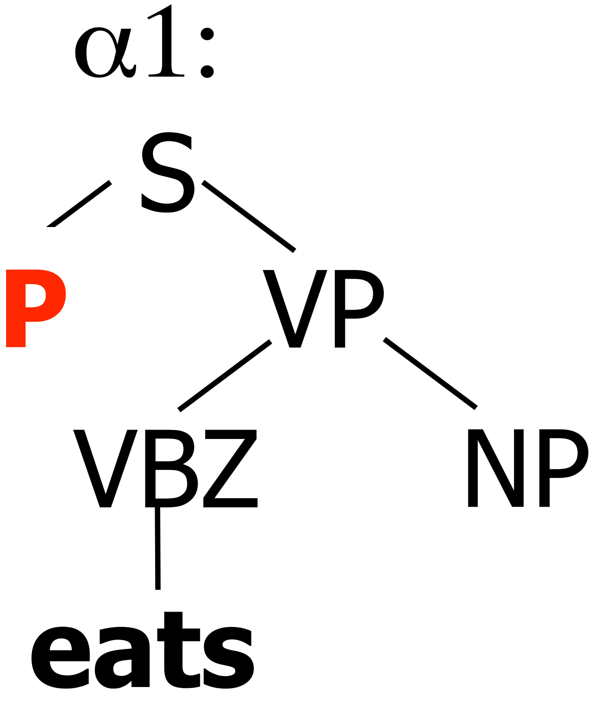


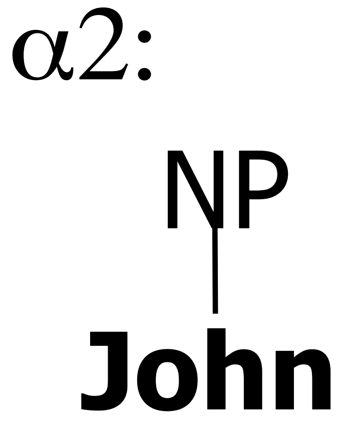


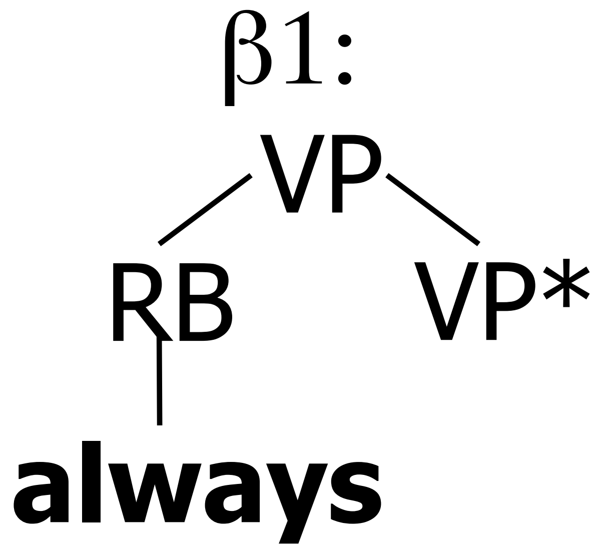


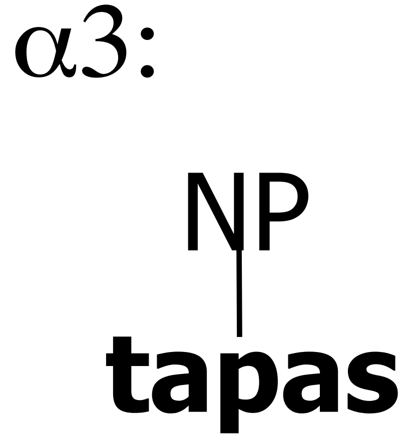
A TAG derivation

α1



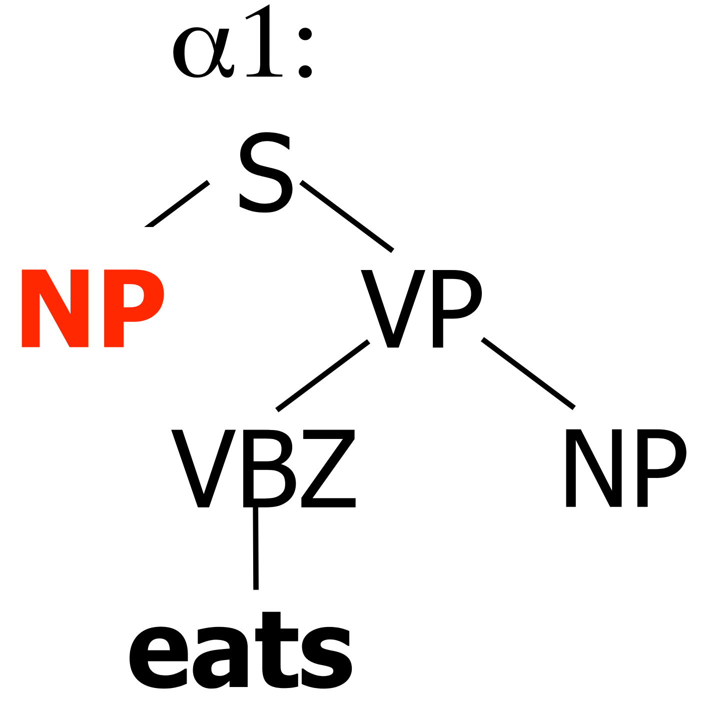


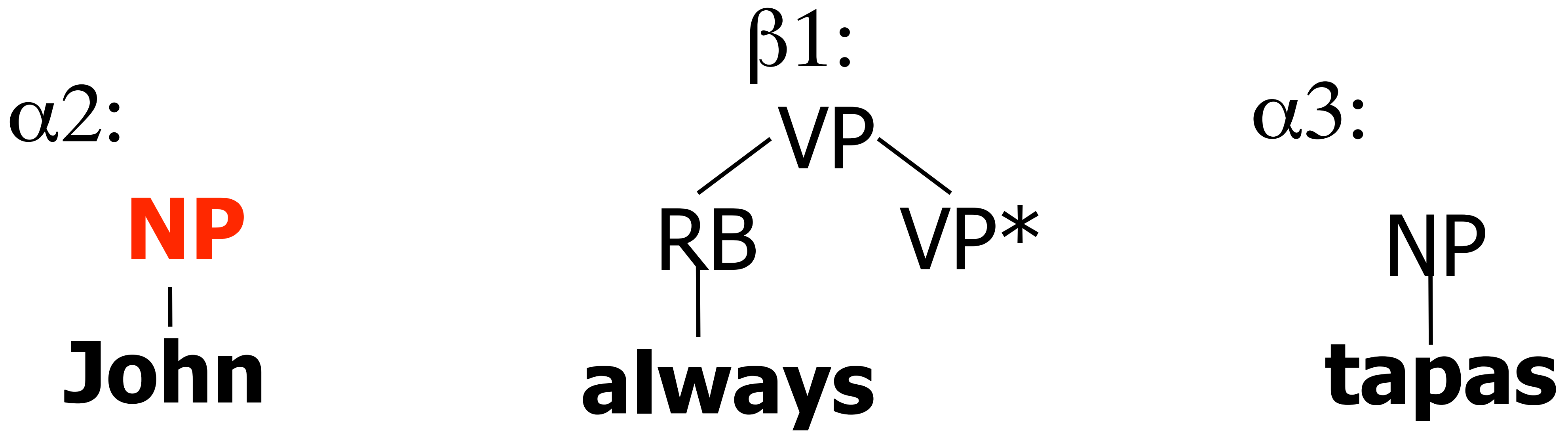




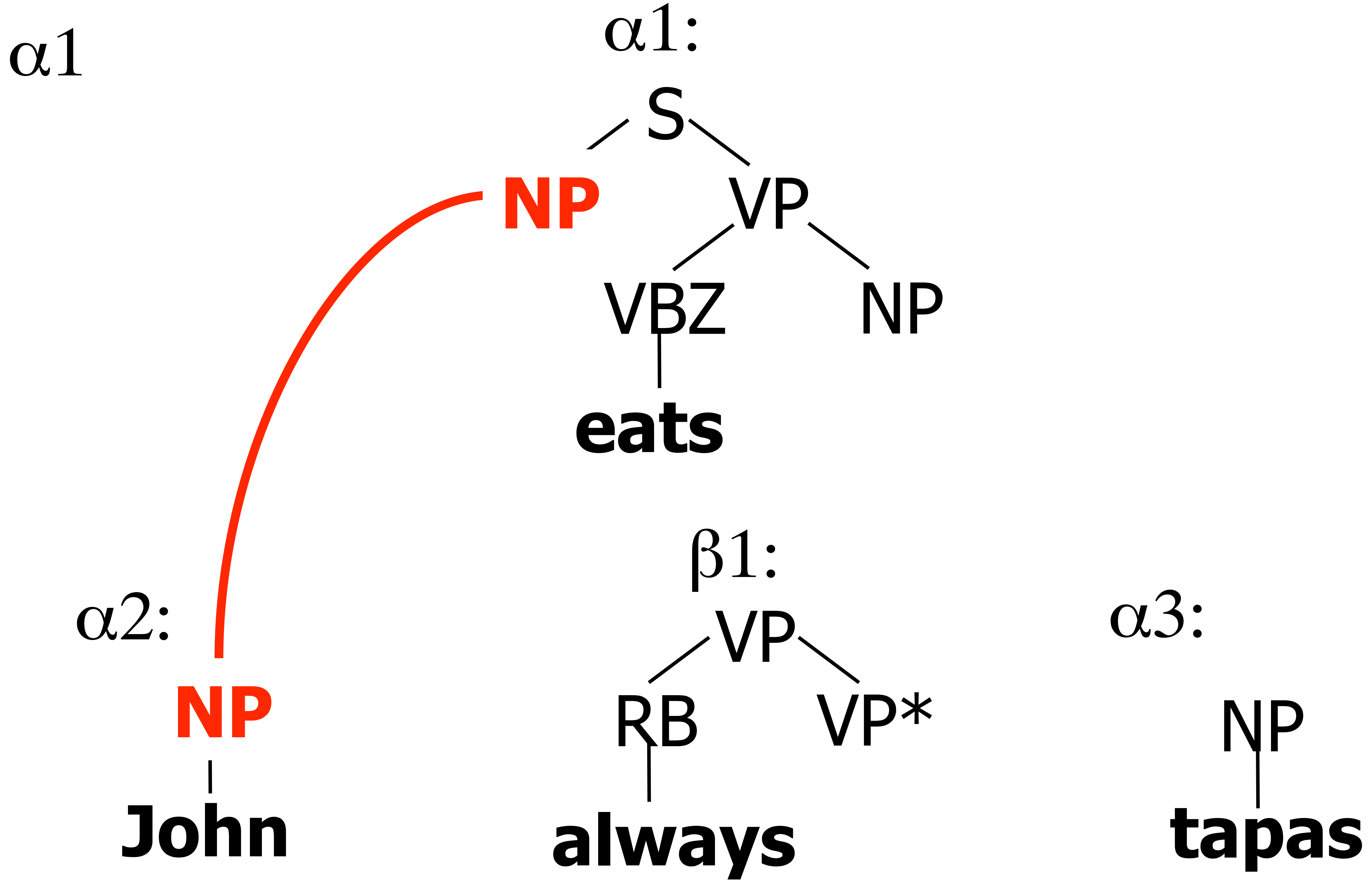
A TAG derivation

α1

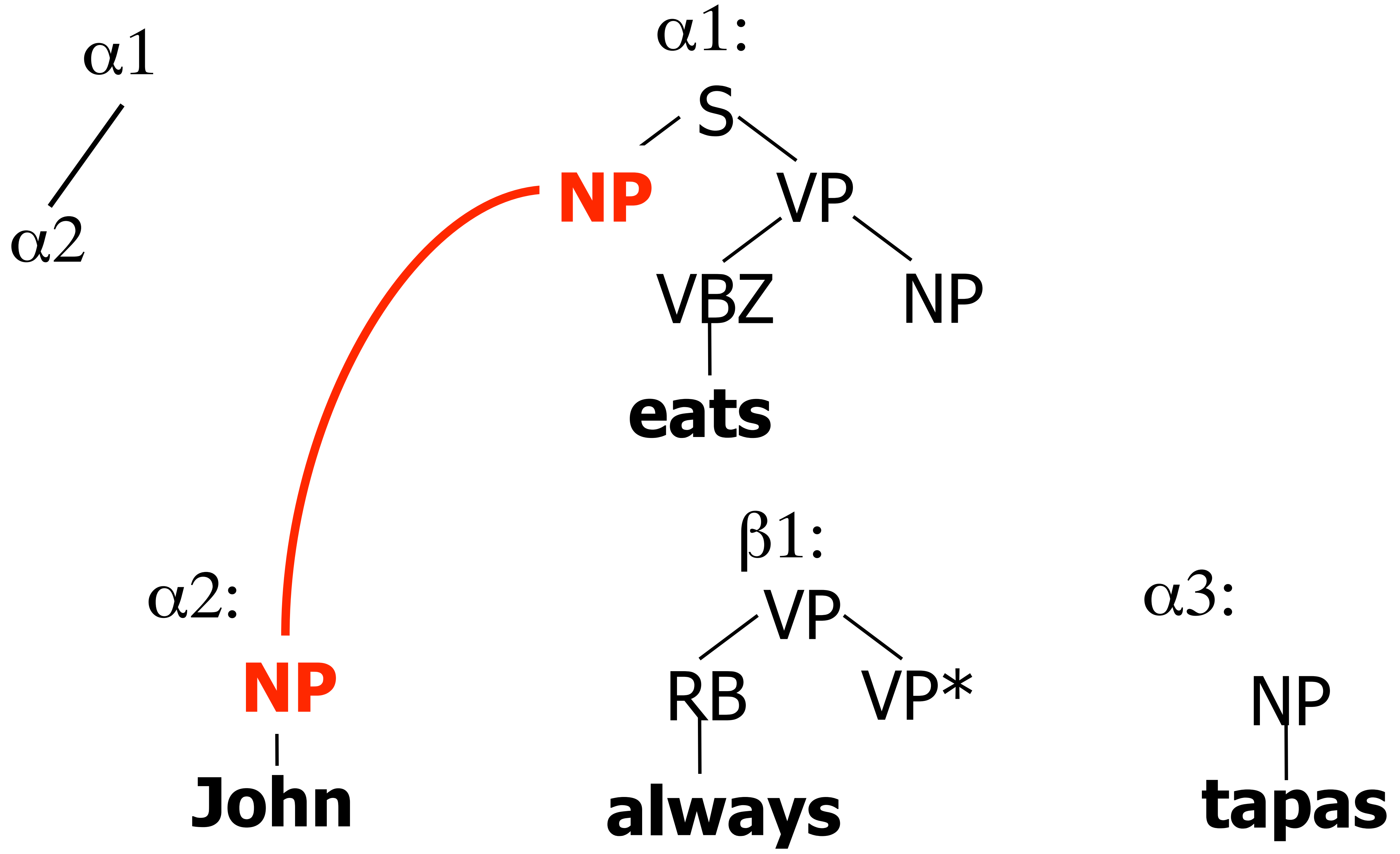




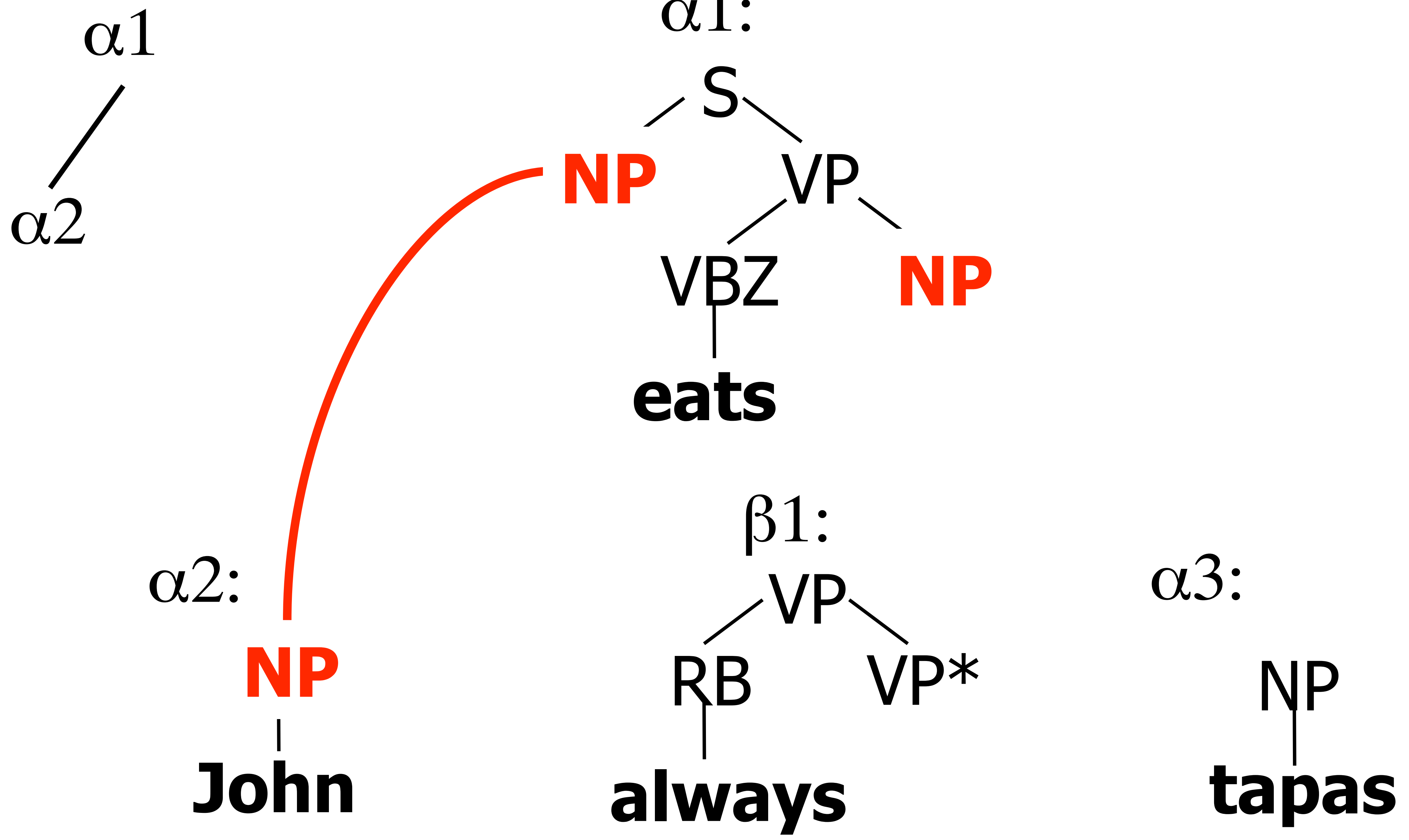
A TAG derivation



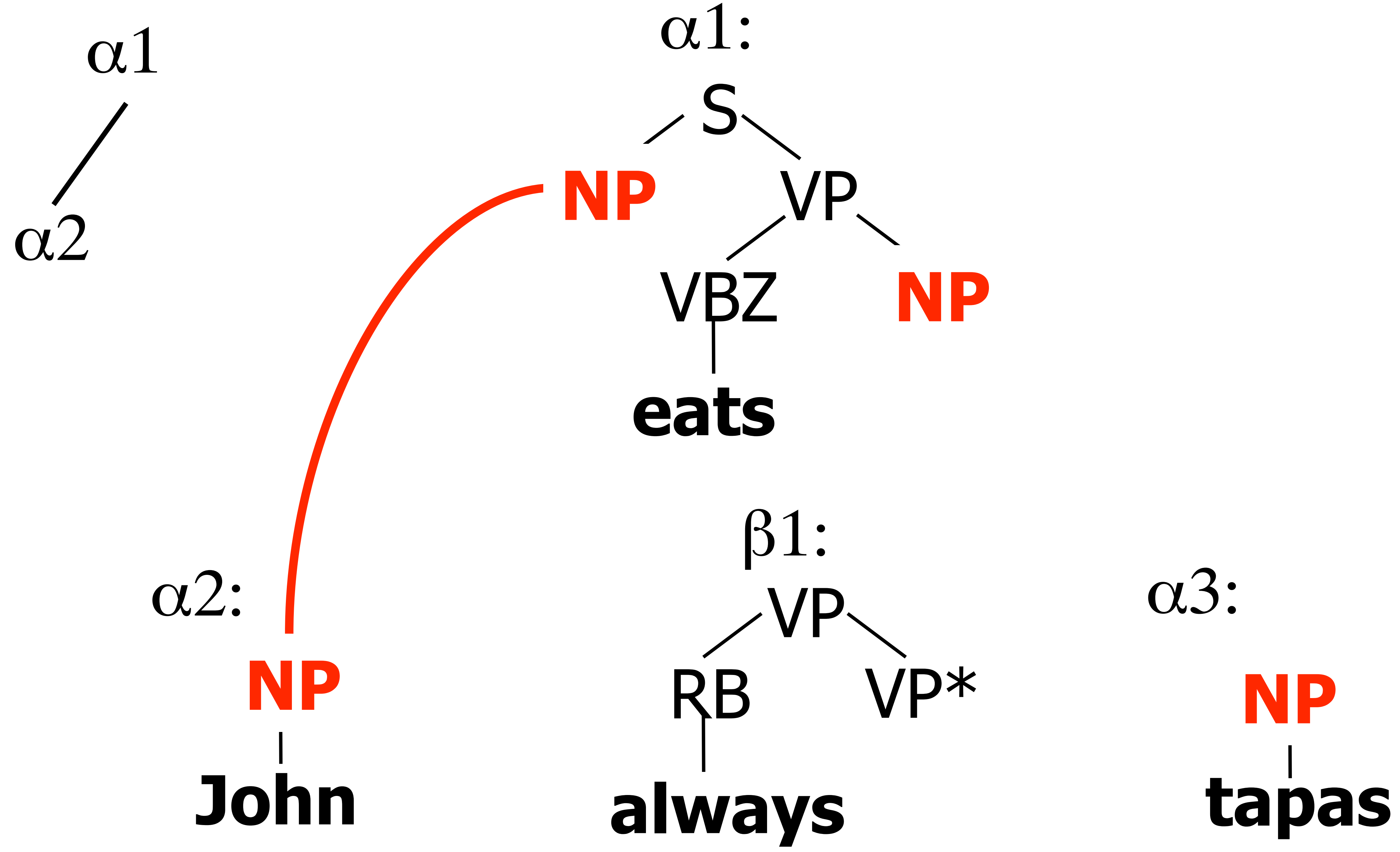
A TAG derivation



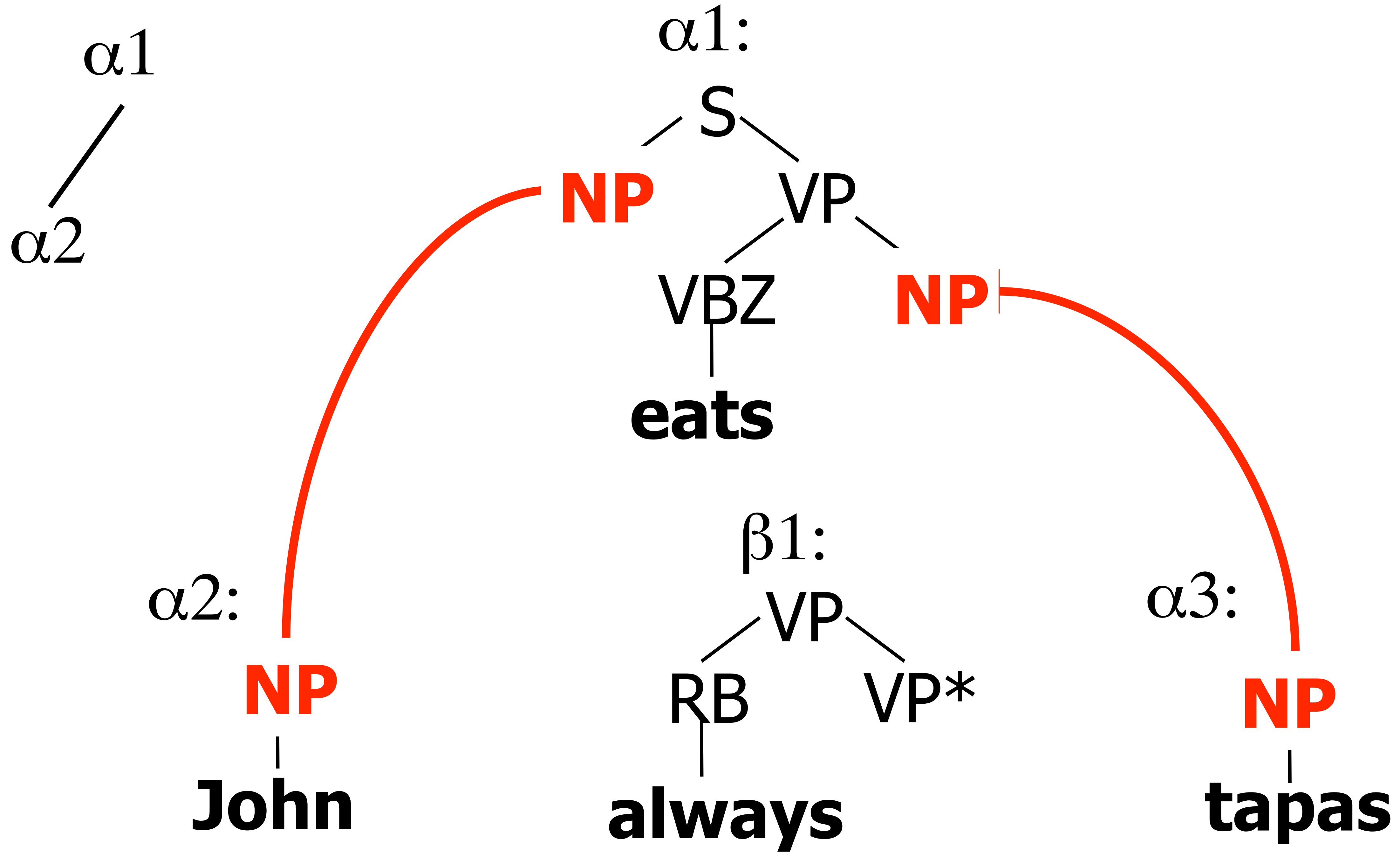
A TAG derivation



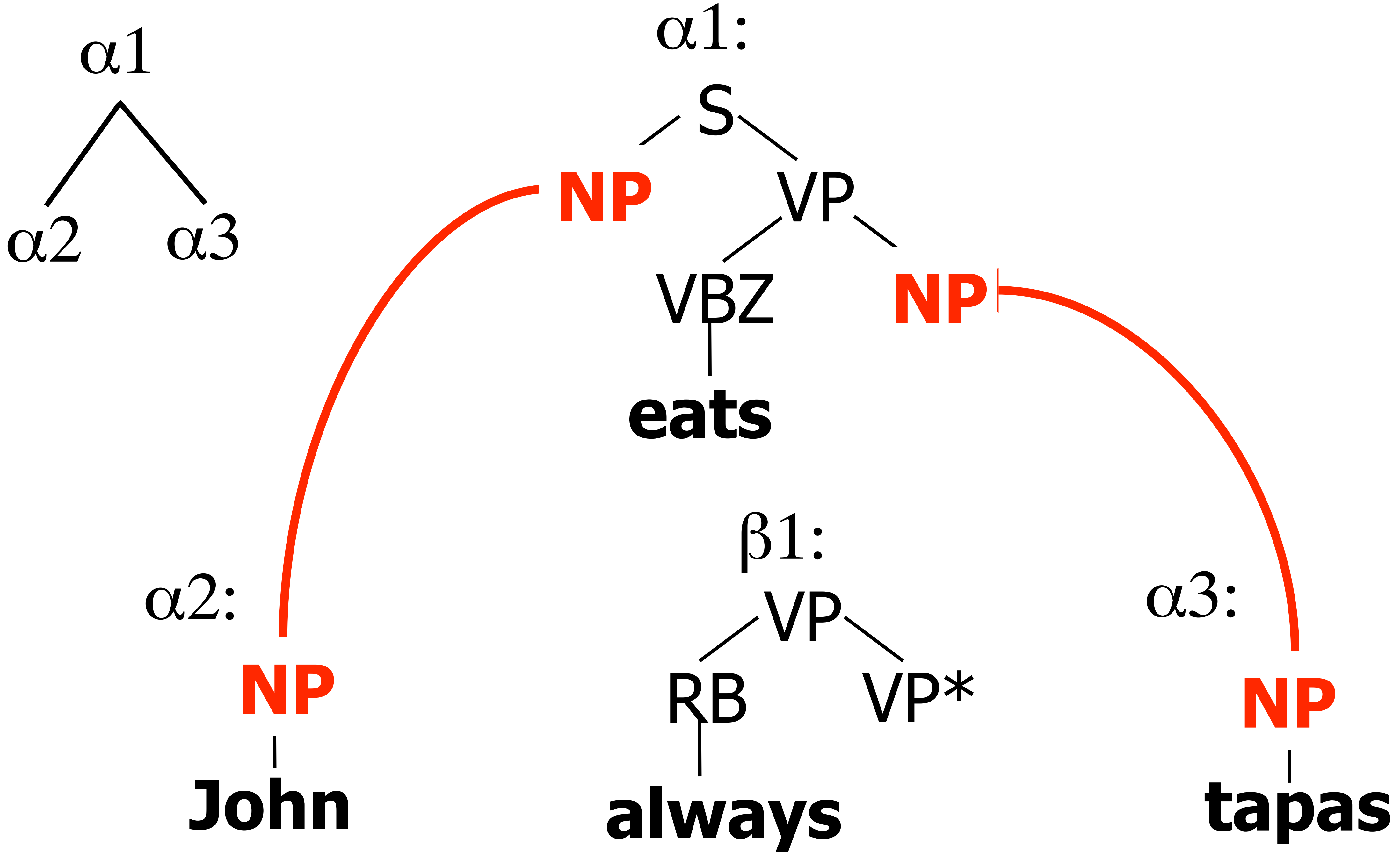
