CS447: Natural Language Processing

http://courses.engr.illinois.edu/cs447

Lecture 20: Expressive grammars

Julia Hockenmaier

juliahmr@illinois.edu

3324 Siebel Center

Projects and Literature Reviews

First report due Nov 15

(PDF written in LaTeX; no length restrictions; submission through Compass)

Purpose of this first report:

Check-in to make sure that you're on track (or, if not, that we can spot problems)

Get feedback from your peers

Rubrics for the *final* reports (due on Reading Day):

https://courses.engr.illinois.edu/CS447/LiteratureReviewRubric.pdf

https://courses.engr.illinois.edu/CS447/FinalProjectRubric.pdf

Projects and Literature Reviews

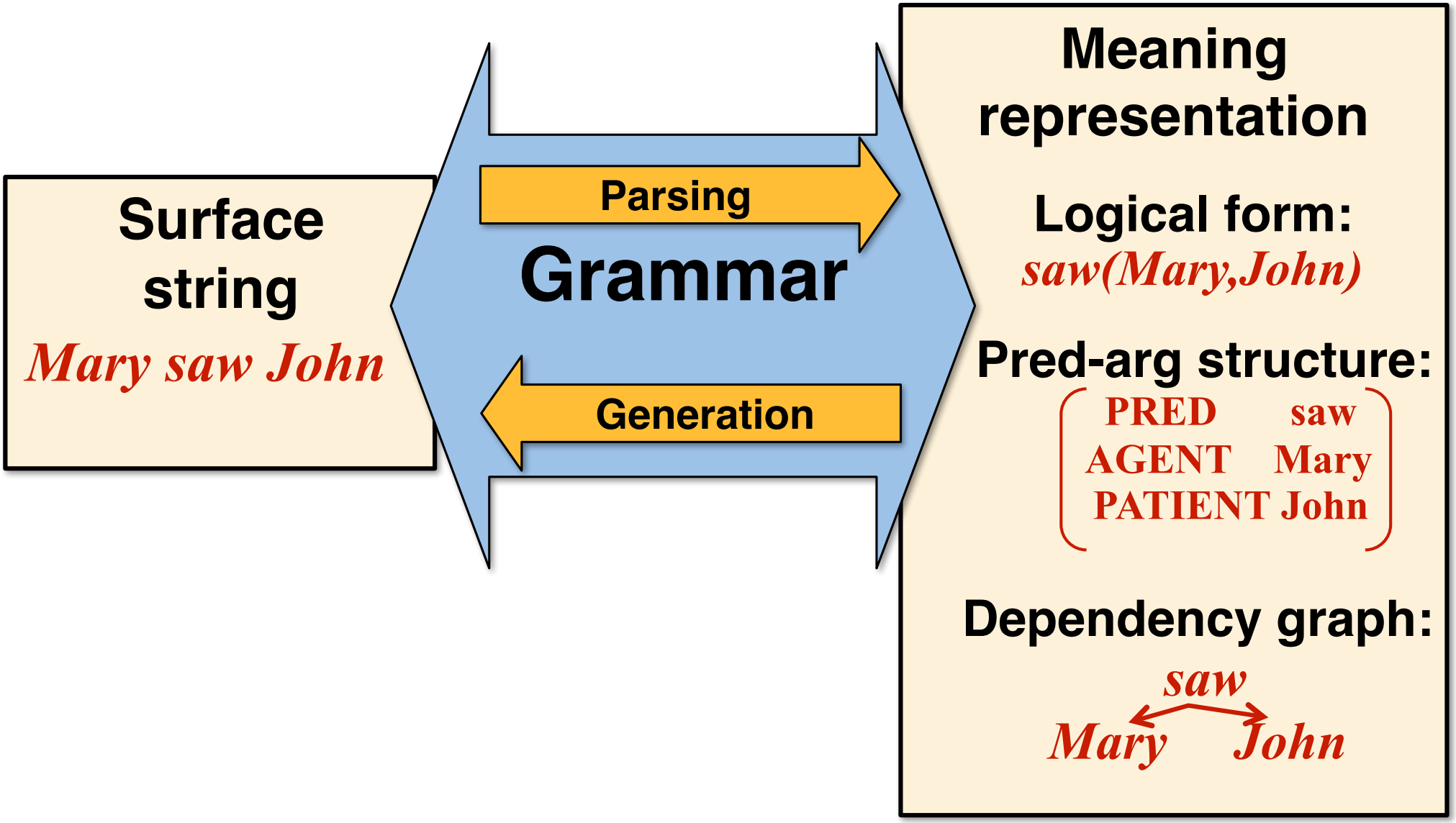
Guidelines for first Project Report:

- What is your project about?
- What are the relevant papers you are building on?
- What data are you using?
- What evaluation metric will you be using?
- What models will you implement/evaluate?
- What is your to-do list?

Guidelines for first Literature Review Report:

- What is your literature review about?
- (What task or what kind of models?
- Do you have any specific questions or focus?)
- What are the papers you will review?
- (If you already have it, give a brief summary of each of them)
- What's your to-do list?

Why grammar?


Surface string
Mary saw John
Parsing
Grammar
Generation
Meaning representation
Logical form:
saw(Mary,John)
Pred-arg structure:
PRED saw
AGENT Mary
PATIENT John
Dependency graph:
saw
Mary
John

Grammar formalisms

Formalisms provide a **language** in which linguistic theories can be expressed and implemented

Formalisms define **elementary objects** (trees, strings, feature structures) and **recursive operations** which generate complex objects from simple objects.

Formalisms may impose **constraints** (e.g. on the kinds of dependencies they can capture)

How do grammar formalisms differ?

Formalisms define different **representations**

Tree-adjoining Grammar (TAG):

Fragments of phrase-structure trees

Lexical-functional Grammar (LFG):

Annotated phrase-structure trees (c-structure)
linked to feature structures (f-structure)

Combinatory Categorial Grammar (CCG):

Syntactic categories paired with meaning representations

Head-Driven Phrase Structure Grammar(HPSG):

Complex feature structures (Attribute-value matrices)

The dependencies so far:

Arguments:

Verbs take arguments: subject, object, complements, ...

Heads subcategorize for their arguments

Adjuncts/Modifiers:

Adjectives modify nouns, adverbs modify VPs or adjectives,

PPs modify NPs or VPs

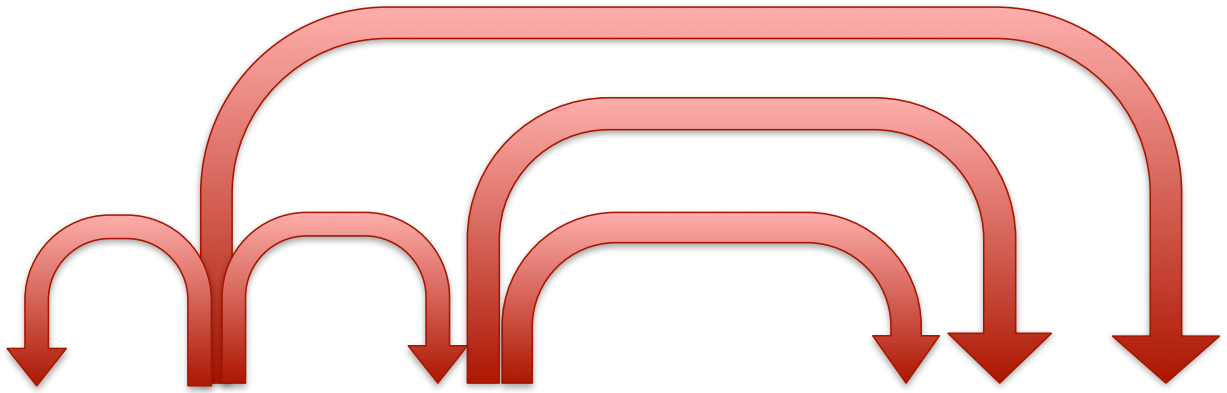
Modifiers subcategorize for the head

Typically, these are *local* dependencies: they can be expressed *within individual CFG rules*

Context-free grammars

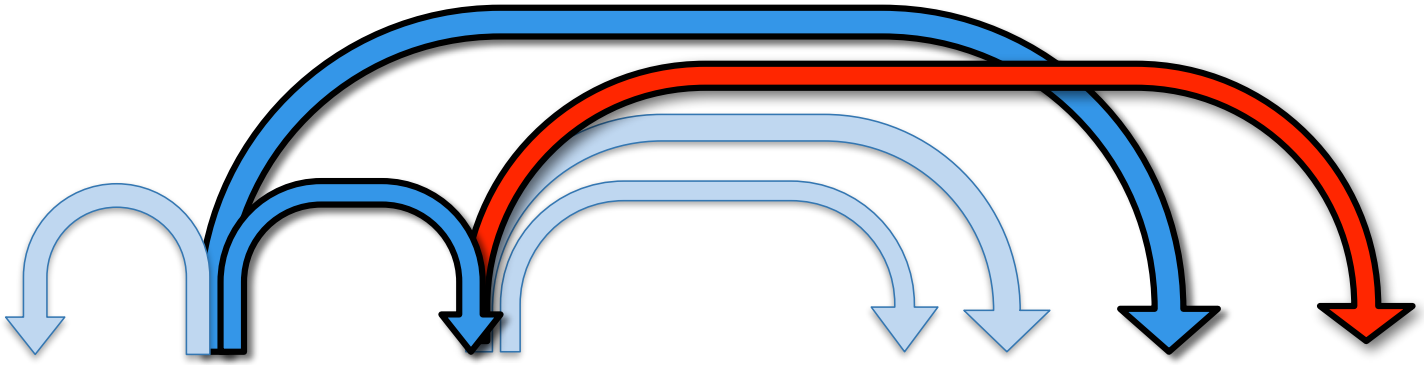
CFGs capture only **nested** dependencies

