Grammar Implementation with Lexicalized Tree Adjoining Grammars and Frame Semantics

Syntactic analyses

Laura Kallmeyer, Timm Lichte, Rainer Osswald & Simon Petitjean

University of Düsseldorf

DGfS CL Fall School, September 12, 2017

SFB 991

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

The ideal grammar formalism

- linguistically adequate:

The ideal grammar formalism

- linguistically adequate:
  - **Phenomena:**
    linearization, agreement, discontinuity, ellipsis, coordination (RNR), …

The ideal grammar formalism

- linguistically adequate:
  - **Phenomena:**
    linearization, agreement, discontinuity, ellipsis, coordination (RNR), …
  - **Generalization:**
    valency, active/passive diathesis, sentence types, alternations, syntax-semantics interface, syntax-discourse interface

The ideal grammar formalism

- linguistically adequate:
  - **Phenomena:**
    linearization, agreement, discontinuity, ellipsis, coordination (RNR), …
  - **Generalization:**
    valency, active/passive diathesis, sentence types, alternations, syntax-semantics interface, syntax-discourse interface
  - intuitive implementation

The ideal grammar formalism

- linguistically adequate:
  - **Phenomena:**
    linearization, agreement, discontinuity, ellipsis, coordination (RNR), …
  - **Generalization:**
    valency, active/passive diathesis, sentence types, alternations, syntax-semantics interface, syntax-discourse interface
  - intuitive implementation
- computationally adequate:

The ideal grammar formalism

- linguistically adequate:
  - **Phenomena:**
    linearization, agreement, discontinuity, ellipsis, coordination (RNR), …
  - **Generalization:**
    valency, active/passive diathesis, sentence types, alternations, syntax-semantics interface, syntax-discourse interface
  - intuitive implementation
- computationally adequate:
  - explicit/formalized

The ideal grammar formalism

- linguistically adequate:
  - **Phenomena:**
    linearization, agreement, discontinuity, ellipsis, coordination (RNR), …
  - **Generalization:**
    valency, active/passive diathesis, sentence types, alternations, syntax-semantics interface, syntax-discourse interface
  - intuitive implementation
- computationally adequate:
  - explicit/formalized
  - decidable, maybe even tractable

The ideal grammar formalism

- linguistically adequate:
  - **Phenomena:**
    linearization, agreement, discontinuity, ellipsis, coordination (RNR), …
  - **Generalization:**
    valency, active/passive diathesis, sentence types, alternations, syntax-semantics interface, syntax-discourse interface
  - intuitive implementation
- computationally adequate:
  - explicit/formalized
  - decidable, maybe even tractable
  - bidirectional

The ideal grammar formalism

- linguistically adequate:
  - **Phenomena:**
    linearization, agreement, discontinuity, ellipsis, coordination (RNR), …
  - **Generalization:**
    valency, active/passive diathesis, sentence types, alternations, syntax-semantics interface, syntax-discourse interface
  - intuitive implementation
- computationally adequate:
  - explicit/formalized
  - decidable, maybe even tractable
  - bidirectional
- psycholinguistically adequate:

The ideal grammar formalism

- linguistically adequate:
  - **Phenomena:**
    linearization, agreement, discontinuity, ellipsis, coordination (RNR), …
  - **Generalization:**
    valency, active/passive diathesis, sentence types, alternations, syntax-semantics interface, syntax-discourse interface
  - intuitive implementation
- computationally adequate:
  - explicit/formalized
  - decidable, maybe even tractable
  - bidirectional
- psycholinguistically adequate:
  - strictly incremental derivations

The ideal grammar formalism

- linguistically adequate:
  - **Phenomena:**
    linearization, agreement, discontinuity, ellipsis, coordination (RNR), …
  - **Generalization:**
    valency, active/passive diathesis, sentence types, alternations, syntax-semantics interface, syntax-discourse interface
  - intuitive implementation
- computationally adequate:
  - explicit/formalized
  - decidable, maybe even tractable
  - bidirectional
- psycholinguistically adequate:
  - strictly incremental derivations
  - correct predictions wrt. processing complexity