A TAG derivation



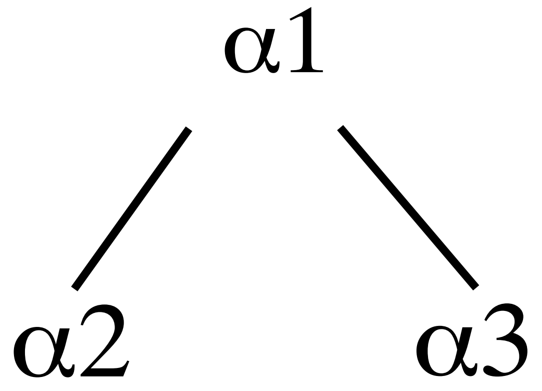
A TAG derivation

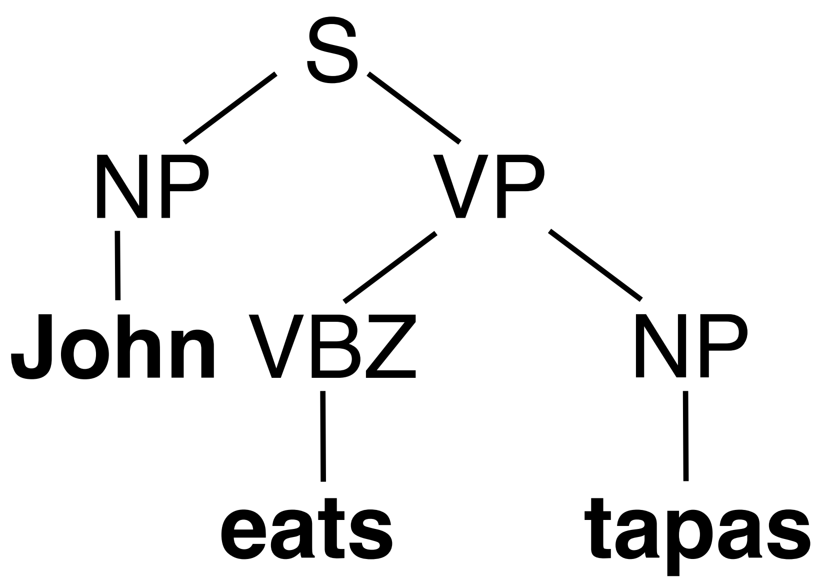


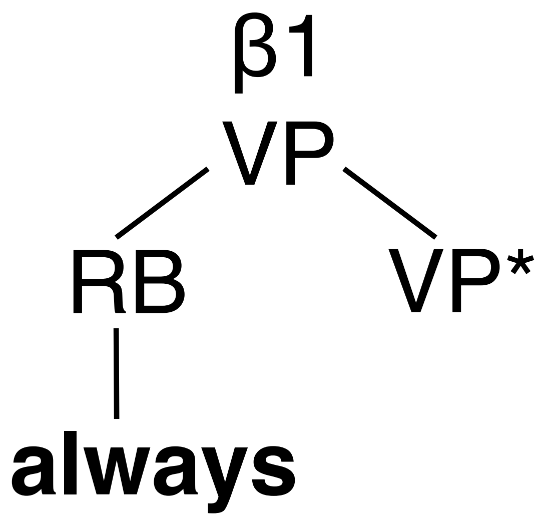
A TAG derivation



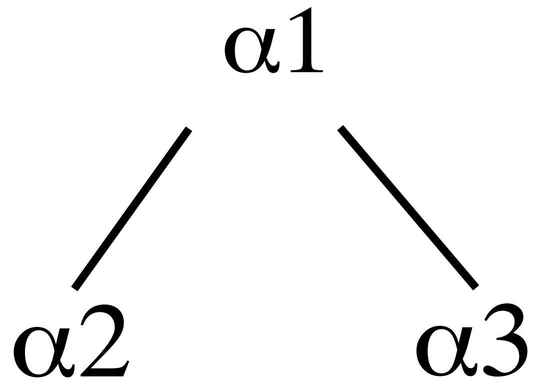
A TAG derivation

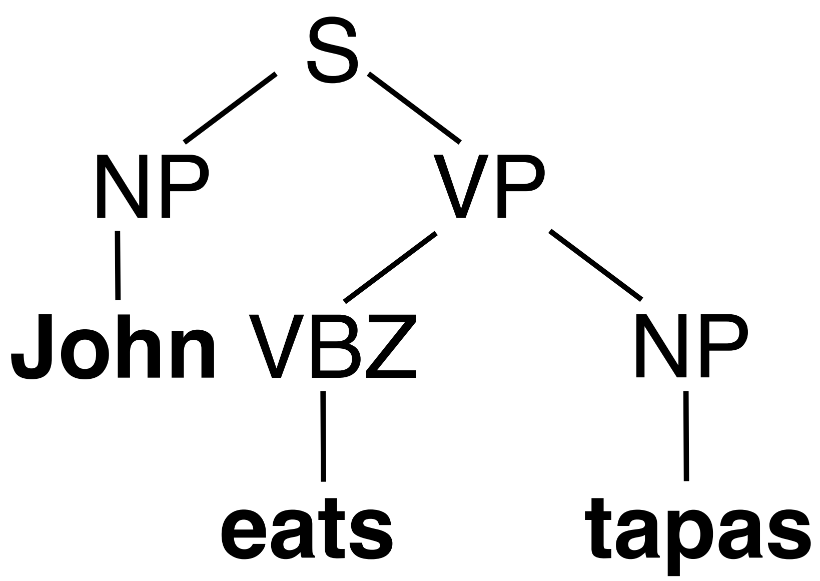


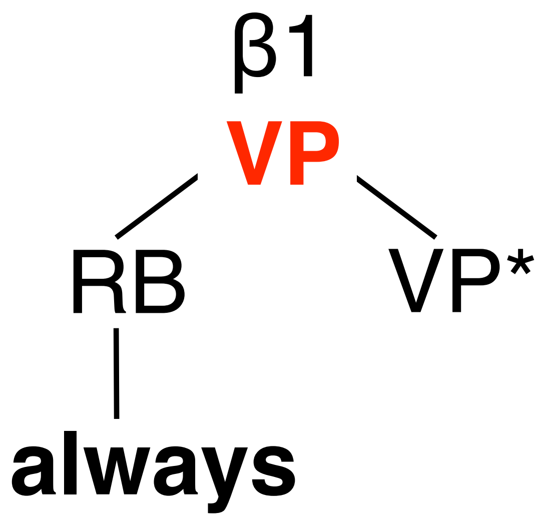




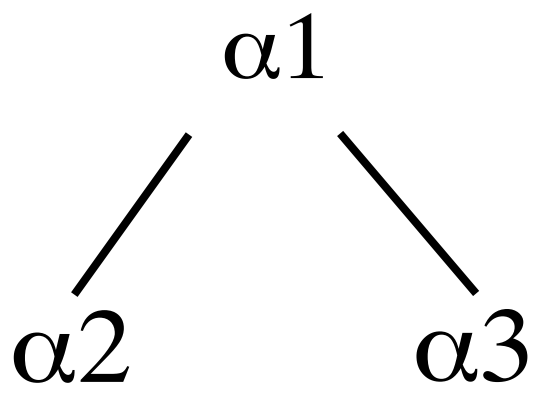
A TAG derivation

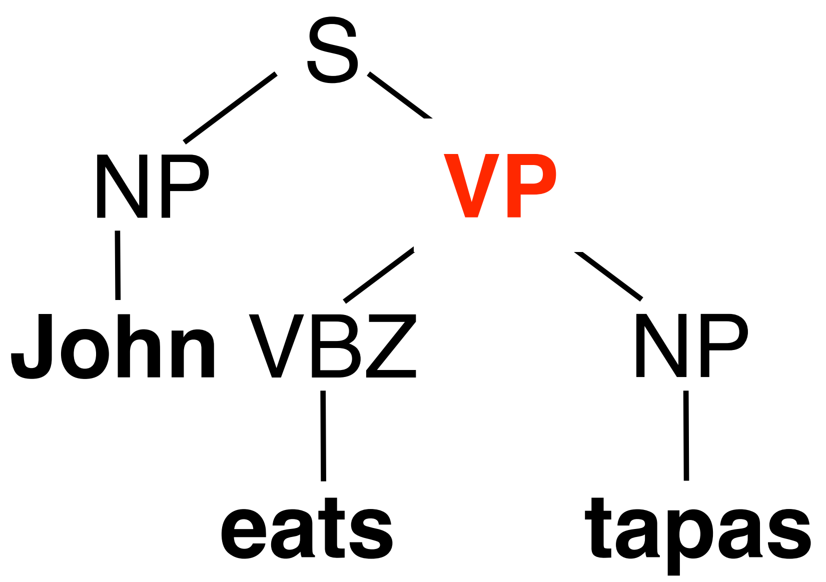


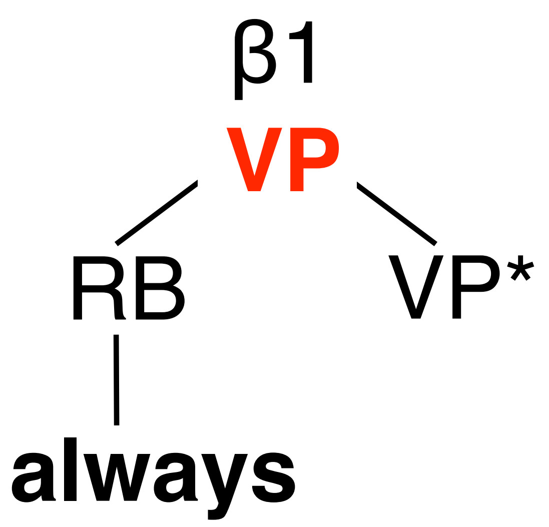




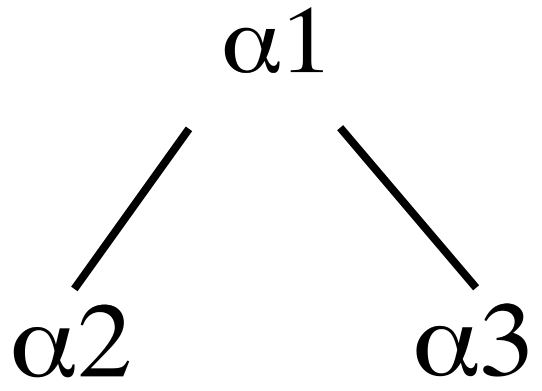
A TAG derivation

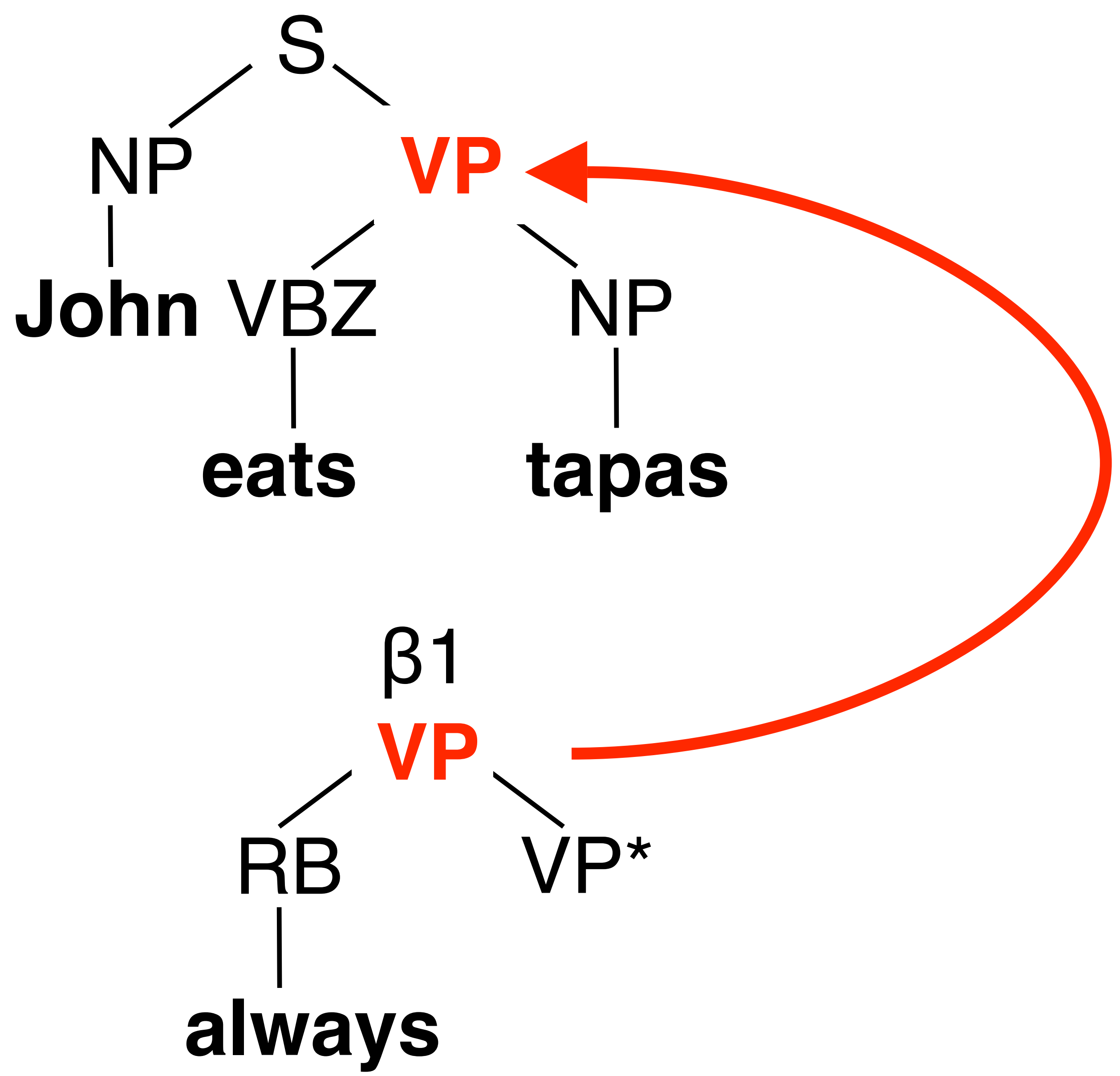