- The dependency graph is a **tree**
- The dependencies **do not cross**



Beyond CFGs: Nonprojective dependencies

Dependencies form a **tree with crossing branches**



Non-projective dependencies

(Non-local) scrambling: In a sentence with multiple verbs, the argument of a verb appears in a different clause from that which contains the verb (arises in languages with freer word order than English)

Die Pizza *hat Klaus versprochen zu* ***bringen***
The pizza has Klaus promised to bring
Klaus has promised to bring the pizza

Extraposition: Here, a modifier of the subject NP is moved to the end of the sentence

The ***guy*** *is coming* ***who is wearing a hat***
Compare with the non-extraposed variant
The ***[guy [who is wearing a hat]]*** *is coming*

Topicalization: Here, the argument of the embedded verb is moved to the front of the sentence.

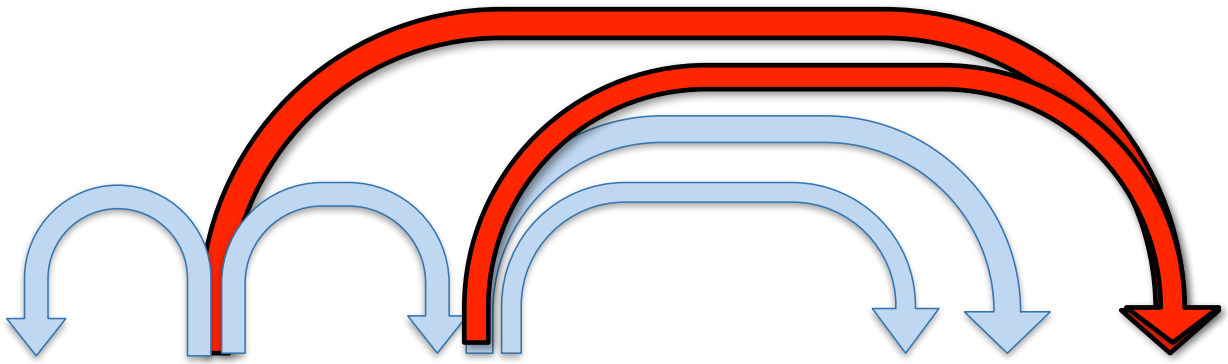
Cheeseburgers*, I [thought [he* ***likes****]]*

Beyond CFGs: Nonlocal dependencies

Dependencies form a **DAG**
(a node may have **multiple incoming edges**)

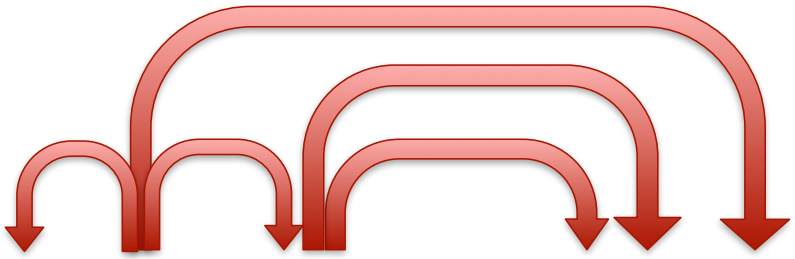
Arise in the following constructions:

- **Control** *(**He** has **promised** me to **go**)*, **raising** (**He seems** to **go**)
- **Wh-movement** *(the **man** who you **saw** yesterday **is** here again)*,
- **Non-constituent** coordination
  (right-node raising, gapping, argument-cluster coordination)

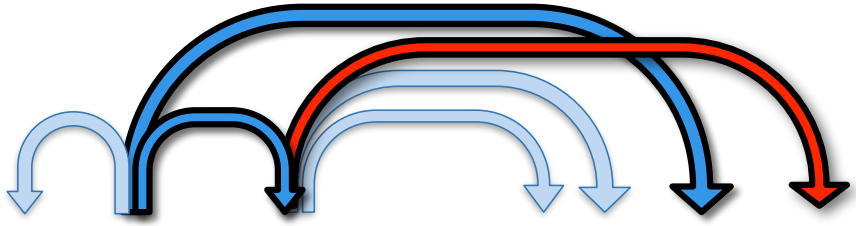


Dependency structures

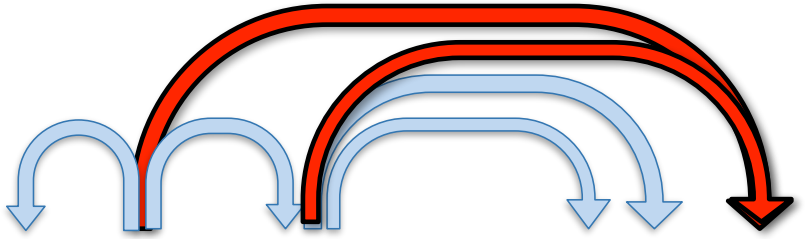
Nested (projective) dependency trees (CFGs)



Non-projective dependency trees



Non-local dependency graphs



Non-local dependencies

Long-range dependencies

Bounded long-range dependencies:

Limited distance between the head and argument

Unbounded long-range dependencies:

Arbitrary distance (within the same sentence) between the head and argument

Unbounded long-range dependencies cannot (in general) be represented with CFGs.

Chomsky's solution:
Add null elements (and co-indexation)

Unbounded nonlocal dependencies

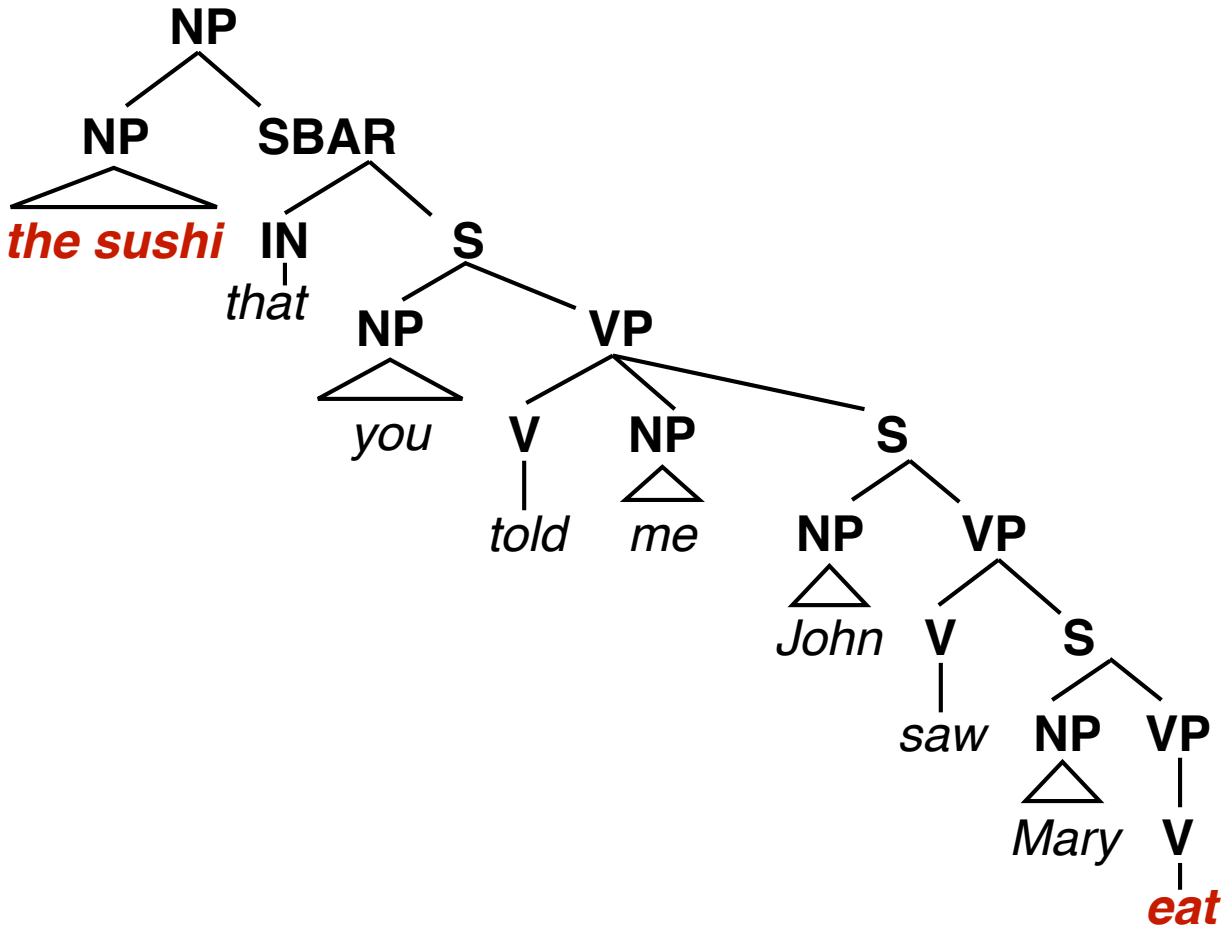
Wh-questions and relative clauses contain ***unbounded* nonlocal dependencies**, where the missing NP may be arbitrarily deeply embedded:

'the sushi that [you told me [John saw [Mary eat]]]'

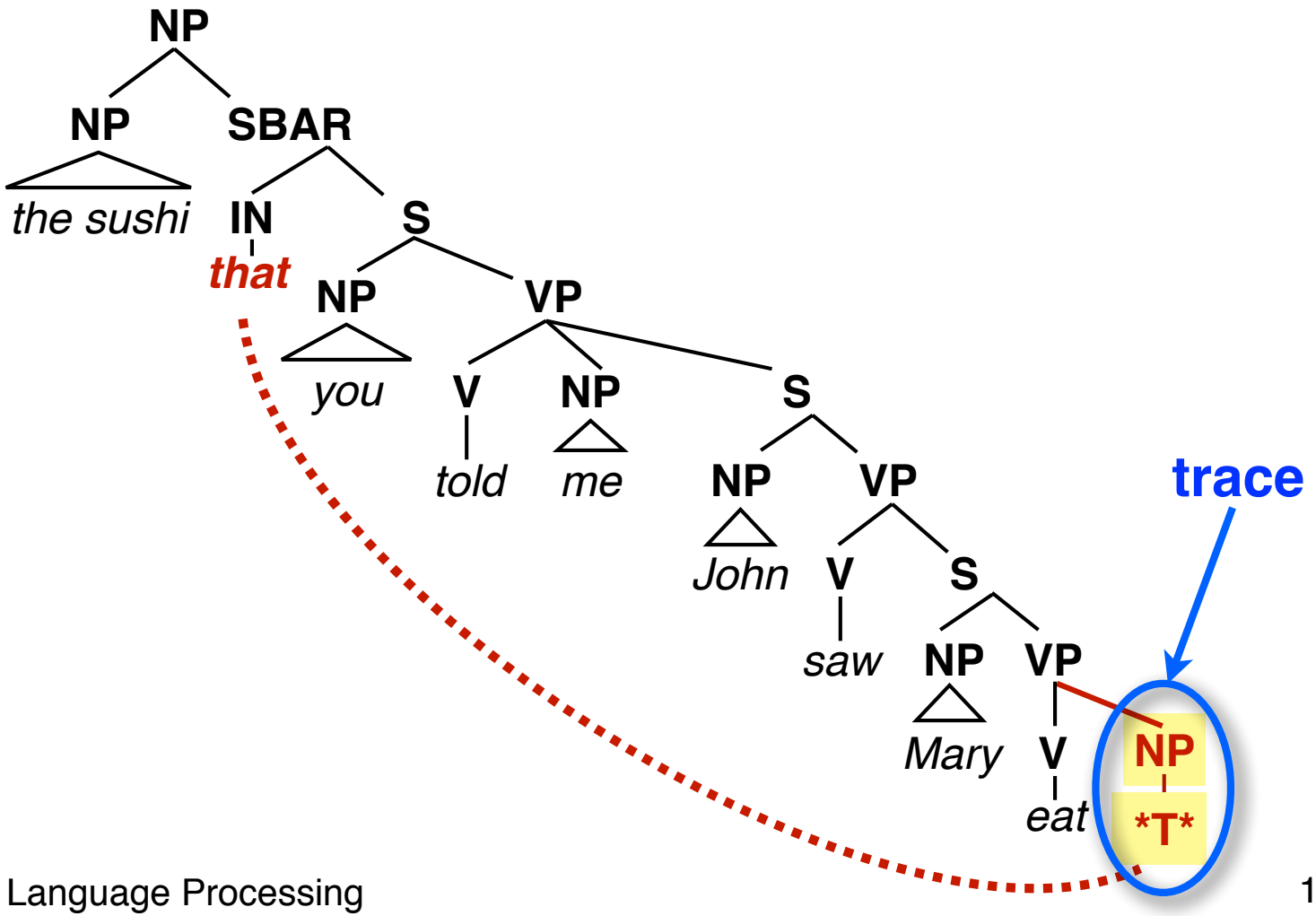
'what [did you tell me [John saw [Mary eat]]]?'

Linguists call this phenomenon **wh-extraction** (wh-movement).

Non-local dependencies in *wh*-extraction



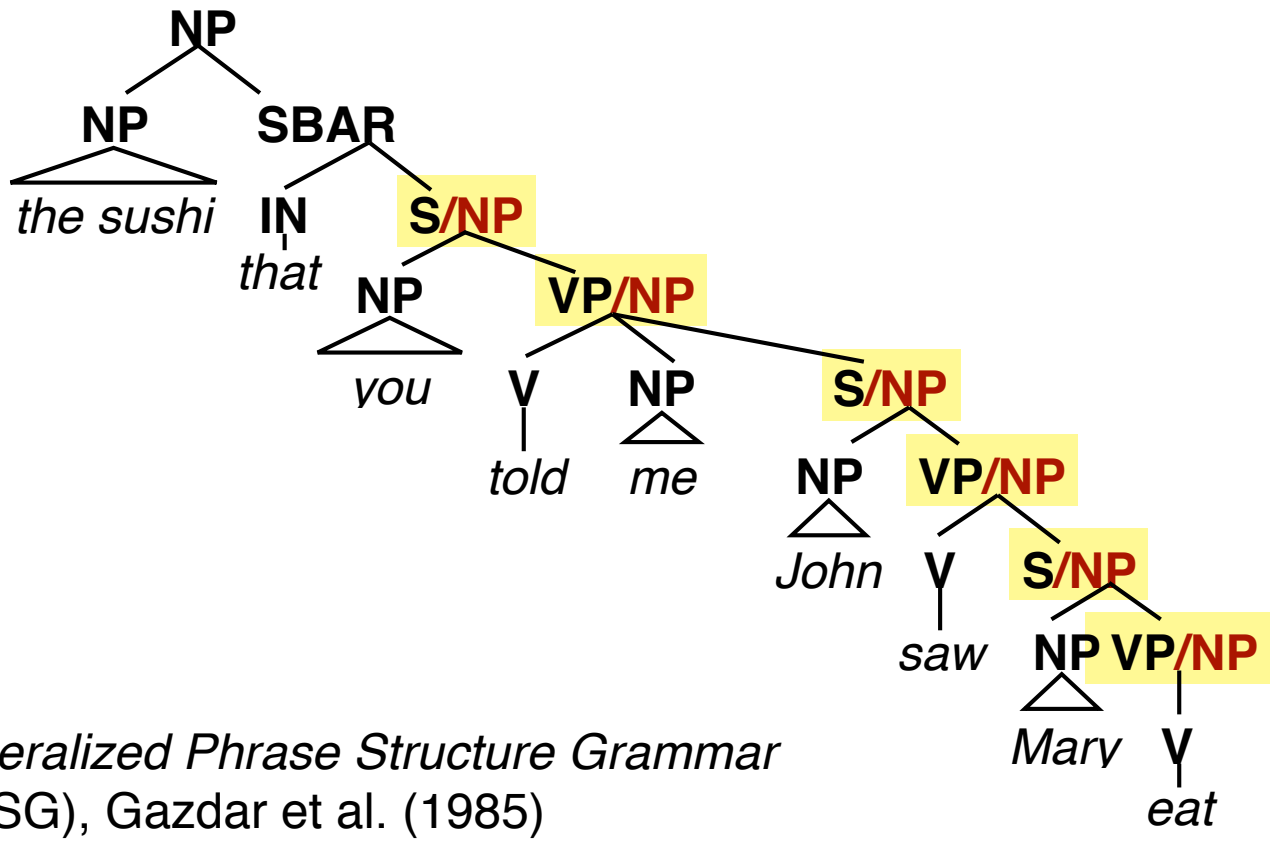
The trace analysis of *wh*-extraction



Slash categories for wh-extraction

Because only one element can be extracted, we can use **slash categories.**

This is still a CFG: the set of nonterminals is finite.