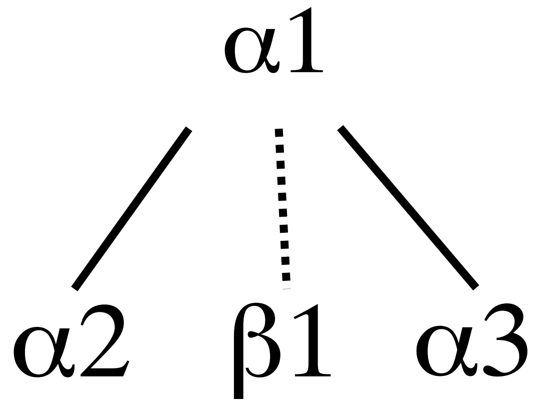


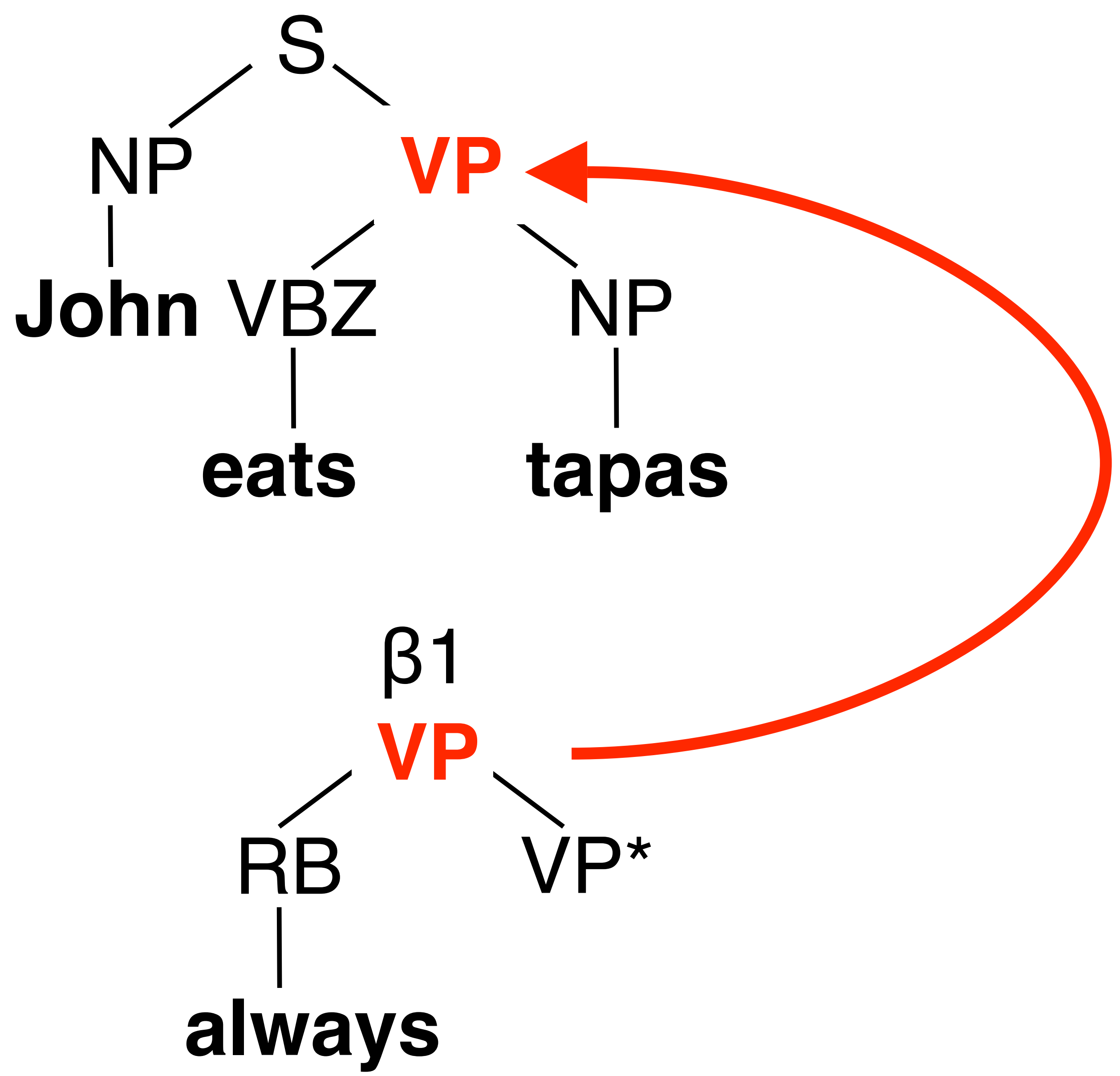
A TAG derivation



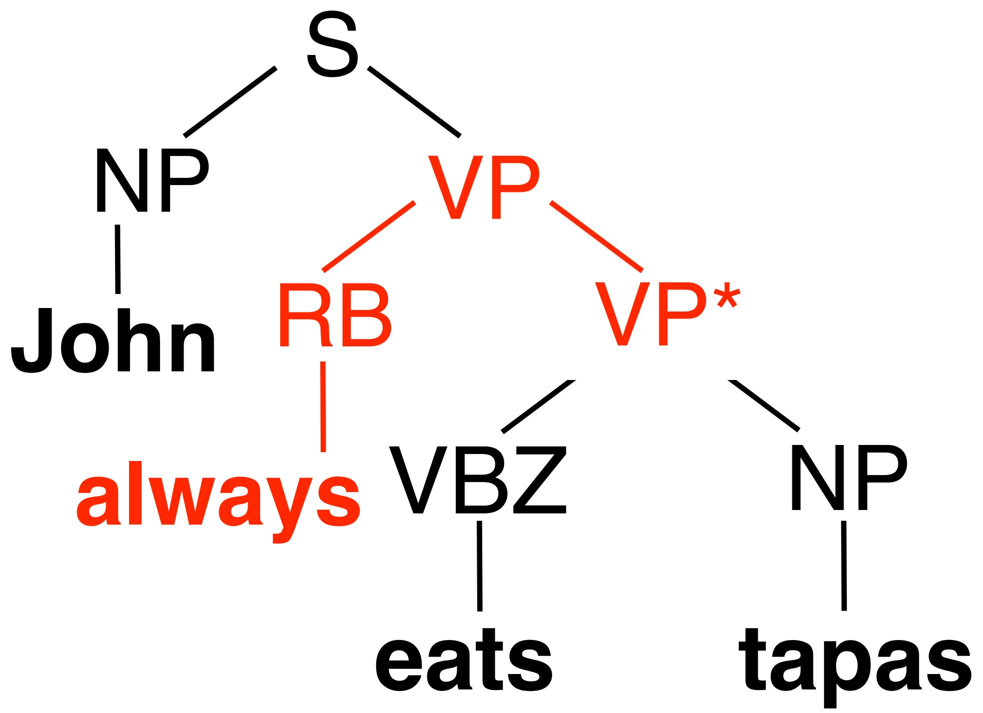


A TAG derivation

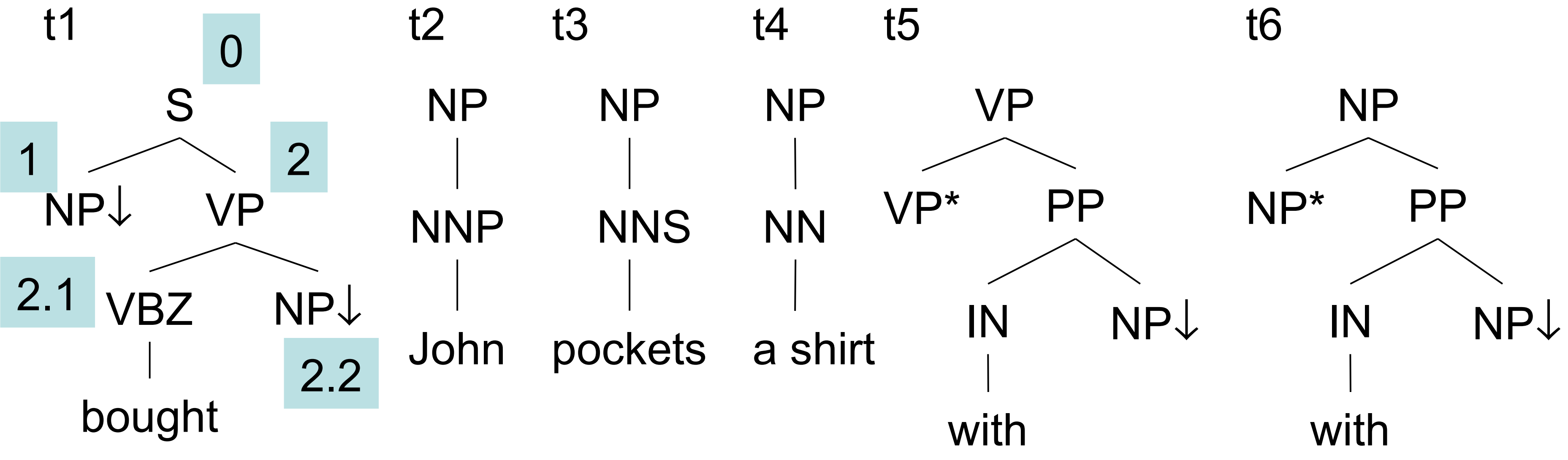




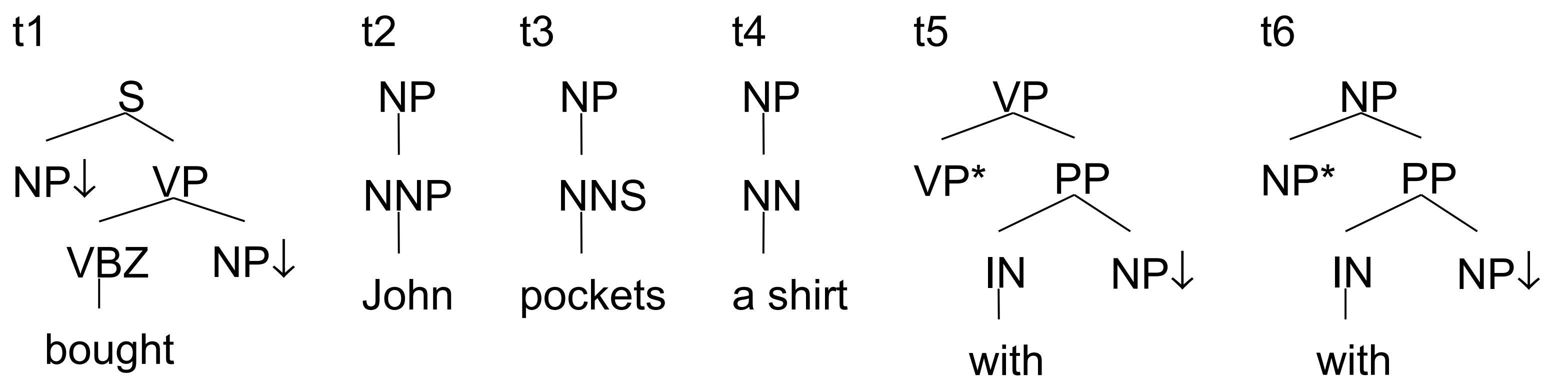
A TAG derivation

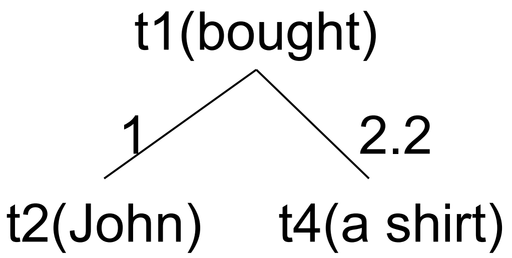


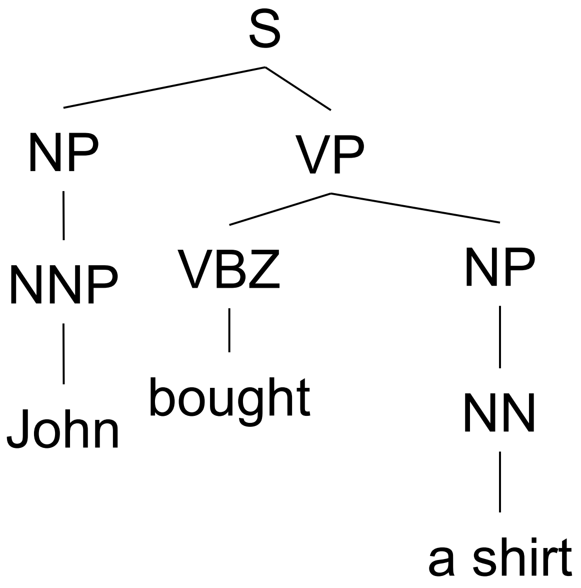
Lexicalized TAG: example



Gorn tree address: an index for each node in the tree

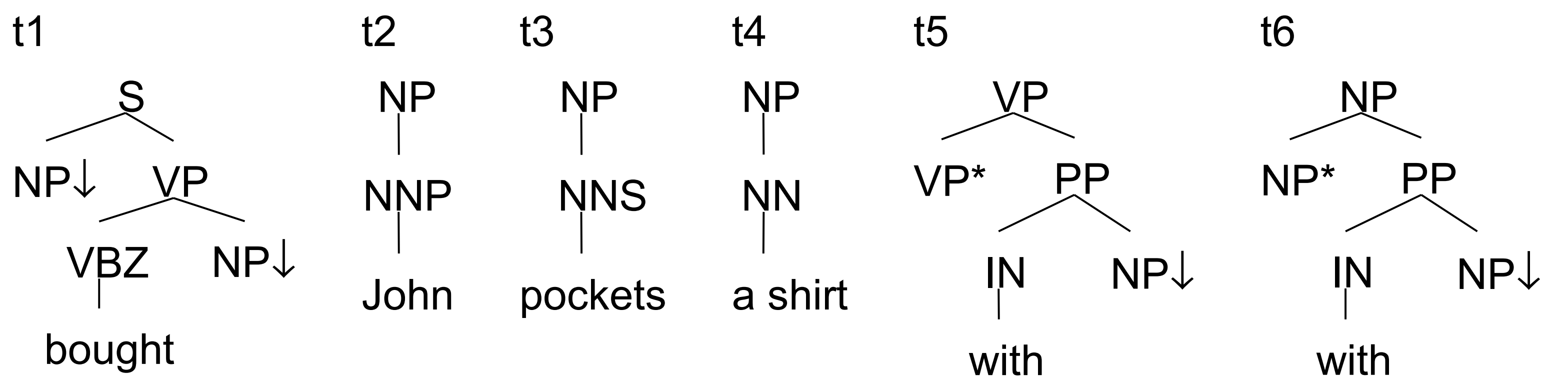


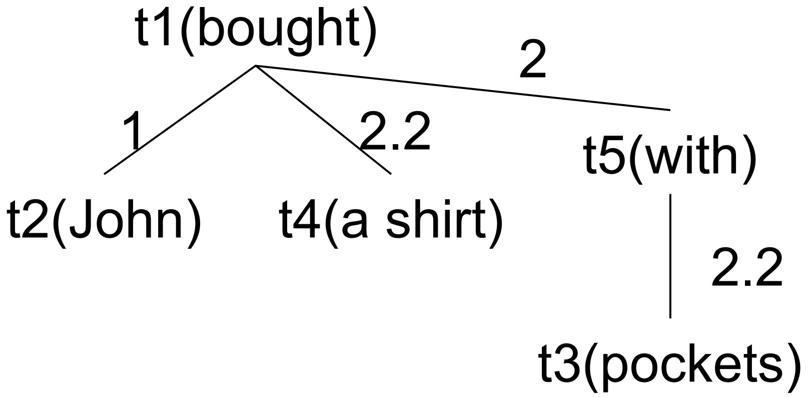




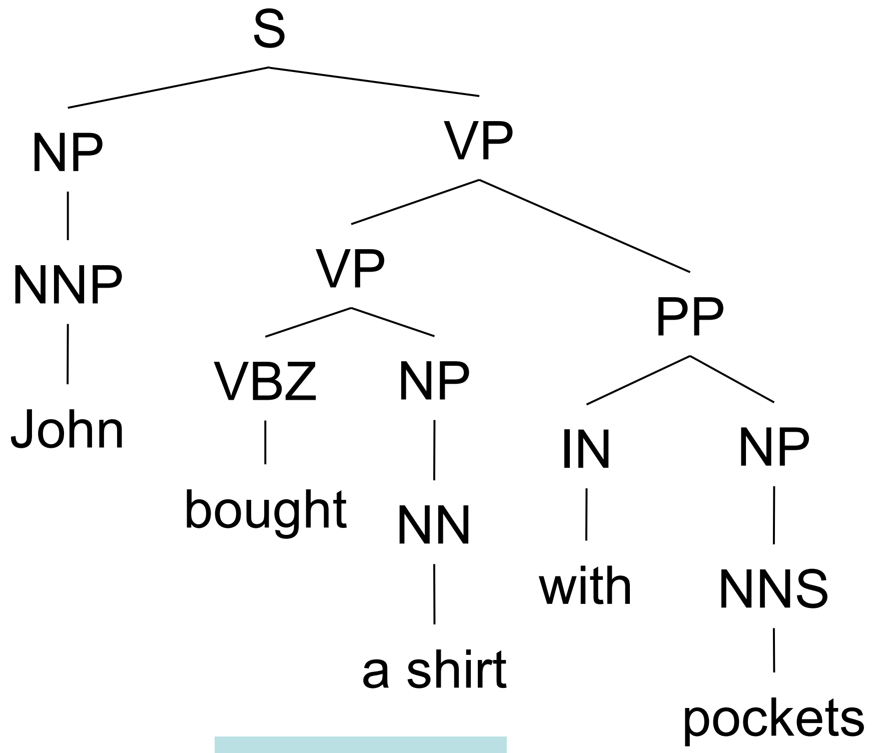
derivation tree

derived tree

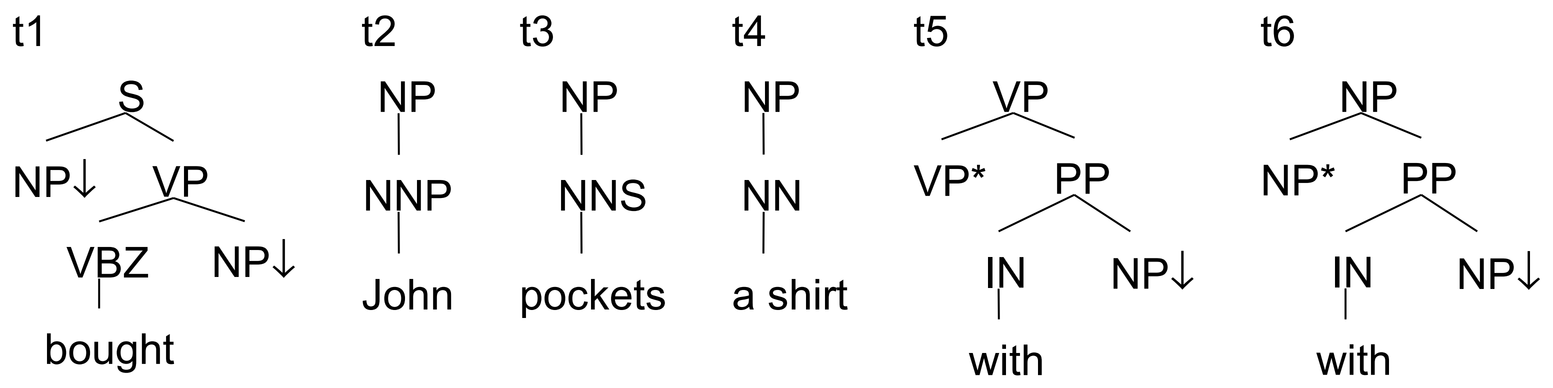


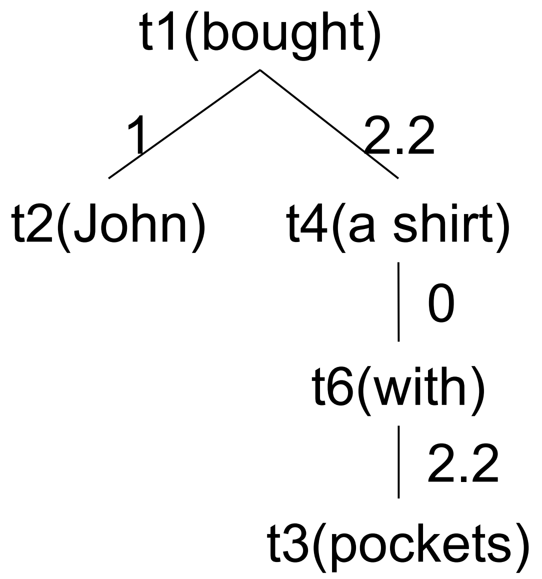


derivation tree

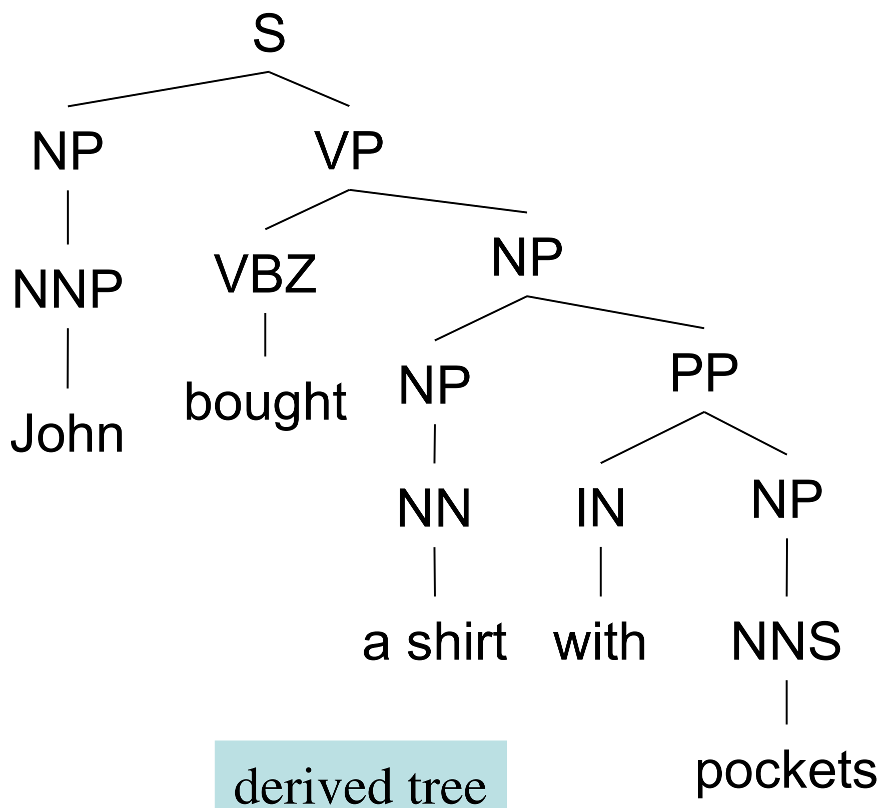


derived tree



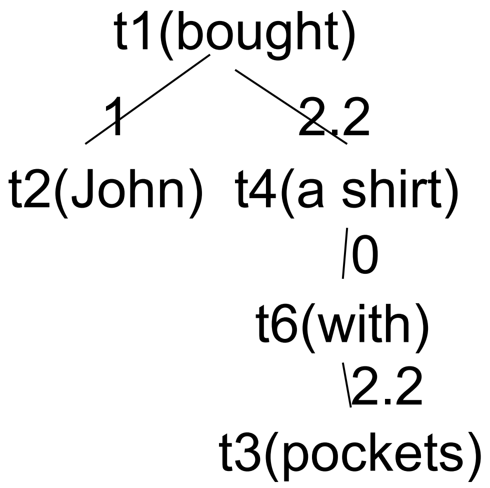


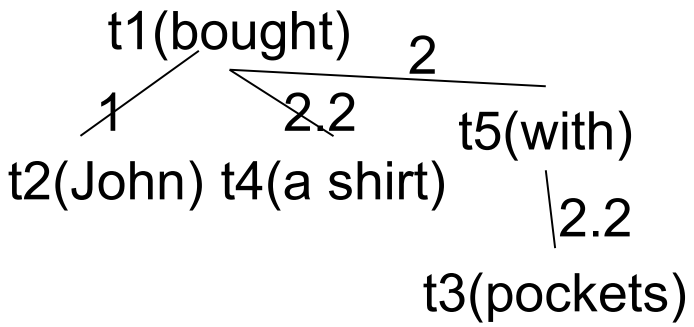
derivation tree



derived tree

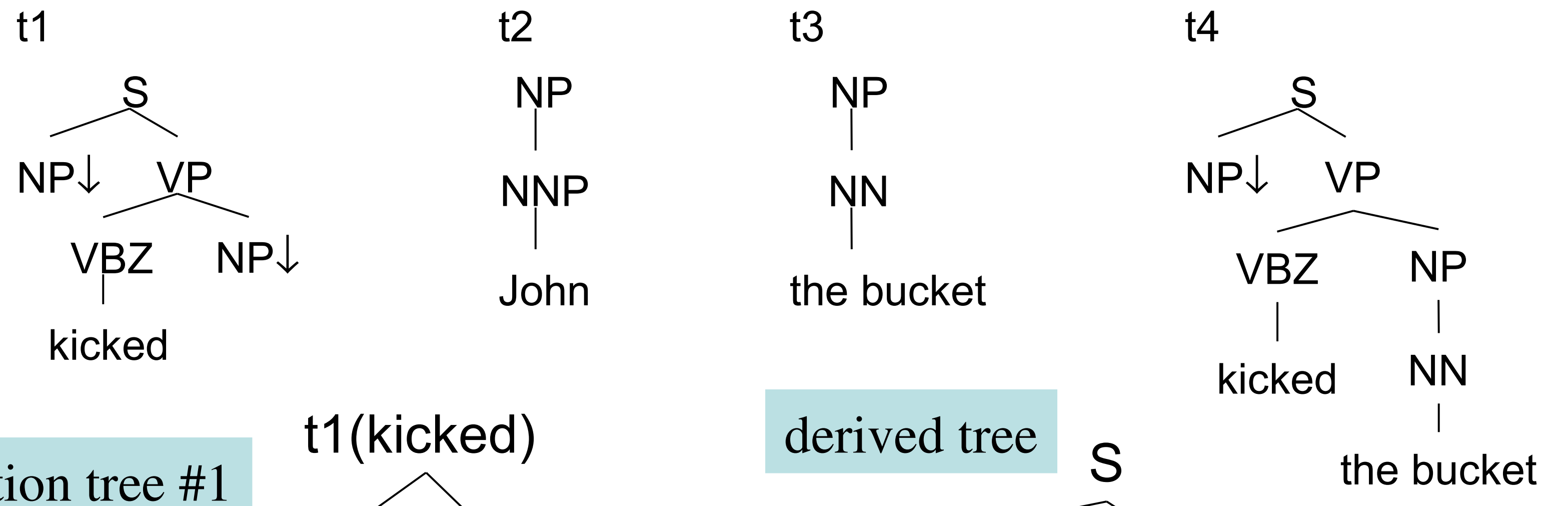
Ambiguity Resolution



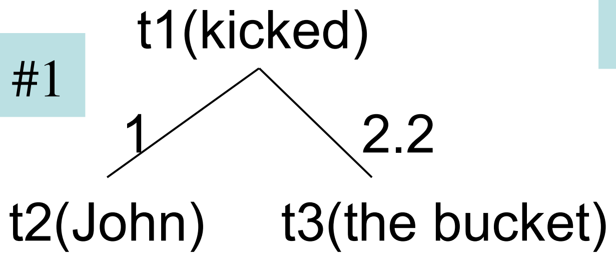


- Two possible derivations for ***John bought a shirt with pockets***
- One of them is more plausible than the other
- A pervasive problem in natural language
- Statistical parsing can be used for this kind of ambiguity resolution