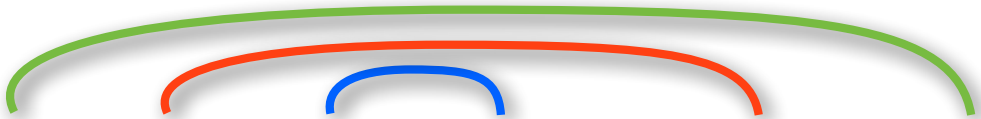


Generalized Phrase Structure Grammar (GPSG), Gazdar et al. (1985)

German: center embedding

...daß ich [Hans schwimmen] sah
...that I Hans swim saw
...that I saw [Hans swim]

...daß ich [Maria [Hans schwimmen] helfen] sah
...that I Maria Hans swim help saw
...that I saw [Mary help [Hans swim]]

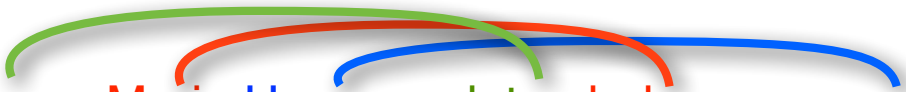


...daß ich [Anna [Maria [Hans schwimmen] helfen] lassen] sah
...that I Anna Maria Hans swim help let saw
...that I saw [Anna let [Mary help [Hans swim]]]

Dutch: cross-serial dependencies

...dat ik Hans zag zwemmen
...that I Hans saw swim
...that I saw [Hans swim]

...dat ik Maria Hans zag helpen zwemmen
...that I Maria Hans saw help swim
...that I saw [Mary help [Hans swim]]

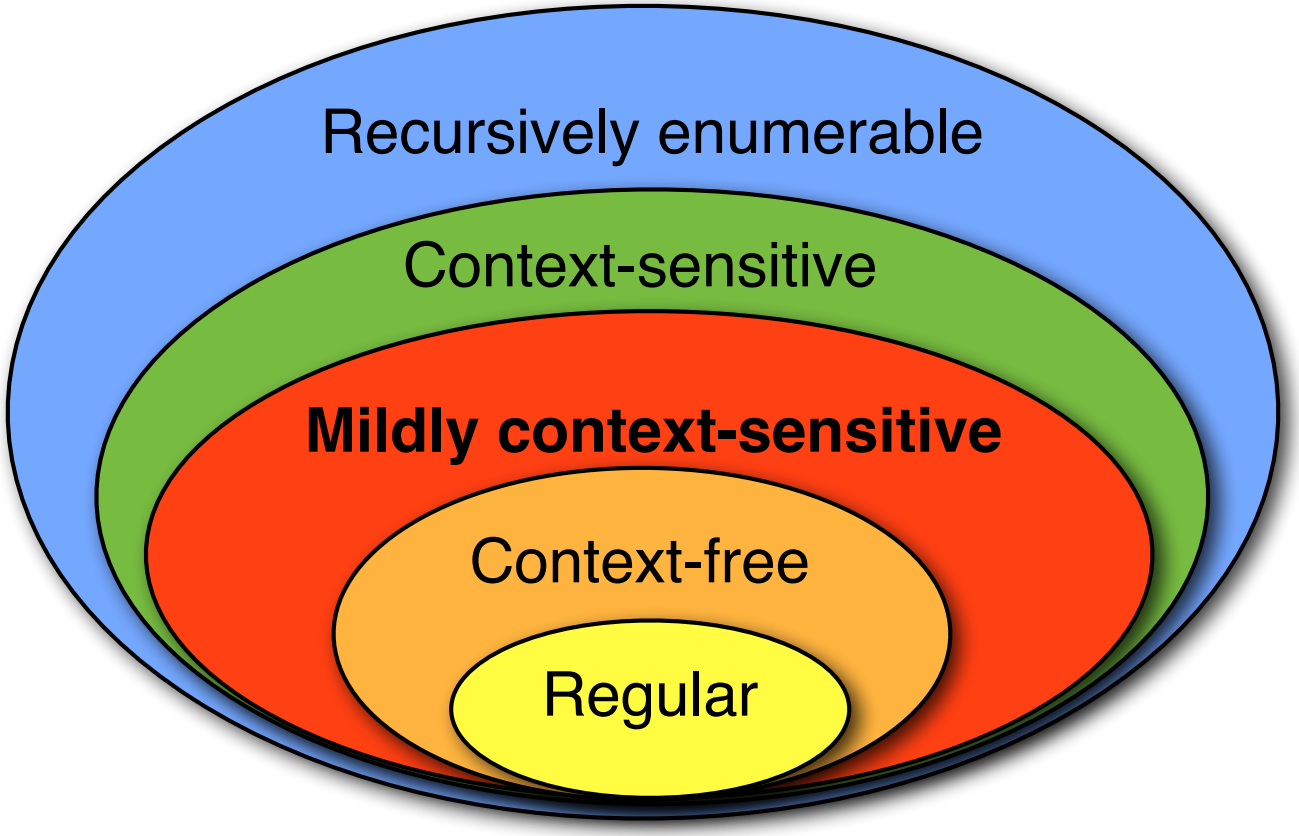


...dat ik Anna Maria Hans zag laten helpen zwemmen
...that I Anna Maria Hans saw let help swim
...that I saw [Anna let [Mary help [Hans swim]]]

Such cross-serial dependencies require
mildly context-sensitive grammars

Two mildly context-sensitive formalisms: TAG and CCG

The Chomsky Hierarchy



Mildly context-sensitive grammars

Contain all context-free grammars/languages

Can be **parsed in polynomial time** (TAG/CCG: $O(n^6)$)

(*Strong* generative capacity) capture certain kinds of dependencies: **nested** (like CFGs) and **cross-serial** (like the Dutch example), but not the MIX language:

MIX: the set of strings $w \in \{a, b, c\}^*$ that contain equal numbers of as, bs and cs

Have the **constant growth** property:
the length of strings grows in a linear way
The power-of-2 language $\{a^{2^n}\}$ does not have the constant growth propery.

TAG and CCG are lexicalized formalisms

The lexicon:

- pairs words with elementary objects
- specifies all language-specific information (e.g. subcategorization information)

The grammatical operations:

- are universal
- define (and impose constraints on) recursion.

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

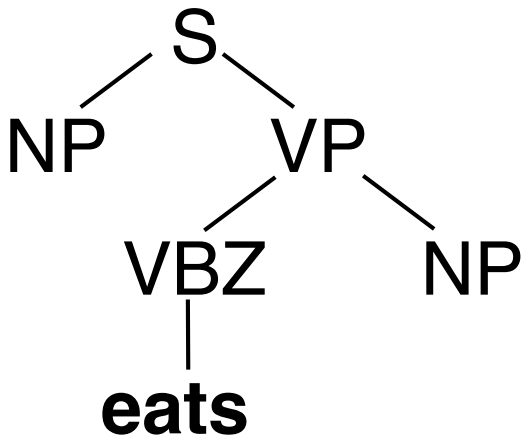
AK Joshi and Y Schabes (1996) Tree Adjoining Grammars. In G. Rosenberg and A. Salomaa, Eds., Handbook of Formal Languages

Extended domain of locality

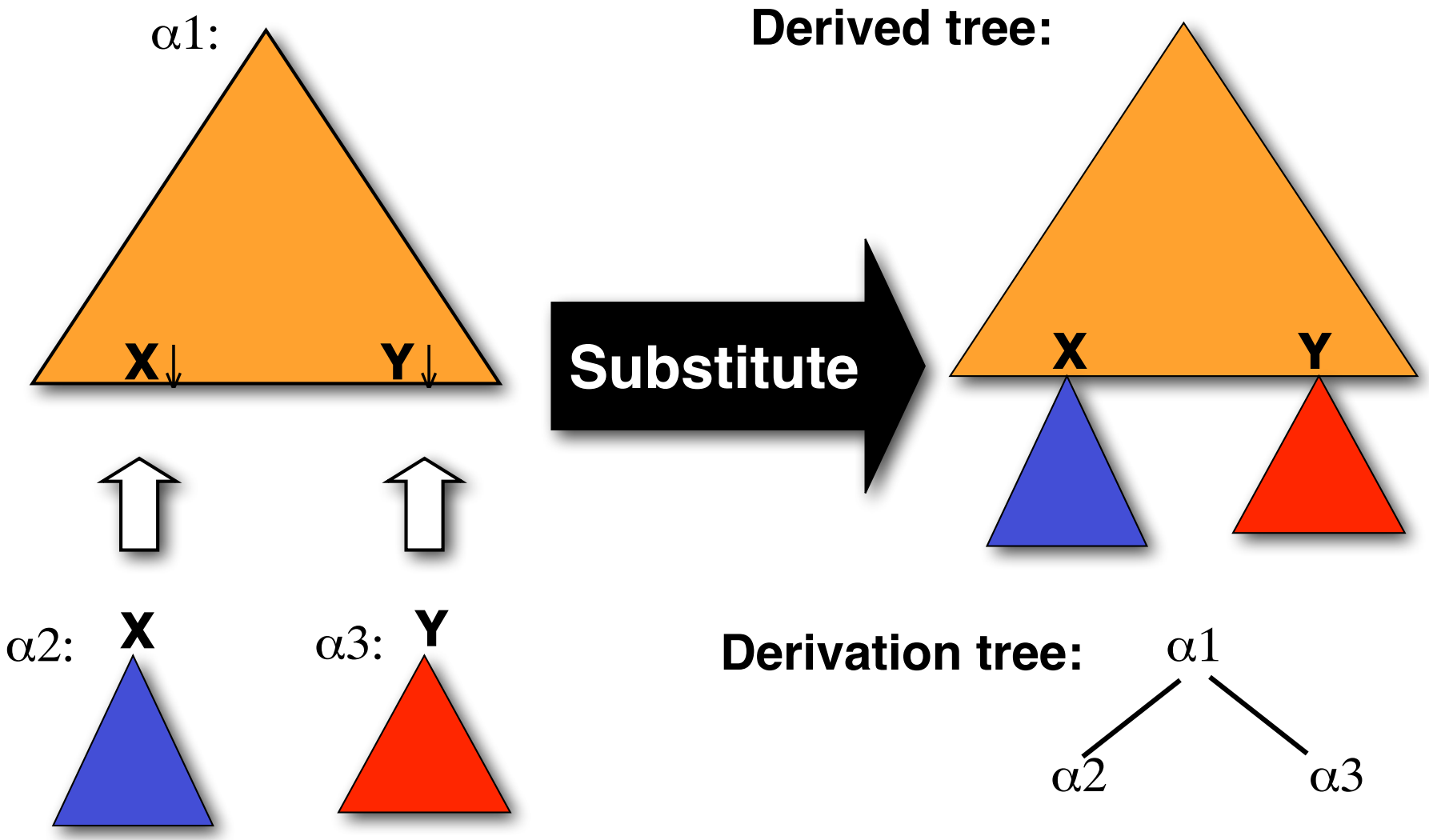
We want to capture **all arguments of a word** in a **single elementary object**.