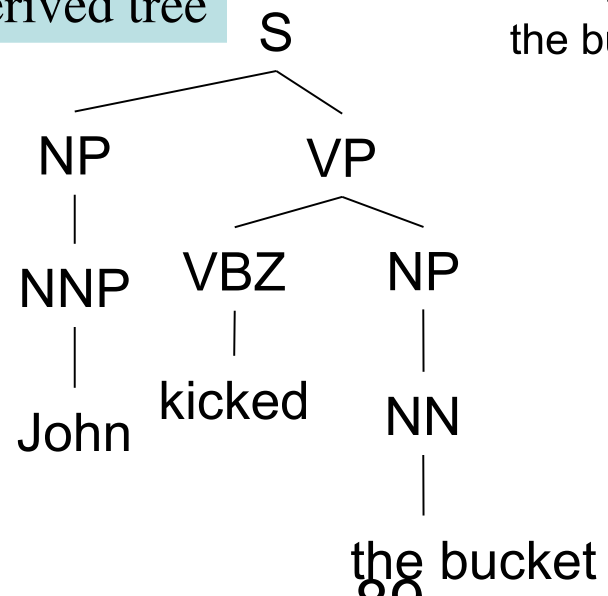
Idioms



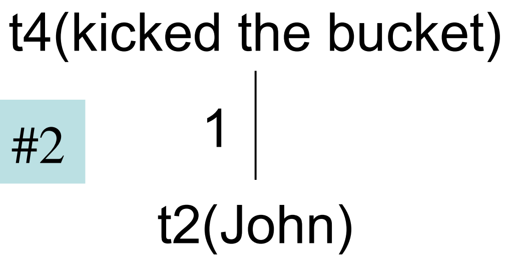
derivation tree #1



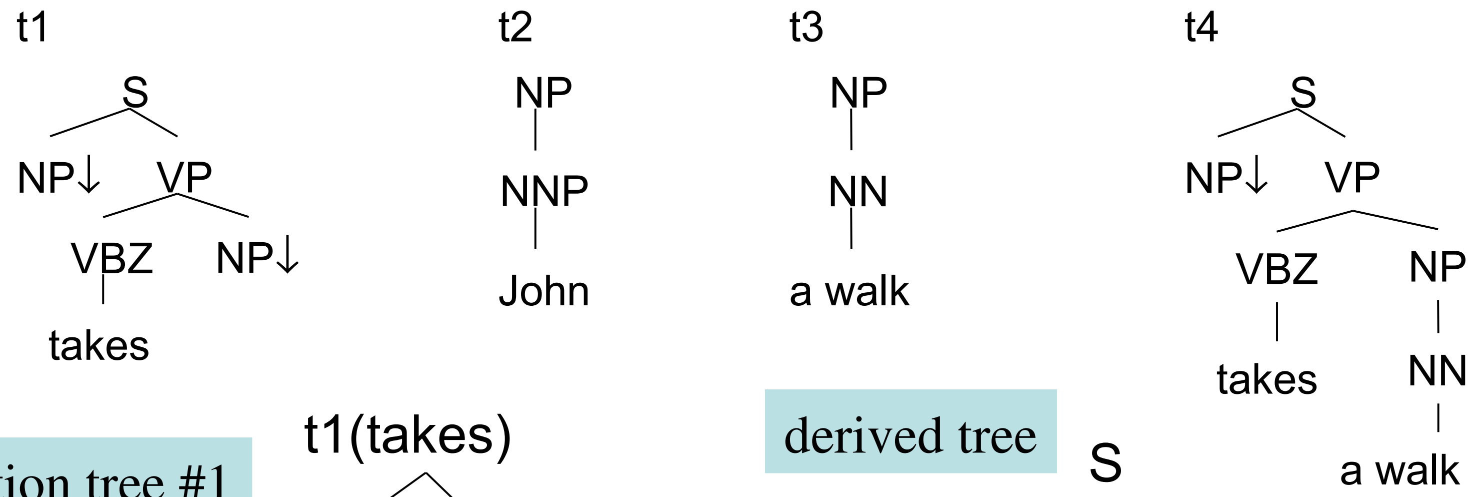
derived tree



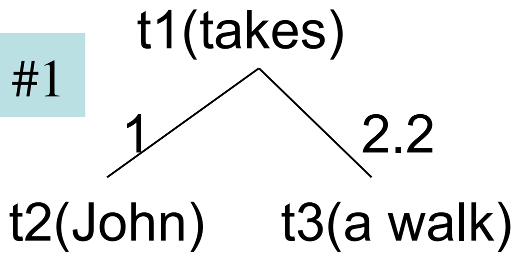
derivation tree #2



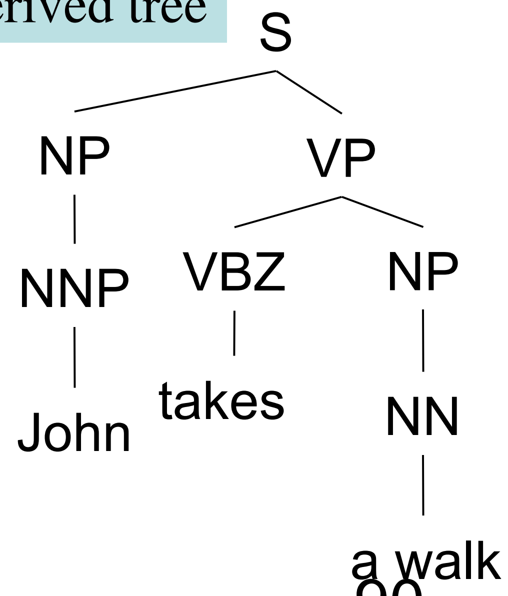
Phrasal/Light Verbs



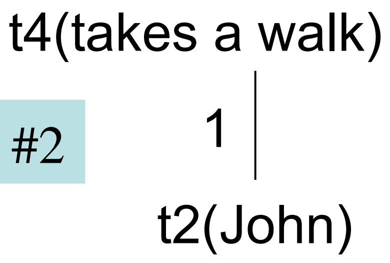
derivation tree #1



derived tree

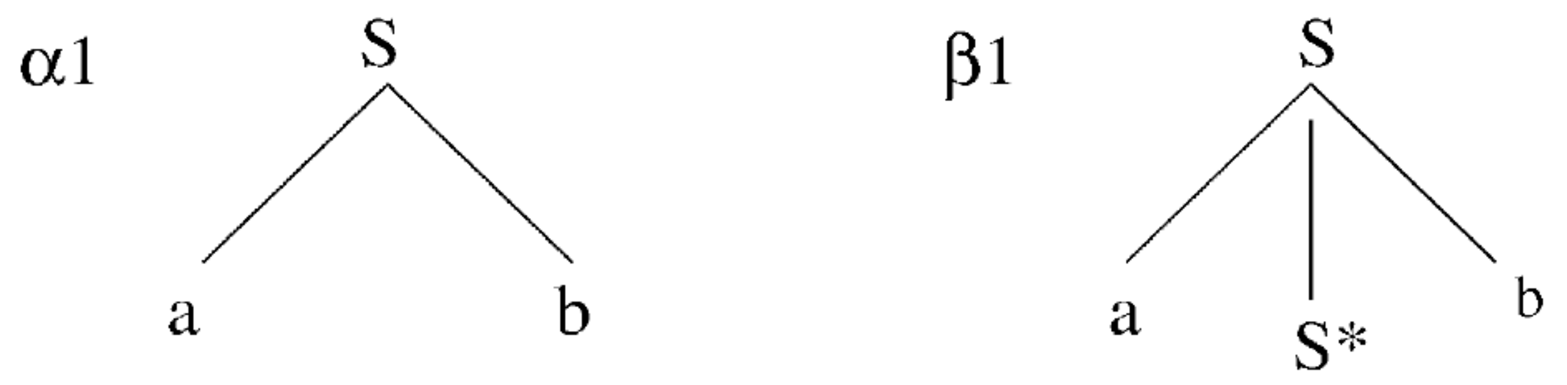


derivation tree #2

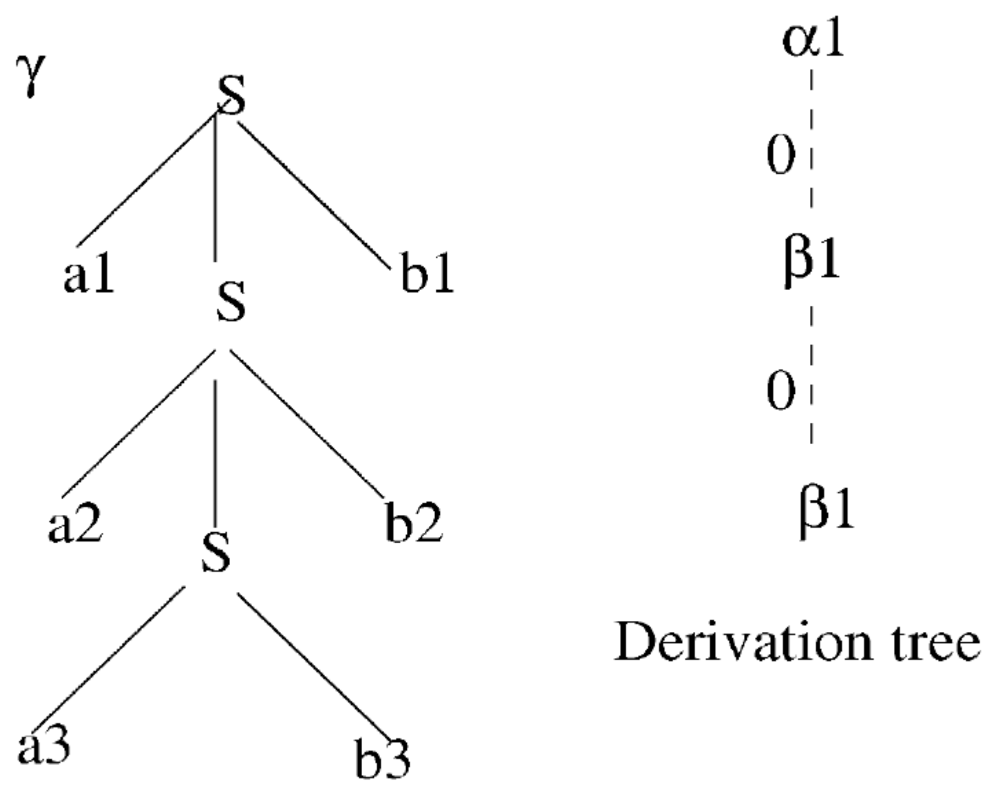


a^nb^n: Nested dependencies

Elementary trees

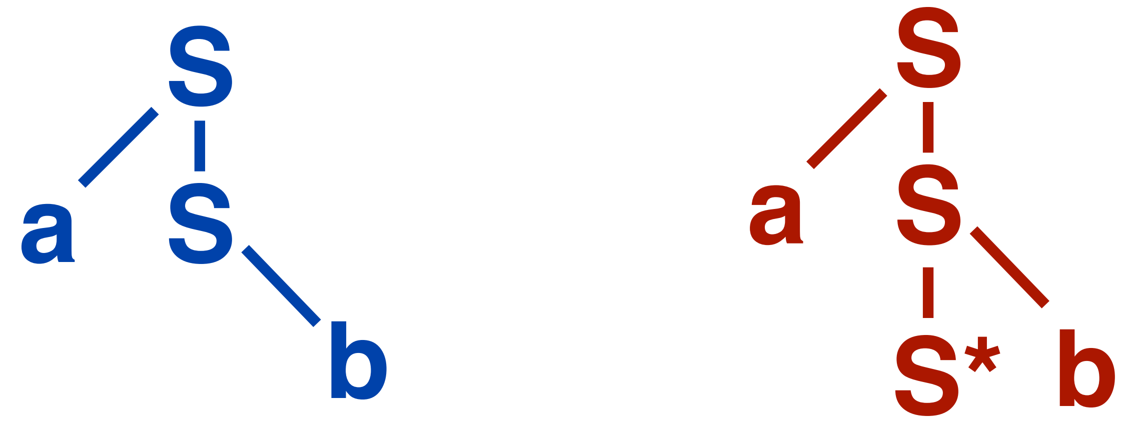


The derived tree and the derivation tree:

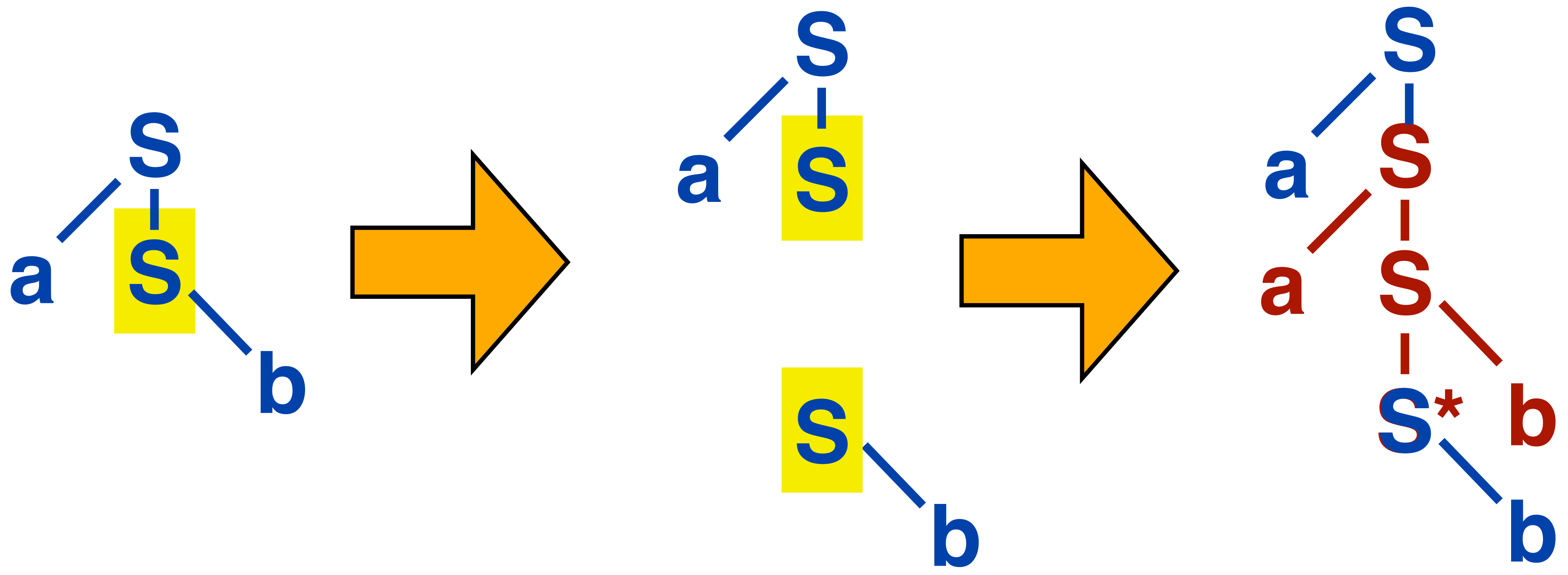


$a^n b^n$: Cross-serial dependencies

Elementary trees:



Deriving aabb



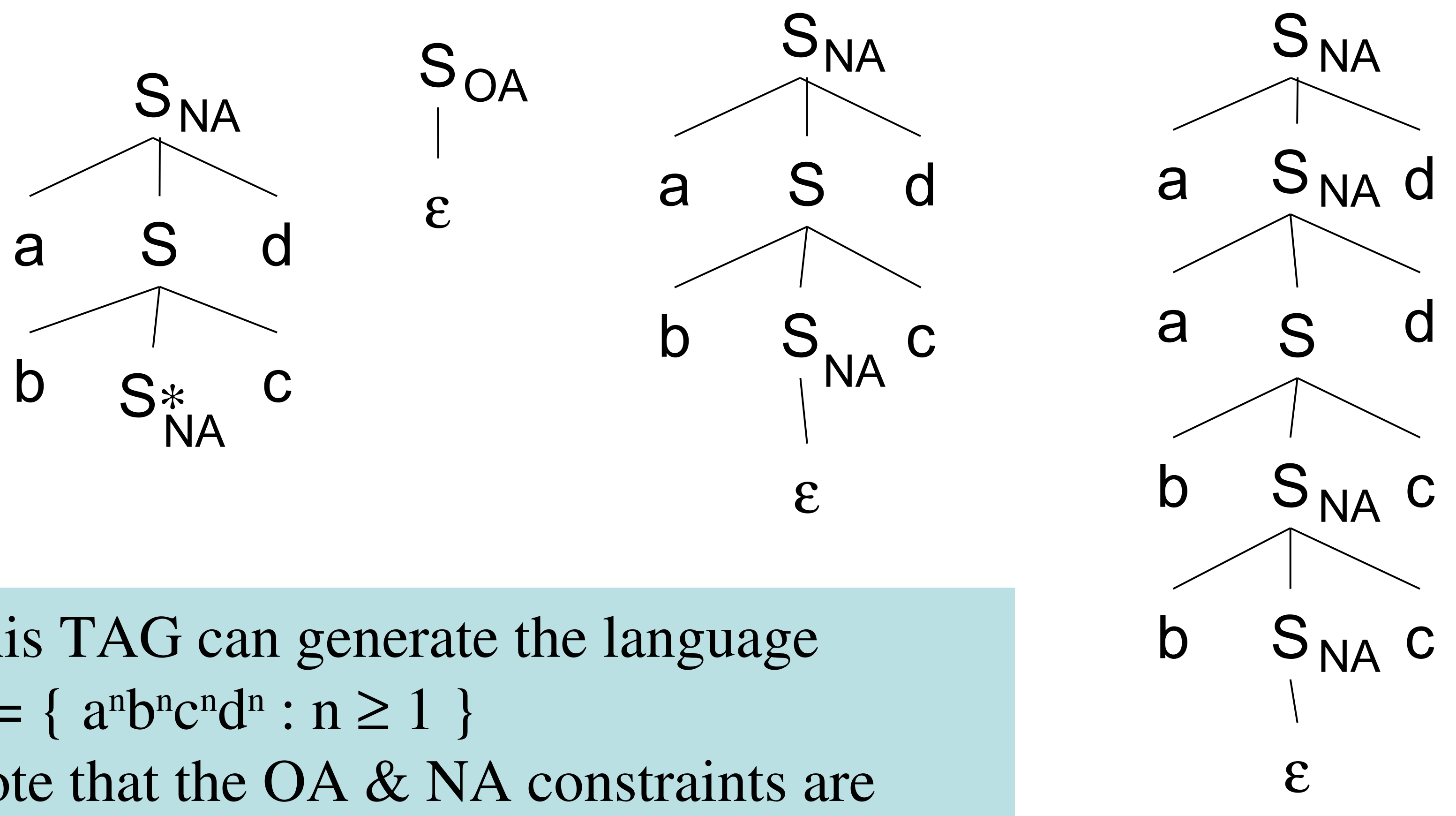
Tree-Adjoining Grammars

- A TAG G = (N, T, I, A, S) where
  - N is the set of non-terminal symbols
  - T is the set of terminal symbols
  - I is the set of initial or non-recursive trees built from N, T and domination predicates
  - A is the set of recursive trees: one leaf node is a non-terminal with same label as the root node
  - S is set of start trees (has to be initial)
  - I and A together are called *elementary trees*

Adjunction Constraints

- Adjunction is the rewriting of a non-terminal in a tree with an auxiliary tree
- We can think of this operation as being “context-free”
- Constraints are essential to control adjunction: both in practice for NLP and for formal closure properties
- Three types of constraints:
  - null adjunction (NA): no adjunction allowed at a node
  - obligatory adjunction (OA): adjunction must occur at a node
  - selective adjunction (SA): adjunction of a pre-specified set of trees can occur at a node

Adjunction Constraints

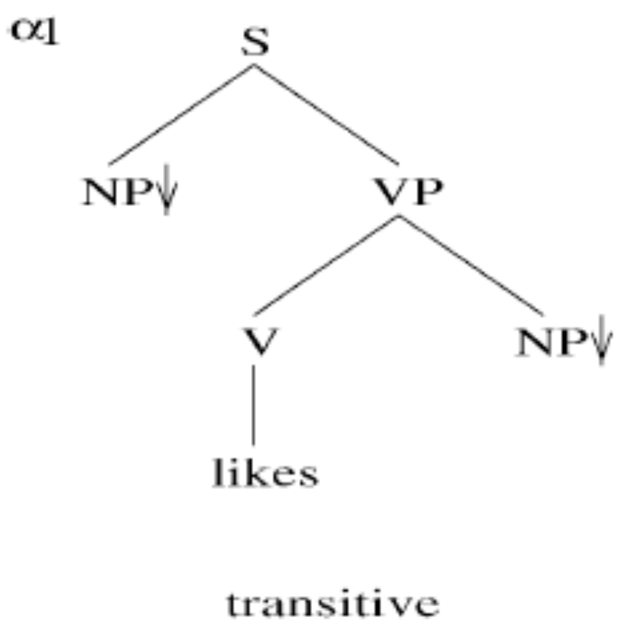


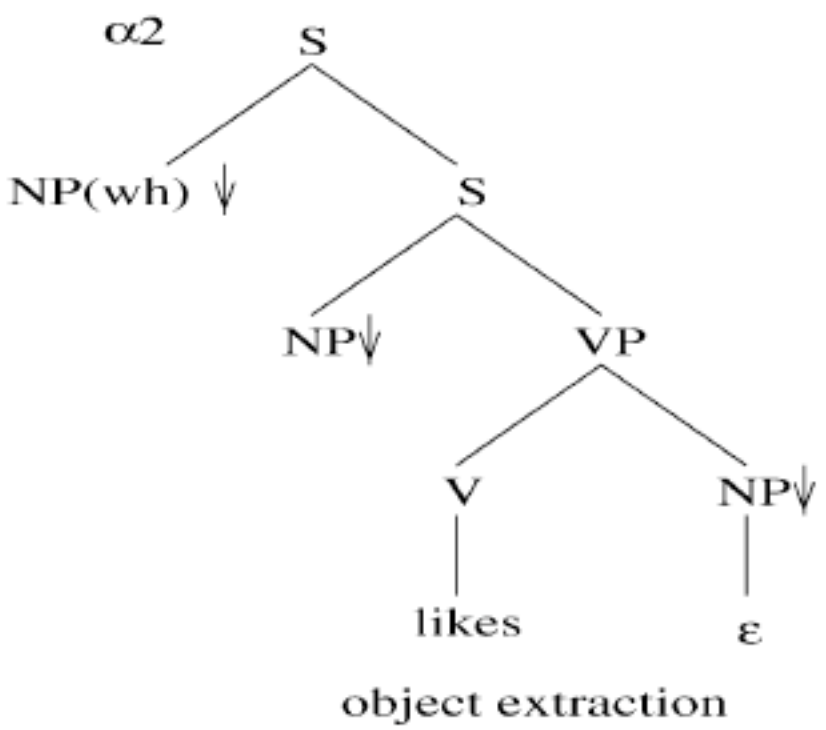
This TAG can generate the language
$L = \{ a^n b^n c^n d^n : n \geq 1 \}$
Note that the OA & NA constraints are crucial to obtain the correct language

Extraction in TAG

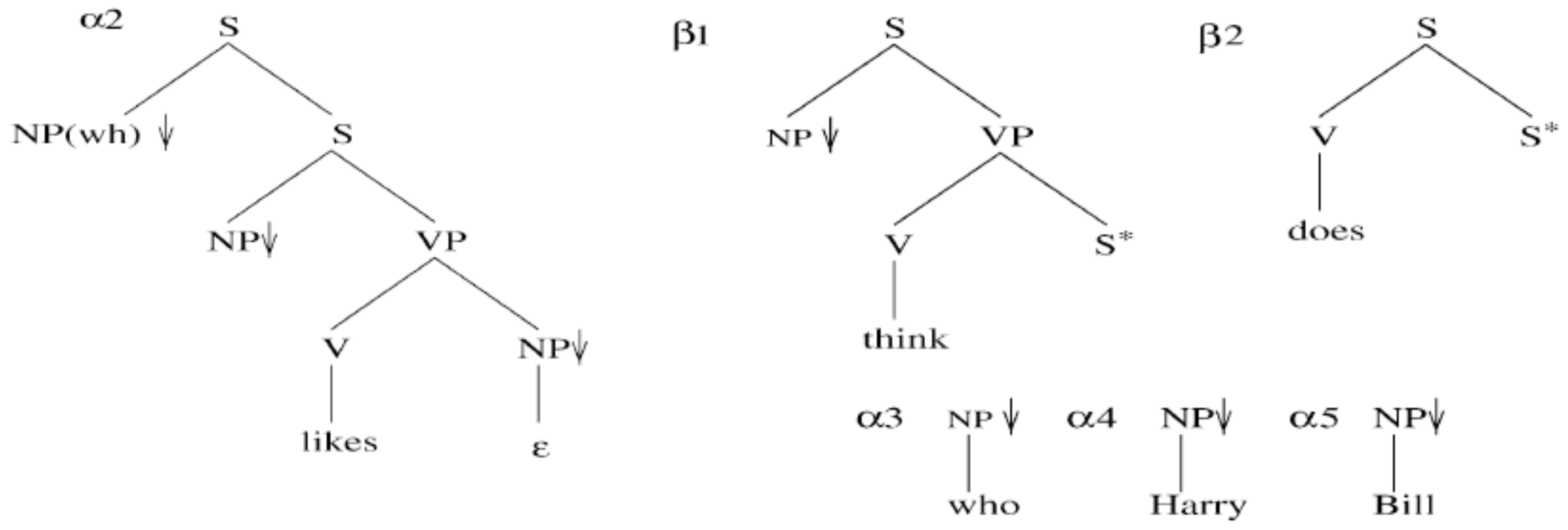
- **Extended domain of locality:**
  All arguments of a word are part of its lexical tree.
  - Different constructions require different trees:

John likes Mary. *Who does John like?*





Auxiliaries, obligatory adjunction



Who does Bill like?
Who does Harry think Bill likes?

'Who' **is an argument of** *'like(s)'.*
'does' **and** *'Harry think'* **adjoin into the** *'like(s)' tree.*

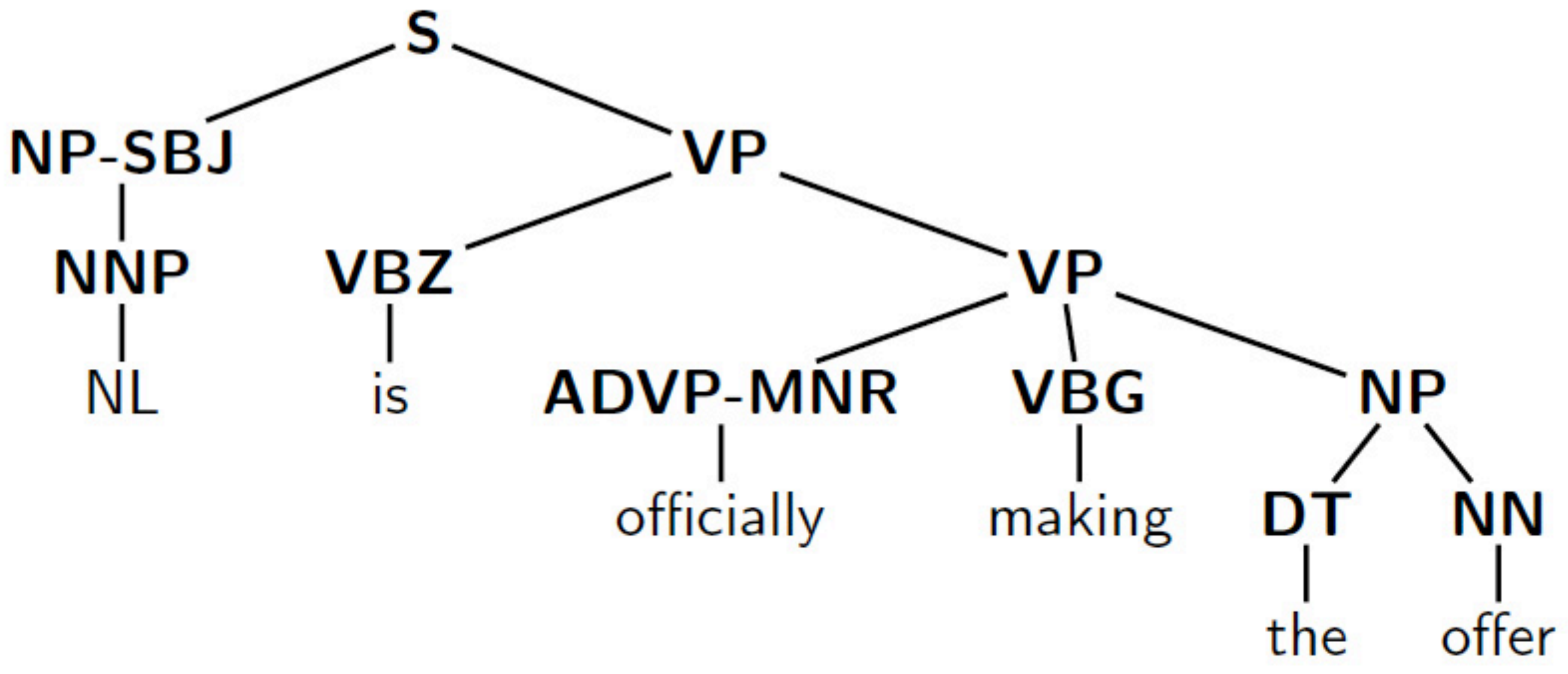
Extracting a TAG from the Penn Treebank

Extracting a TAG from the Treebank

- **Two different approaches:**
  - F. Xia. Automatic Grammar Generation From Two Different Perspectives. PhD thesis, University of Pennsylvania, 2001.
  - J. Chen, S. Bangalore, K. Vijay-Shanker. Automated Extraction of Tree-Adjoining Grammars from Treebanks, Natural Language Engineering (2005)
- **This lecture: just the basic ideas!**
- **The first attempt to extract an expressive grammar from the Treebank.**
- **Other approaches use similar methods, and yield similar results.**

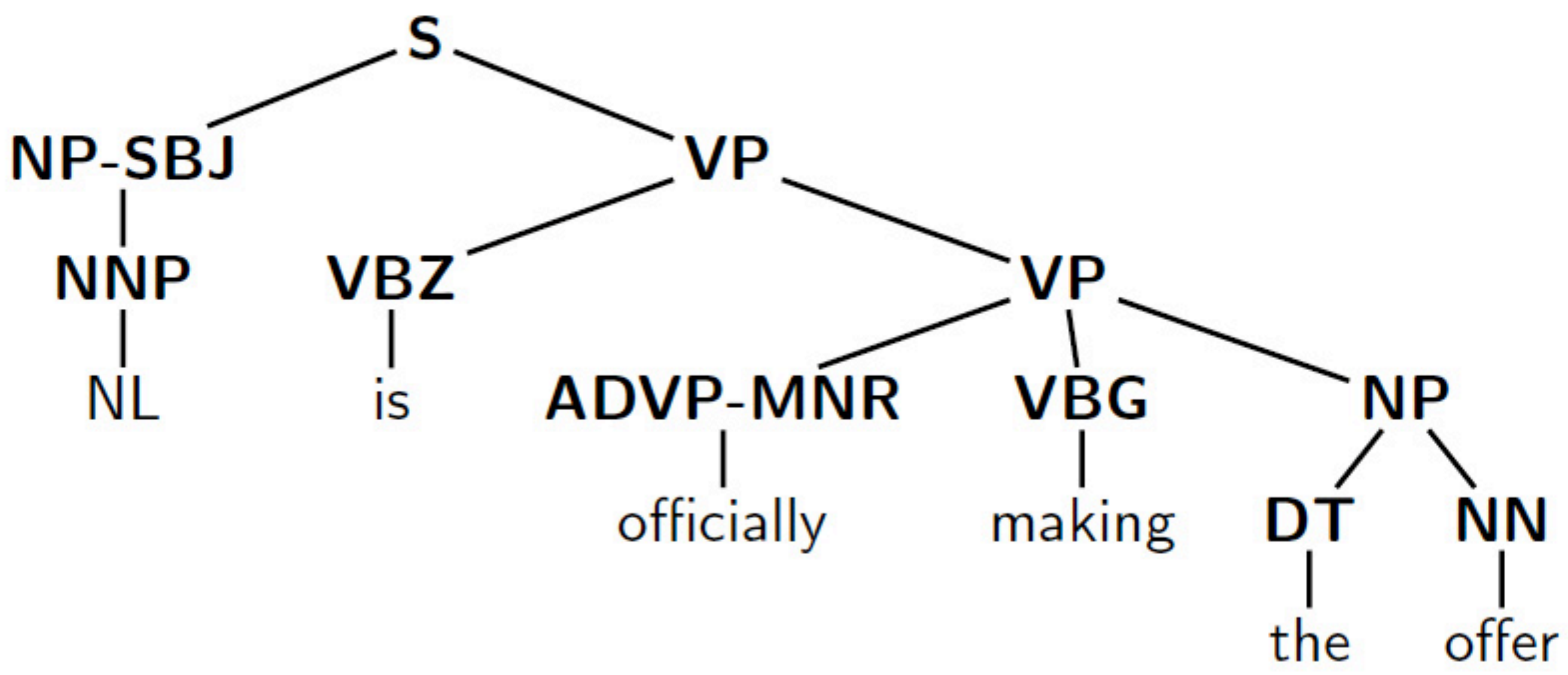
Extracting a TAG from the Penn Treebank

Input: a Treebank tree
(= the TAG derived tree)