We also want to retain certain syntactic structures (e.g. VPs).

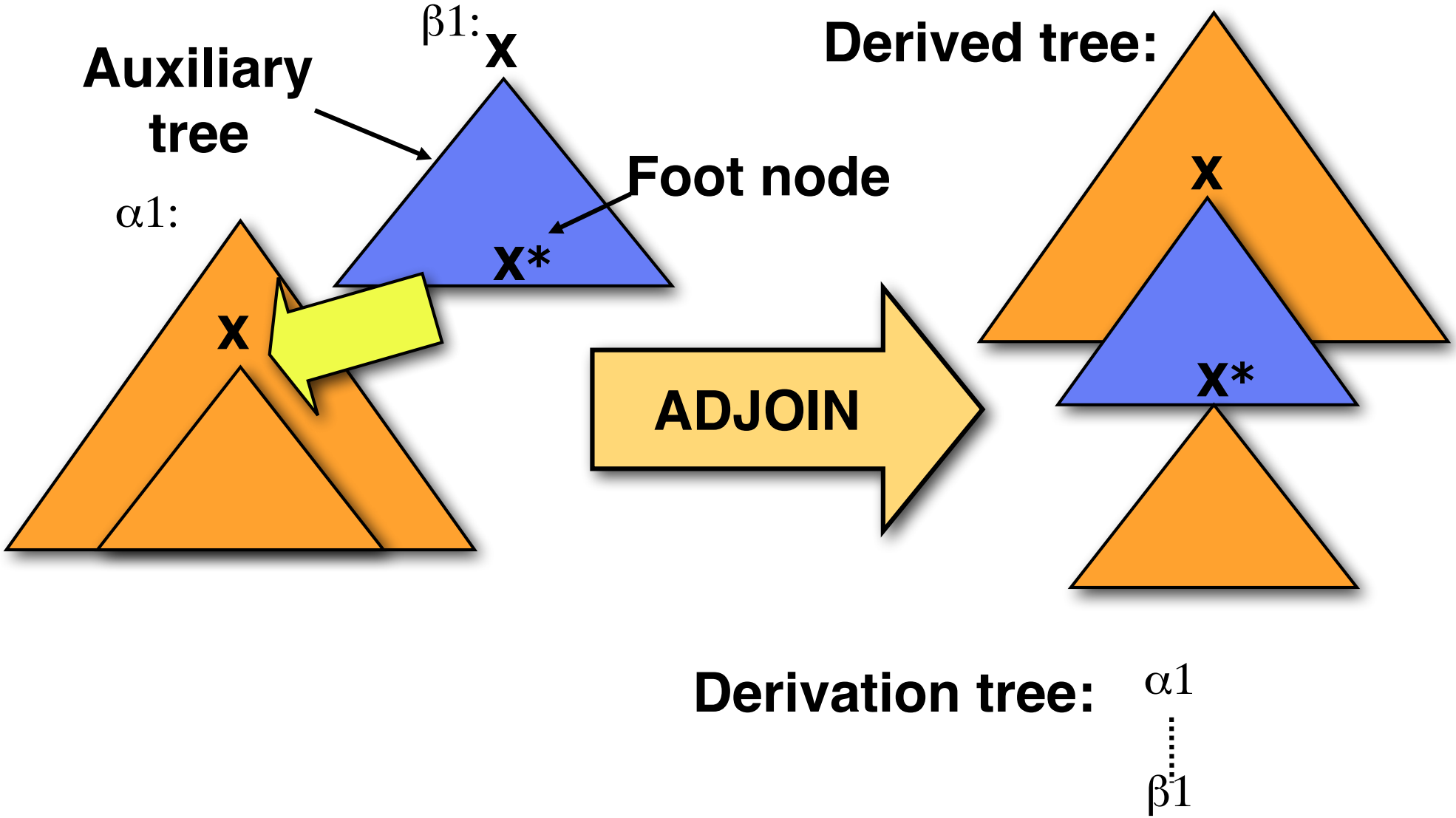
Our elementary objects are tree fragments:



TAG substitution (arguments)

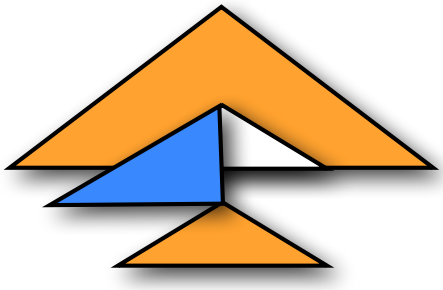


TAG adjunction

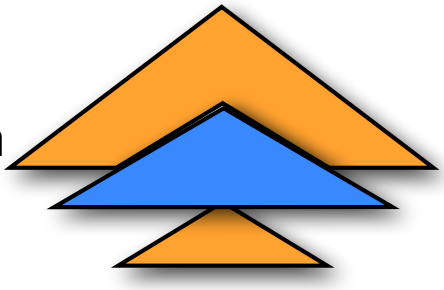


The effect of adjunction

TIG: sister adjunction



TAG: wrapping adjunction



No adjunction: TSG (Tree substitution grammar)

TSG is context-free

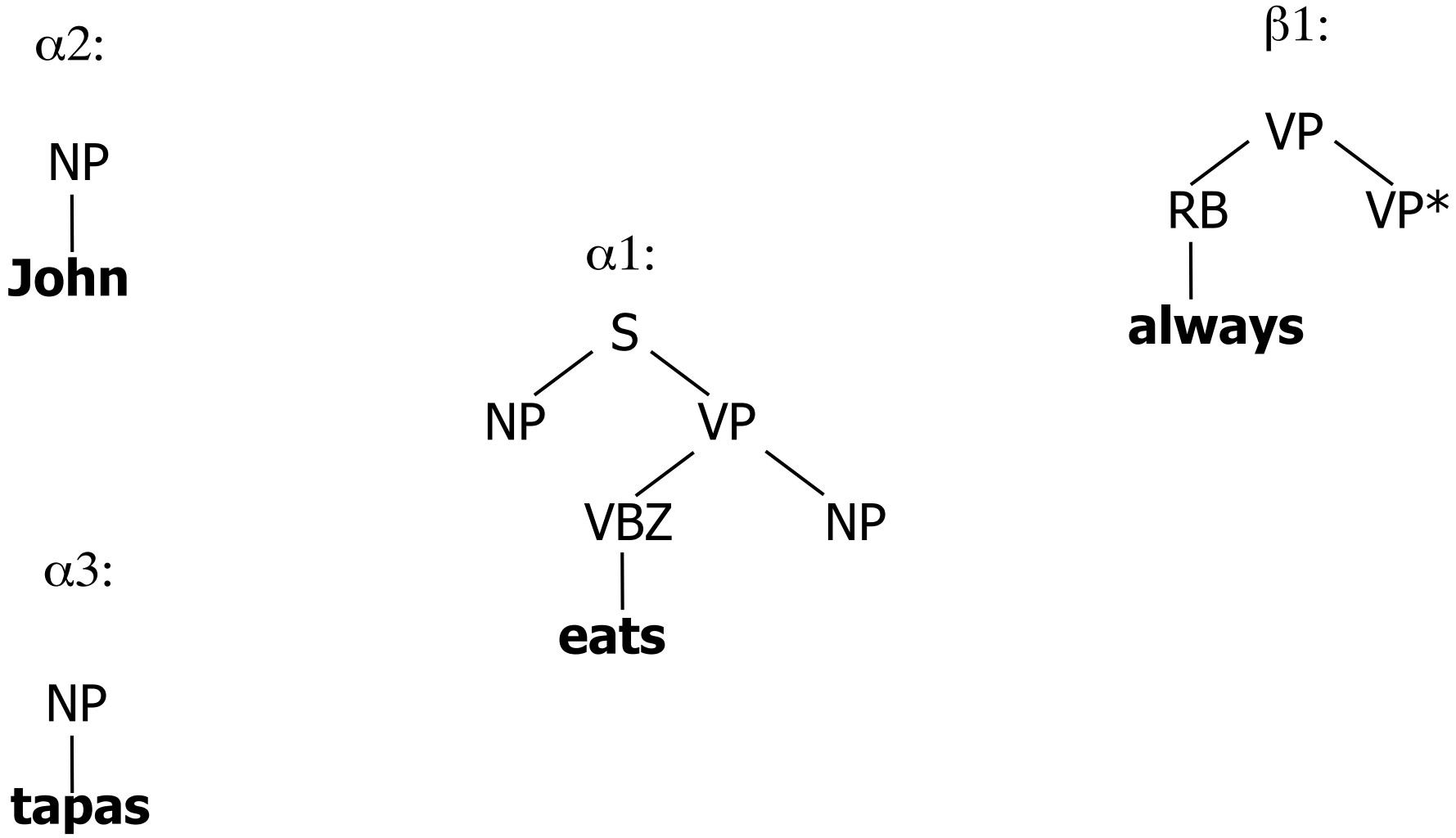
Sister adjunction: TIG (Tree insertion grammar)

TIG is also context-free, but has a linguistically more adequate treatment of modifiers

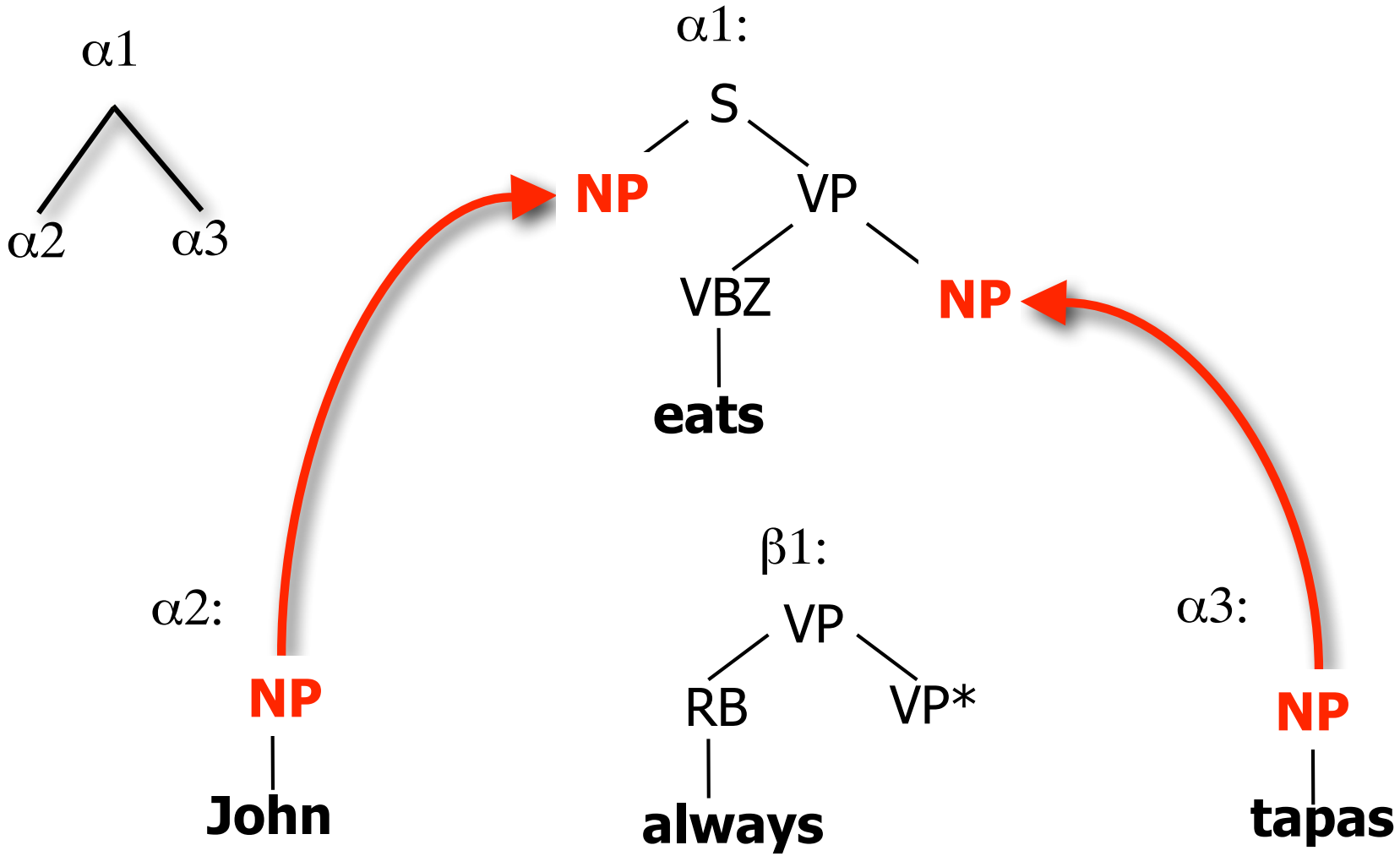
Wrapping adjunction: TAG (Tree-adjoining grammar)