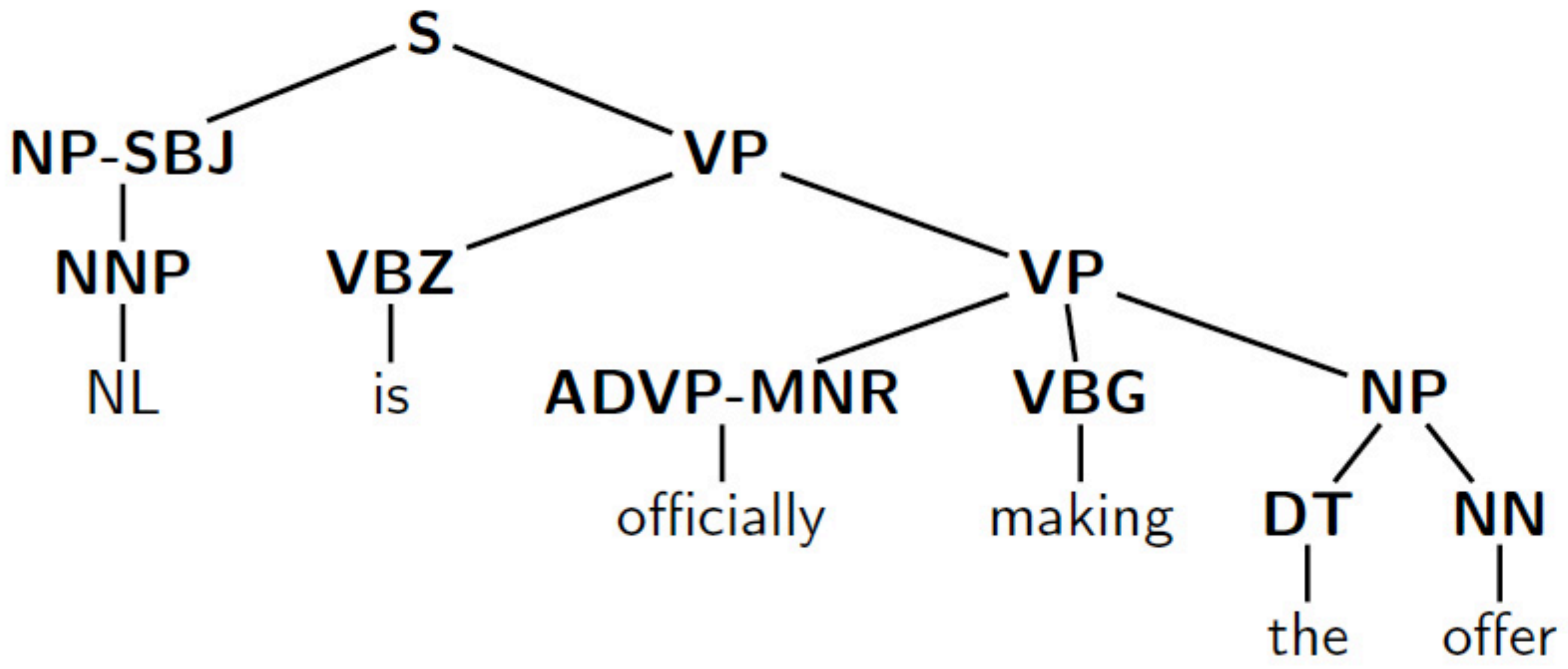
Output: a set of elementary trees
(= the TAG lexicon)

Extracting a TAG: the head



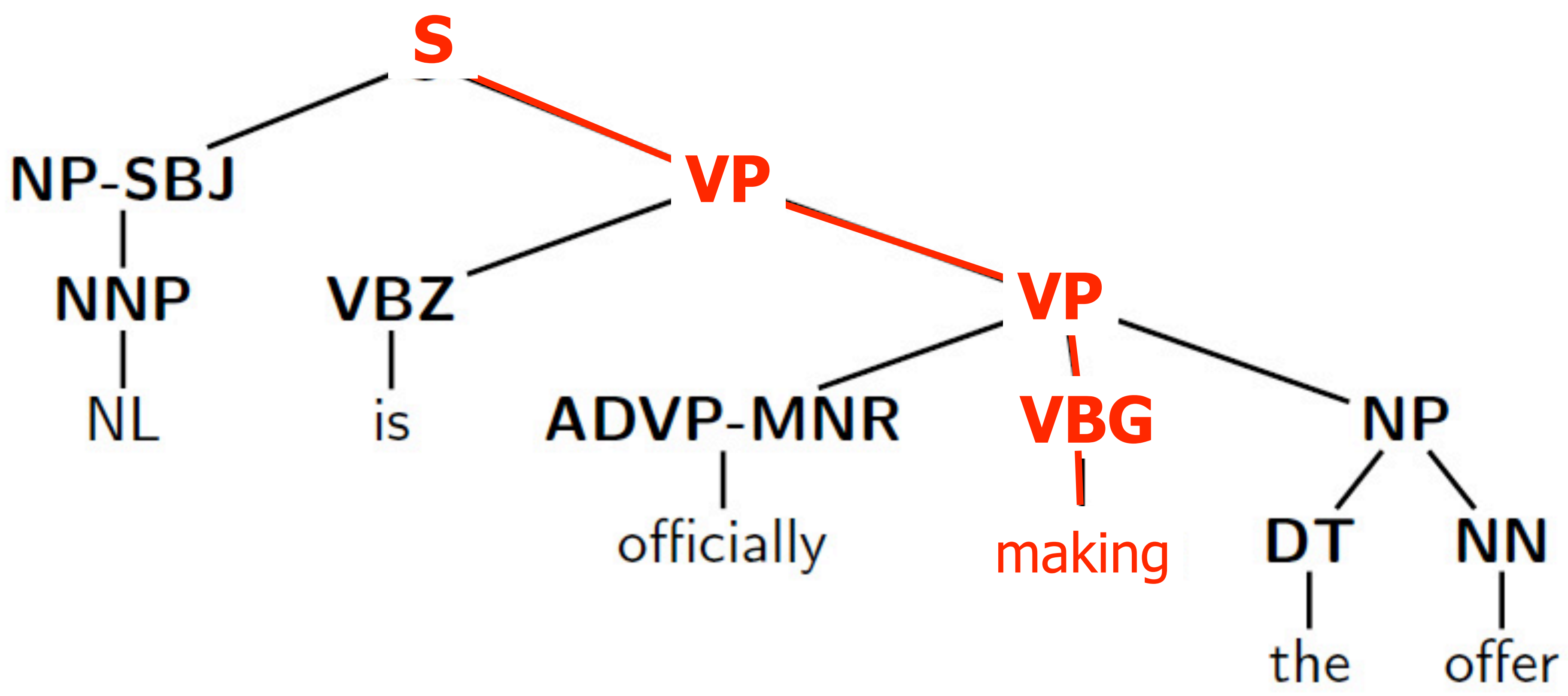
Extracting a TAG: the head

- **Identify the head path** (requires a head percolation table)



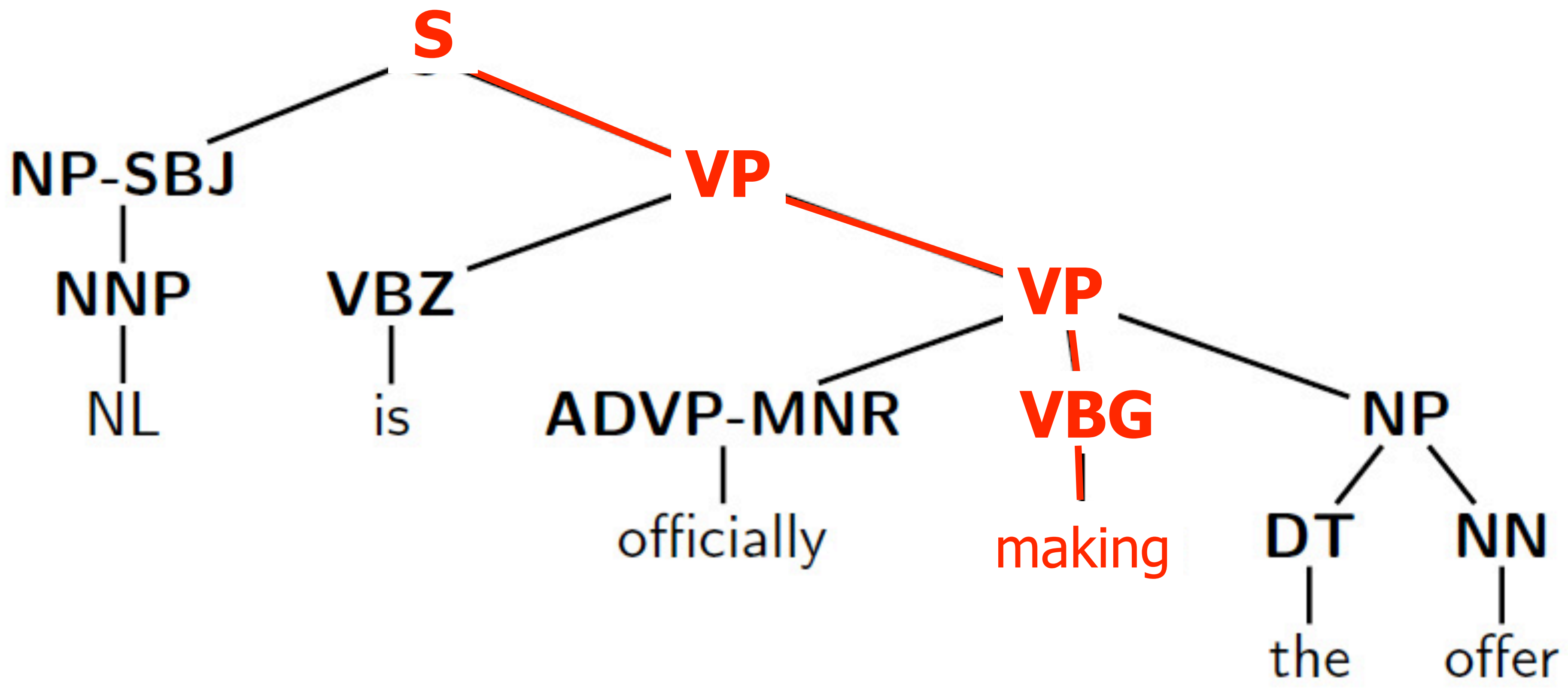
Extracting a TAG: the head

- **Identify the head path** (requires a head percolation table)



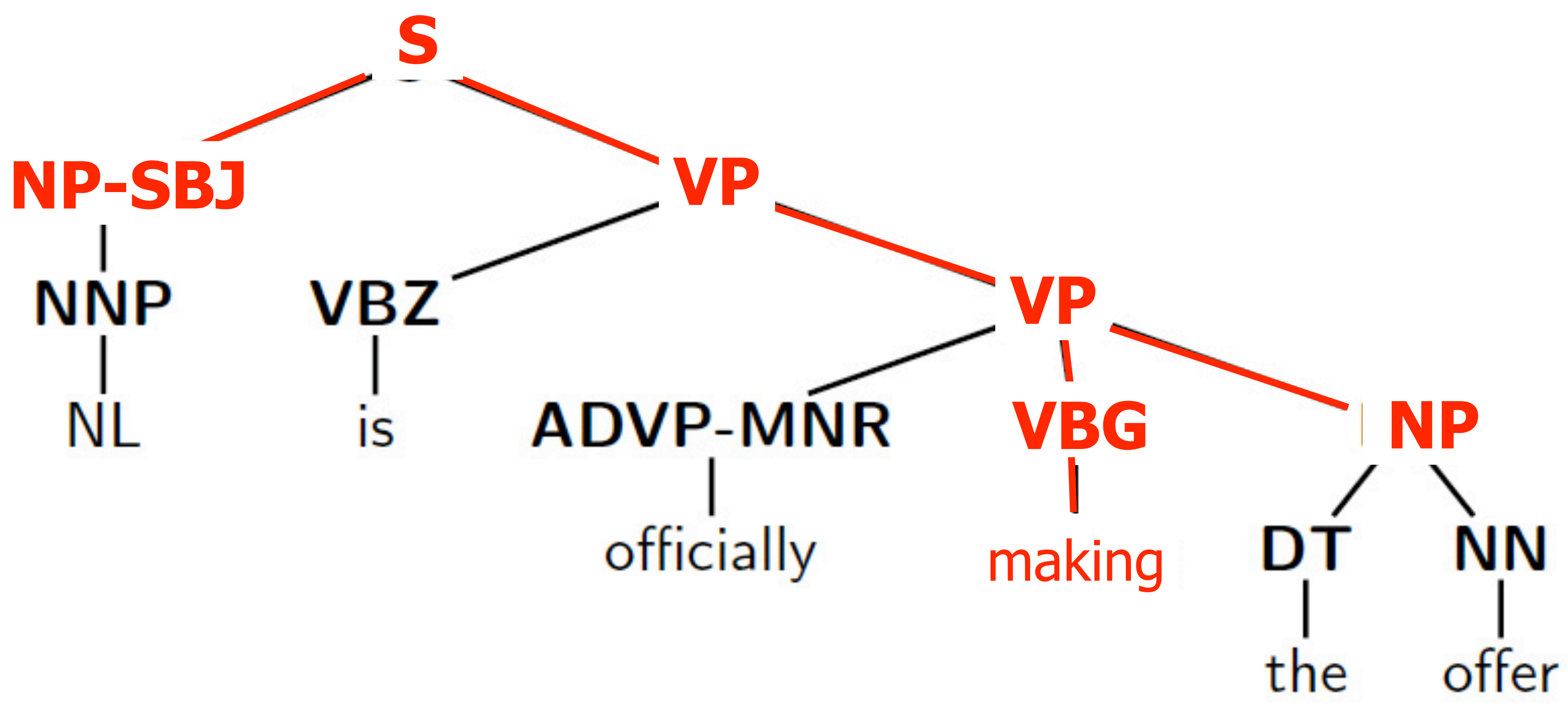
Extracting a TAG: the head

- **Identify the head path** (requires a head percolation table)
- **Find the arguments of the head** (requires an argument table)



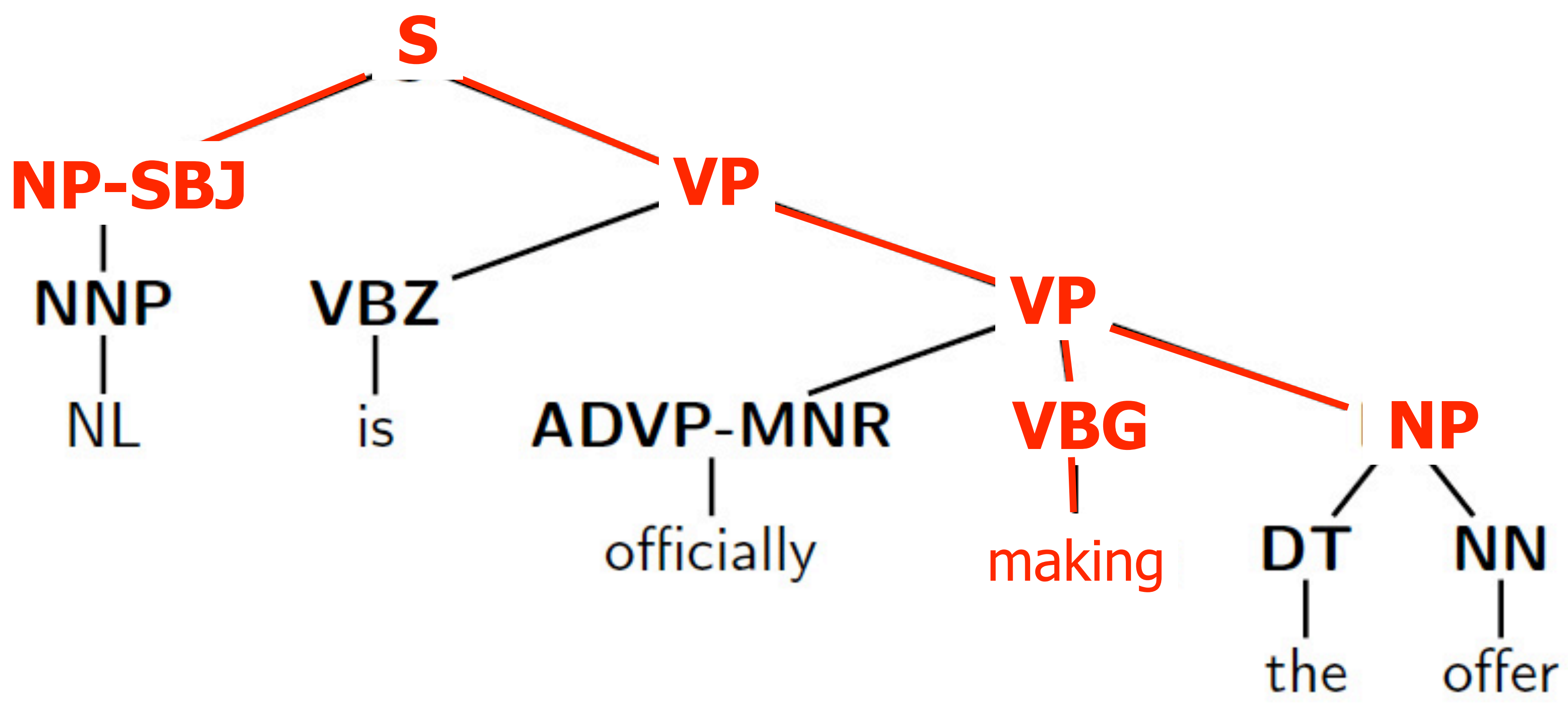
Extracting a TAG: the head

- **Identify the head path** (requires a head percolation table)
- **Find the arguments of the head** (requires an argument table)



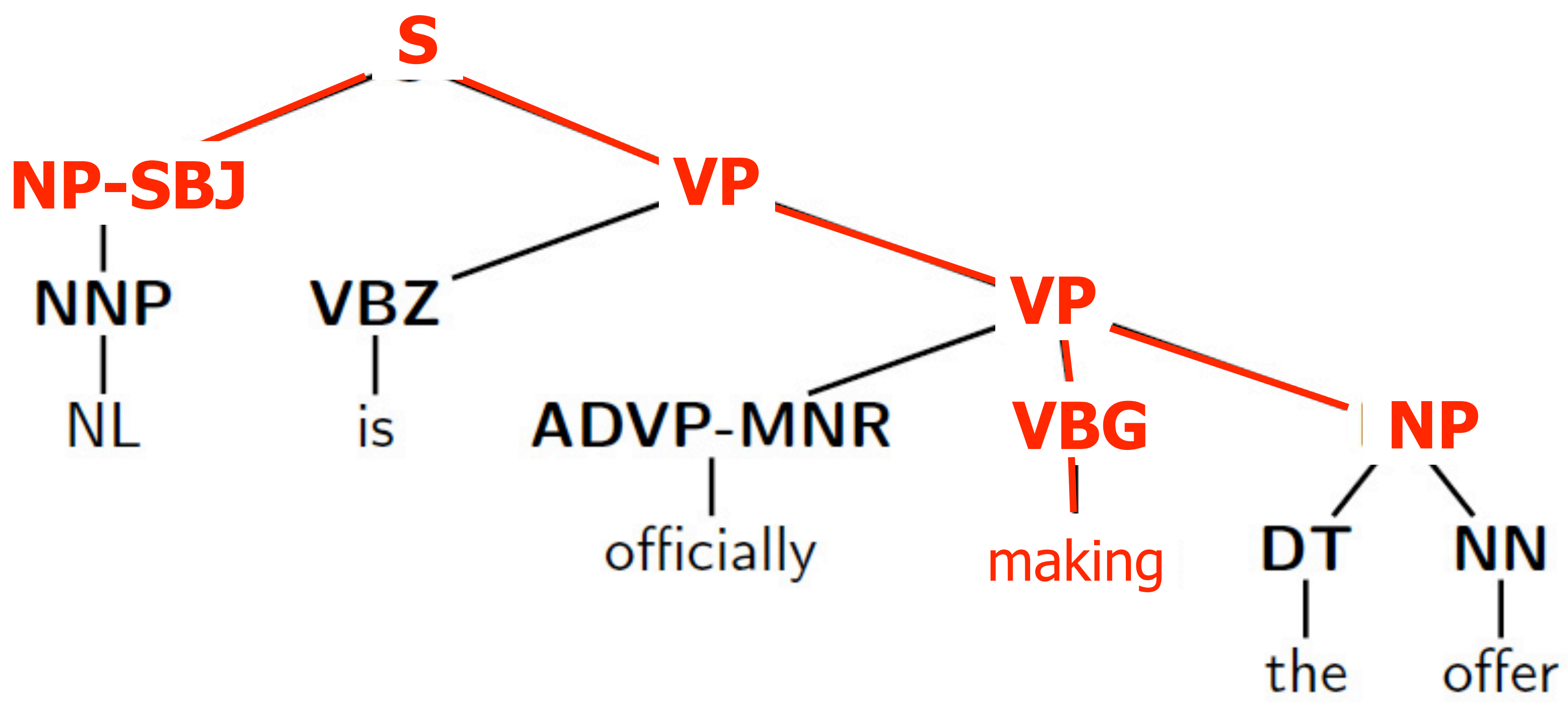
Extracting a TAG: the head

- **Identify the head path** (requires a head percolation table)
- **Find the arguments of the head** (requires an argument table)
- **Ignore modifiers** (requires an adjunct table)



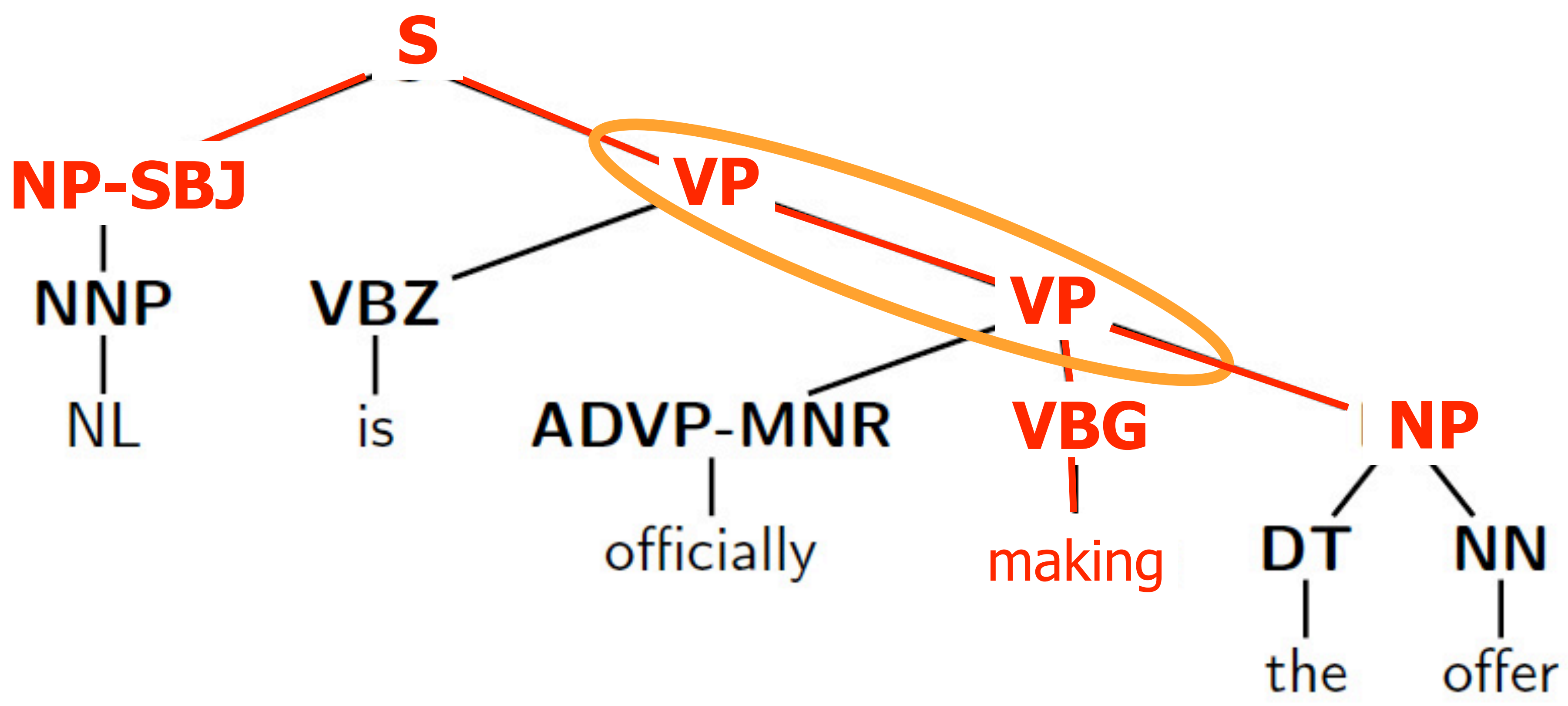
Extracting a TAG: the head

- **Identify the head path** (requires a head percolation table)
- **Find the arguments of the head** (requires an argument table)
- **Ignore modifiers** (requires an adjunct table)
- **Merge unary productions** (VP -> VP)



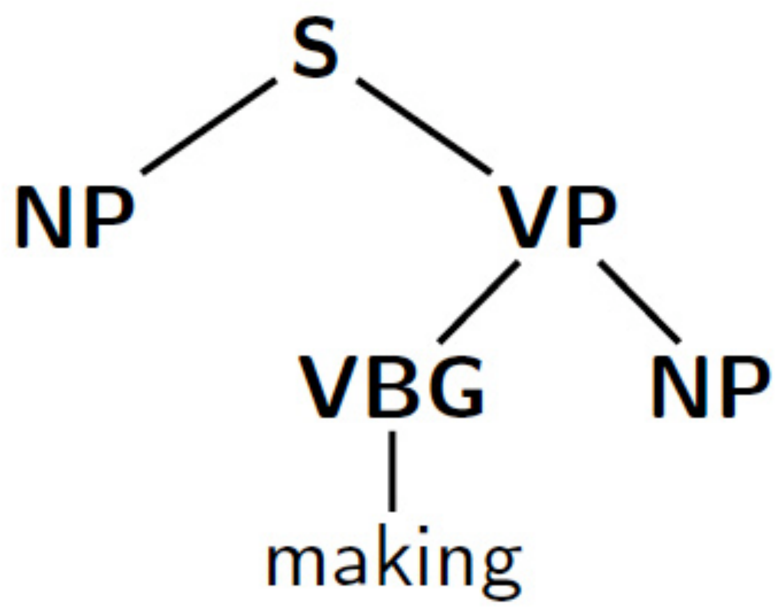
Extracting a TAG: the head

- **Identify the head path** (requires a head percolation table)
- **Find the arguments of the head** (requires an argument table)
- **Ignore modifiers** (requires an adjunct table)
- **Merge unary productions** (VP -> VP)



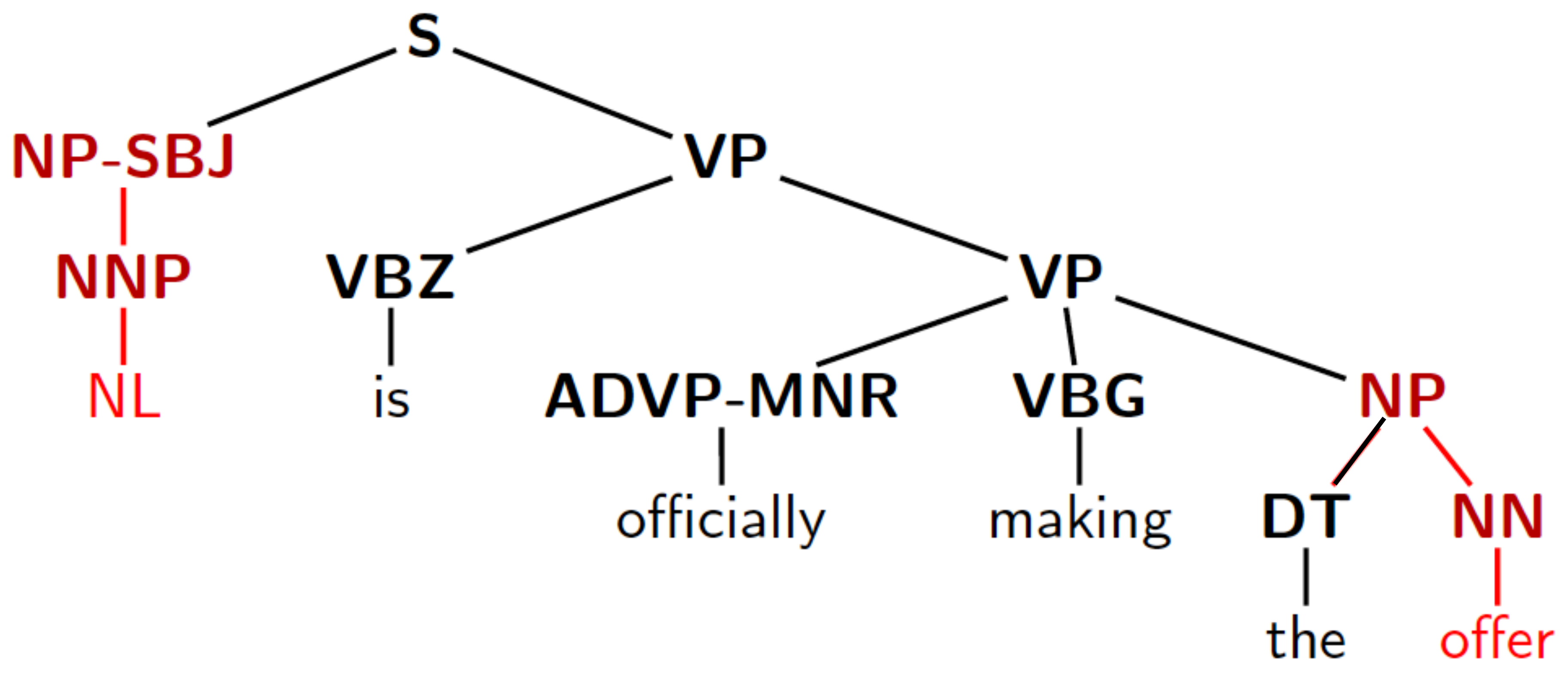
Extracting a TAG: the head

- **This is the elementary tree for the head:**



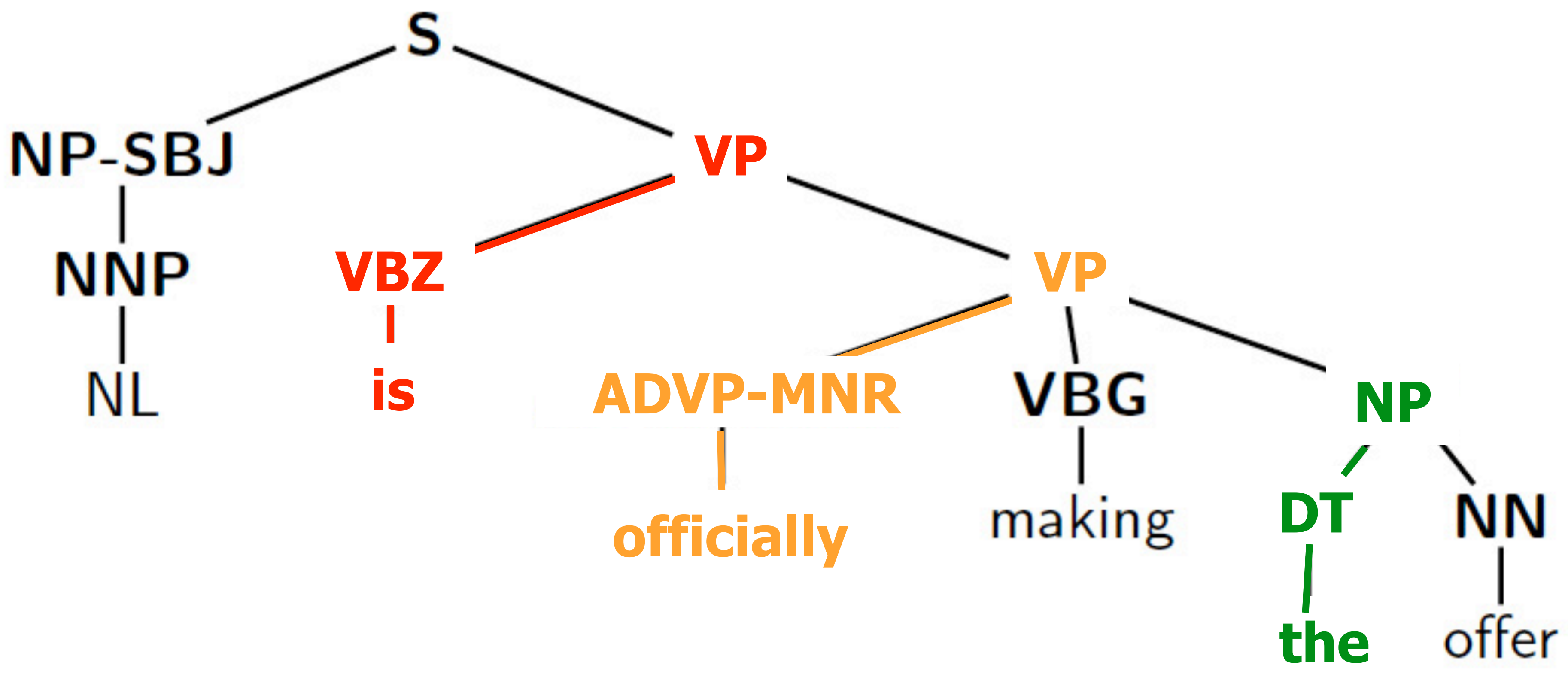
Extracting a TAG: arguments

- **Arguments are combined via substitution**
- **Recurse on the arguments:**



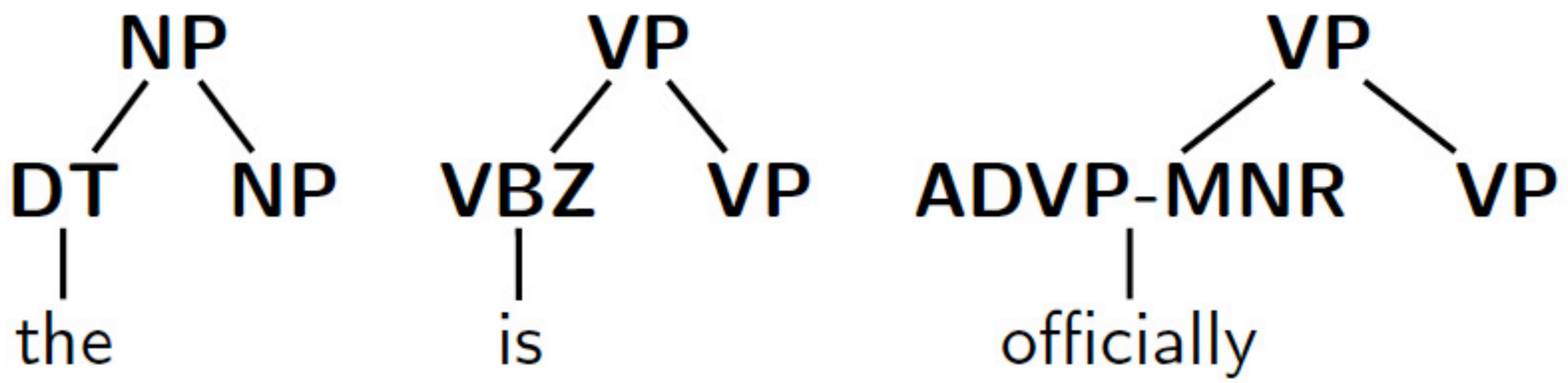
Extracting a TAG: adjuncts

- **Adjuncts require auxiliary trees**
  (use adjunction to be combined with the head)
- **Auxiliary trees require a foot node**
  (with the same label as the root)



Extracting a TAG: adjuncts

- **Adjuncts require auxiliary trees**
  (use adjunction to be combined with the head)
- **Auxiliary trees require a foot node**
  (with the same label as the root)


NP
DT
NP
the
VP
VBZ
VP
is
VP
ADVP-MNR
VP
officially