TAG is mildy context-sensitive

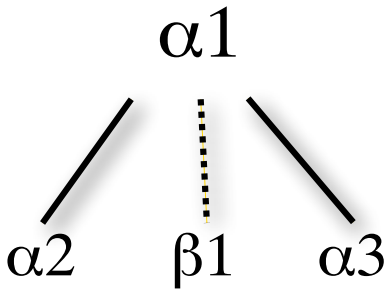
A small TAG lexicon

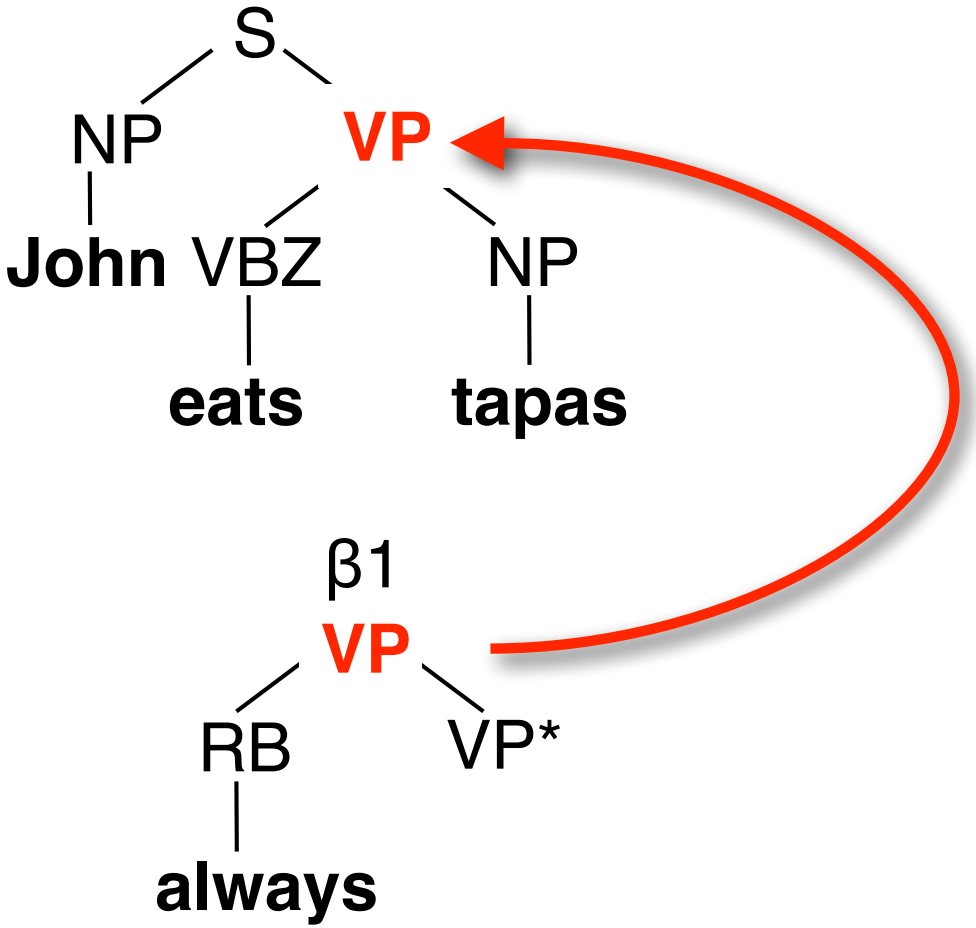


A TAG derivation

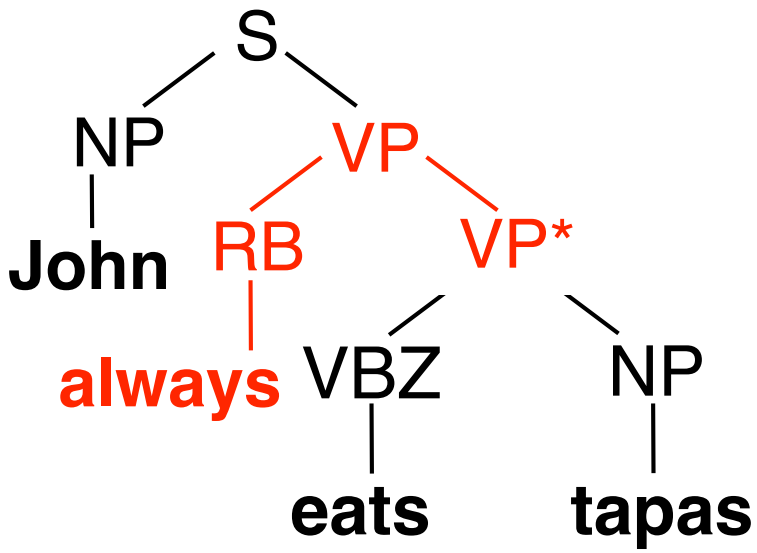


A TAG derivation



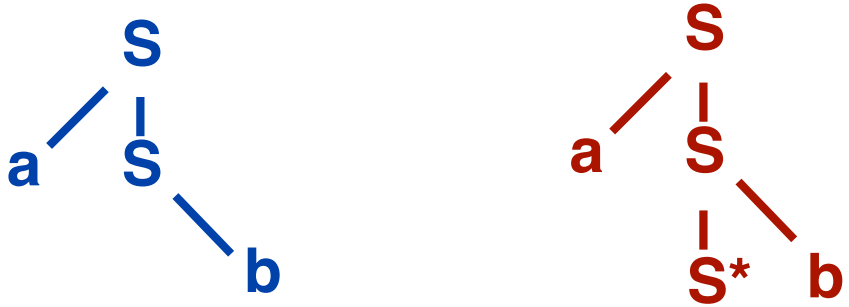


A TAG derivation

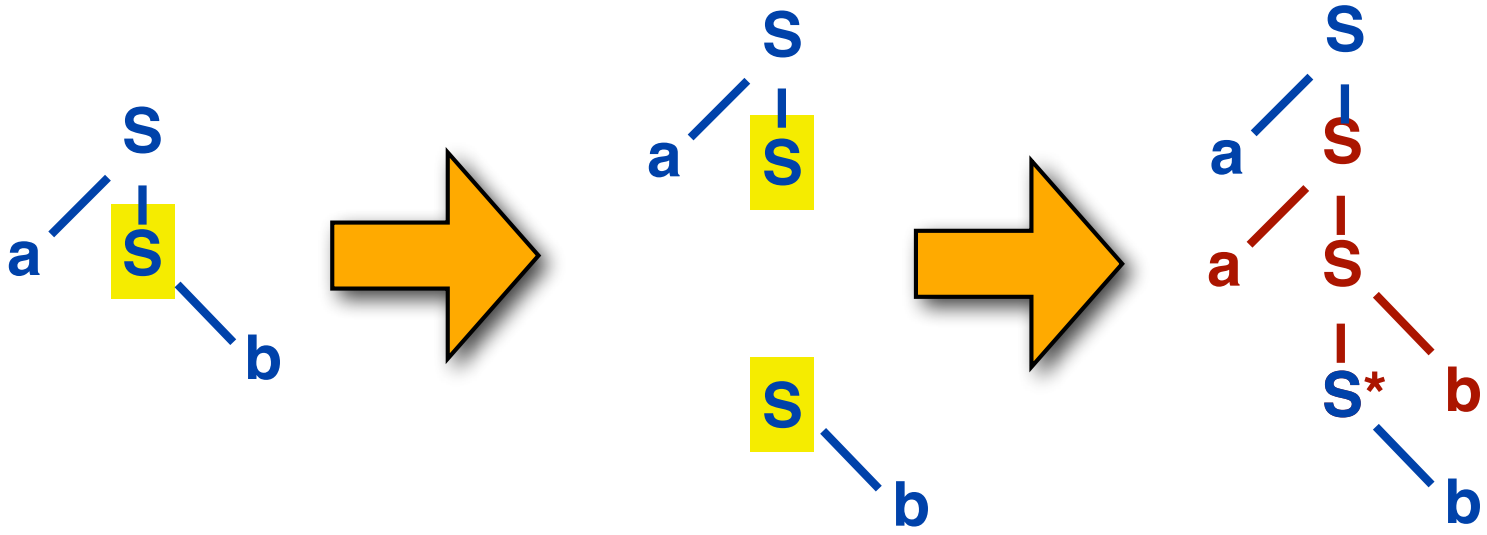


$a^n b^n$: Cross-serial dependencies

Elementary trees:



Deriving aabb



Combinatory Categorial Grammar

CCG: the machinery

Categories:

specify subcat lists of words/constituents.

Combinatory rules:

specify how constituents can combine.

The lexicon:

specifies which categories a word can have.

Derivations:

spell out process of combining constituents.

CCG categories

Simple (atomic) categories: NP, S, PP

Complex categories (functions):

Return a **result** when combined with an **argument**

VP, intransitive verb	**S\NP**
Transitive verb	**(S\NP)/NP**
Adverb	**(S\NP)\(S\NP)**
Prepositions	**((S\NP)\(S\NP))/NP** **(NP\NP)/NP** **PP/NP**

Function application

Forward application (>):

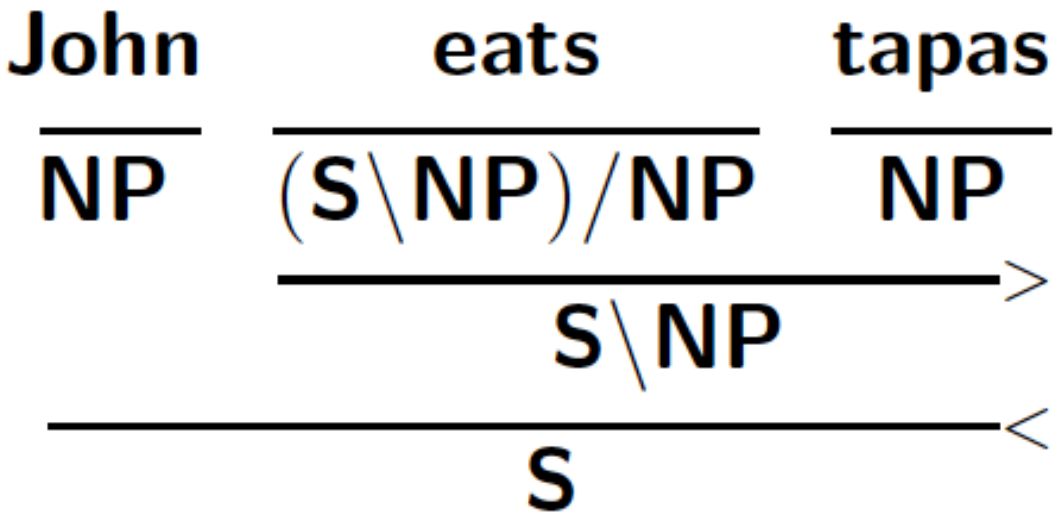
(S\NP)/NP	**NP**	$\Rightarrow_>$	**S\NP**
eats	tapas		eats tapas

Backward application (<):

NP	**S\NP**	$\Rightarrow_<$	**S**
John	eats tapas		John eats tapas

Used in all variants of categorial grammar

A (C)CG derivation



Function composition

Harmonic forward composition (>B):

$X / Y \quad Y / Z \quad \Rightarrow_{>B} \quad X / Z$

Harmonic backward composition (<B):

$Y \backslash Z \quad X \backslash Y \quad \Rightarrow_{<B} \quad X \backslash Z$

Forward crossing composition (>B^{x}):

$X / Y \quad Y \backslash Z \quad \Rightarrow_{>Bx} \quad X \backslash Z$

Backward crossing composition (<B^{x}):

$Y / Z \quad X \backslash Y \quad \Rightarrow_{<Bx} \quad X / Z$

Type-raising

Forward type-raising (>T):

$$X \quad \Rightarrow_{>T} \quad T / (T \backslash X)$$

Backward type-raising (<T):

$$X \quad \Rightarrow_{<T} \quad T \backslash (T / X)$$

Type-raising and composition

Type-raising: X → T/(T\X)

Turns an argument into a function.

NP	→	S/(S\NP)	(subject)
NP	→	(S\NP)\((S\NP)/NP)	(object)

Harmonic composition: X/Y Y/Z → X/Z

Composes two functions (complex categories)

(S\NP)/PP PP/NP	→	(S\NP)/NP
S/(S\NP) (S\NP)/NP	→	S/NP

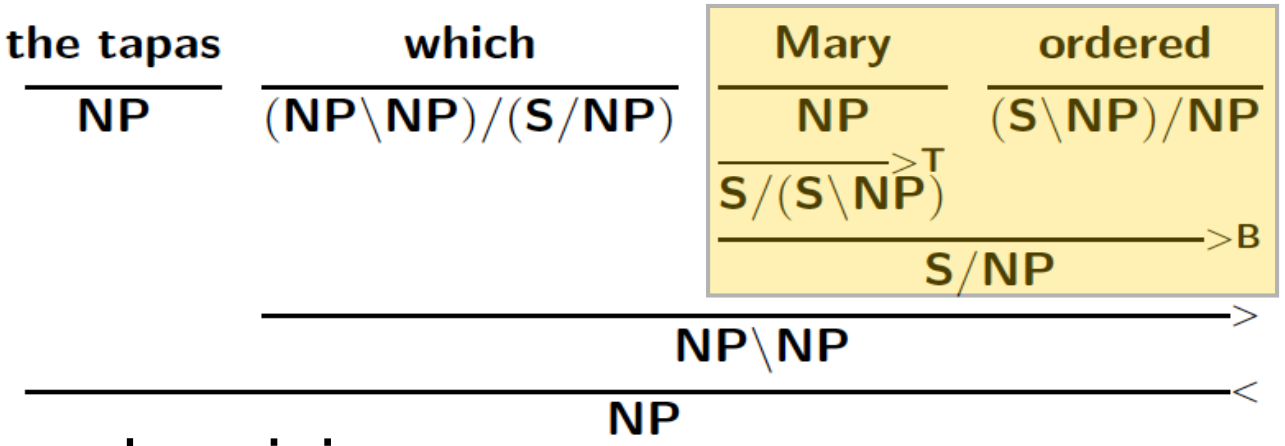
Crossing function composition: X/Y Y\Z → X\Z

Composes two functions (complex categories)

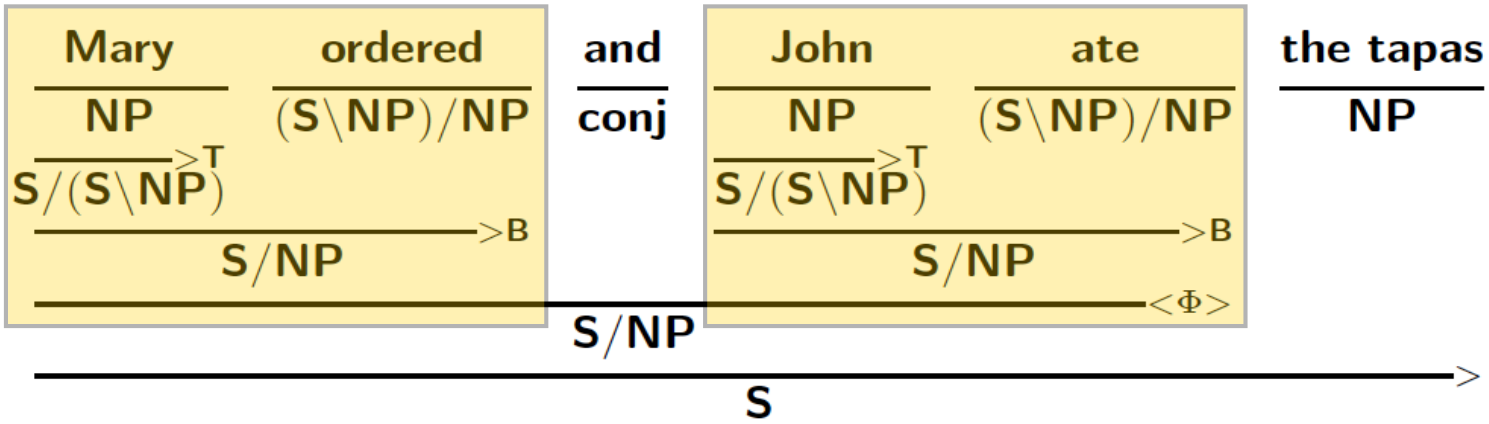
(S\NP)/S S\NP	→	(S\NP)\NP

Type-raising and composition

Wh-movement (relative clause):



Right-node raising:



Function application (> and <):

$$X / Y \quad Y \quad \Rightarrow_{>} \quad X$$

$$Y \quad X \backslash Y \quad \Rightarrow_{<} \quad X$$

Harmonic composition (>B and <B):

$$X / Y \quad Y / Z \quad \Rightarrow_{>B} \quad X / Z$$

$$Y \backslash Z \quad X \backslash Y \quad \Rightarrow_{<B} \quad X \backslash Z$$

Crossing composition (>B^{x} and <B^{x}):

$$X / Y \quad Y \backslash Z \quad \Rightarrow_{>Bx} \quad X \backslash Z$$

$$Y / Z \quad X \backslash Y \quad \Rightarrow_{<Bx} \quad X / Z$$

Generalized composition (>B^{n} and <B^{n}):

$$X / Y \quad (\ldots(Y \mid Z_1)\mid\ldots)\mid Z_n \quad \Rightarrow_{>Bn} \quad (\ldots(X \mid Z_1)\mid\ldots)\mid Z_n$$

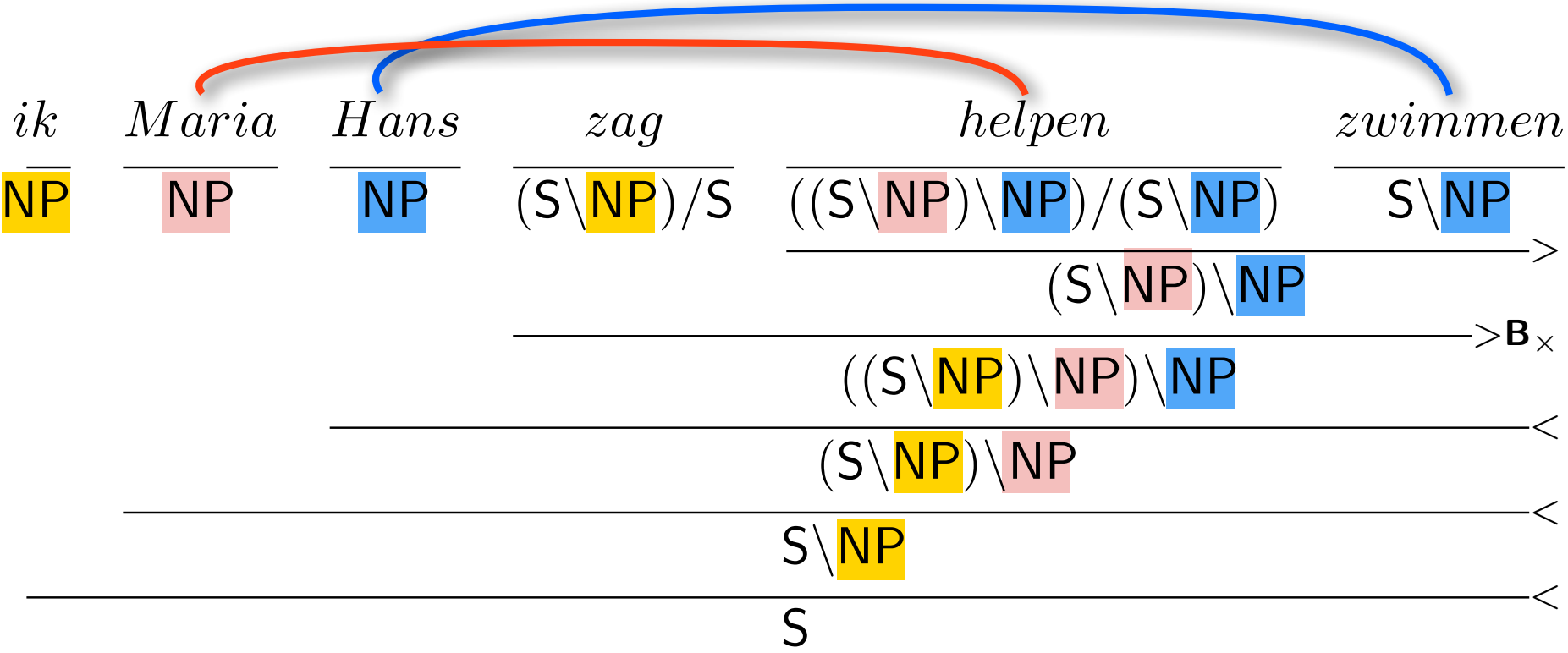
$$(\ldots(Y \mid Z_1)\mid\ldots)\mid Z_n \quad X \backslash Y \quad \Rightarrow_{<Bn} \quad (\ldots(X \mid Z_1)\mid\ldots)\mid Z_n$$

Type-raising (>T and <T):

$$X \quad \Rightarrow_{>B} \quad T / (T \backslash X)$$

$$X \quad \Rightarrow_{<B} \quad T \backslash (T / X)$$

Dutch cross-serial dependencies



Combinatory Categorial Grammar

- CCG is **lexicalized**
  (the "rules" of the grammar are completely general, all language-specific information is given in the lexicon)
- CCG is **mildly context-sensitive**
  (can capture Dutch crossing dependencies, but is still efficiently parseable)
- CCG has a **flexible constituent structure**
- CCG has a unified treatment of extraction/coordination
- CCG has a transparent syntax-semantics interface
  (every syntactic category and operation has a semantic counterpart)
- CCG rules are monotonic
  (movement or traces don't exist)

Today's key concepts

Phenomena that require extensions of standard context-free grammars:

- non-local dependencies
- cross-serial dependencies

Two lexicalized formalisms:

- Tree-adjoining Grammar
- Combinatory Categorial Grammar