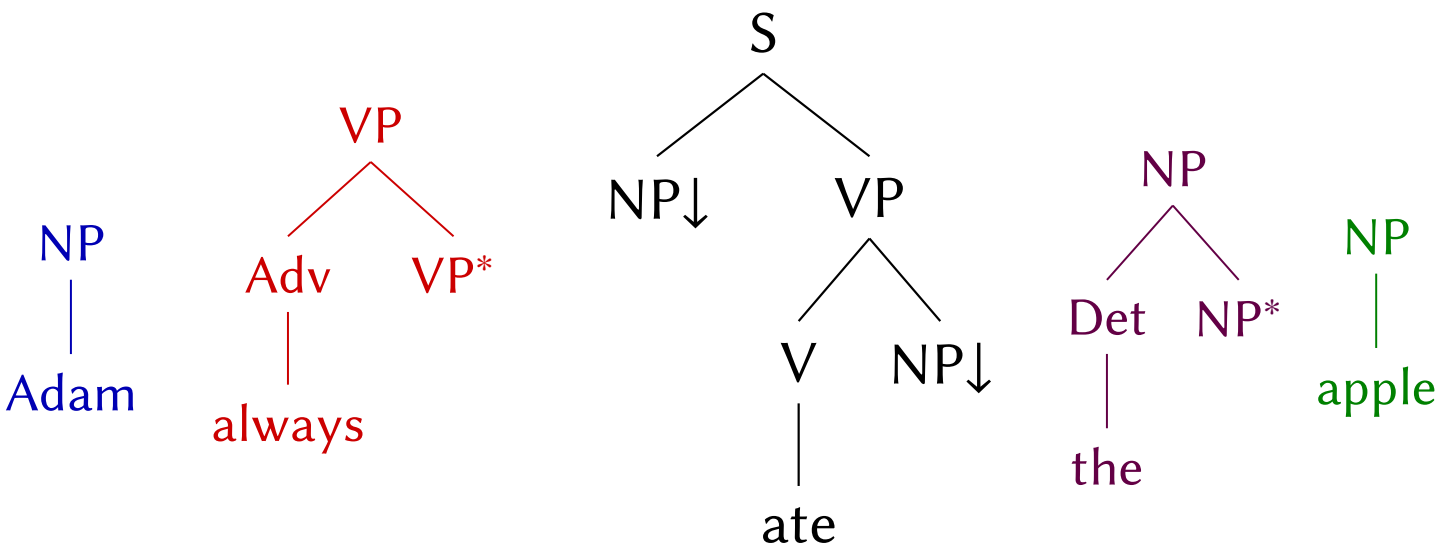
Outline of today's course

1. The derivation tree
2. Design principles for elementary trees
3. Sample derivations
   - NP and PP complements
   - Sentential complements and long-distance dependencies
   - Modifiers
4. Feature based TAG
5. Summary and outlook

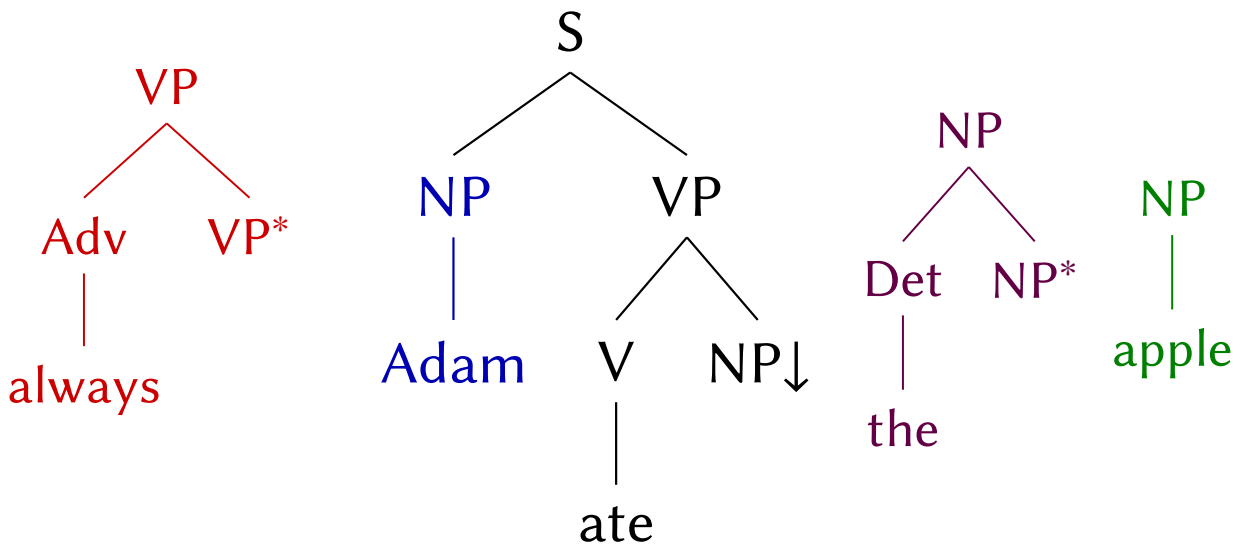
Outline of today's course

1. The derivation tree
2. Design principles for elementary trees
3. Sample derivations
   - NP and PP complements
   - Sentential complements and long-distance dependencies
   - Modifiers
4. Feature based TAG
5. Summary and outlook

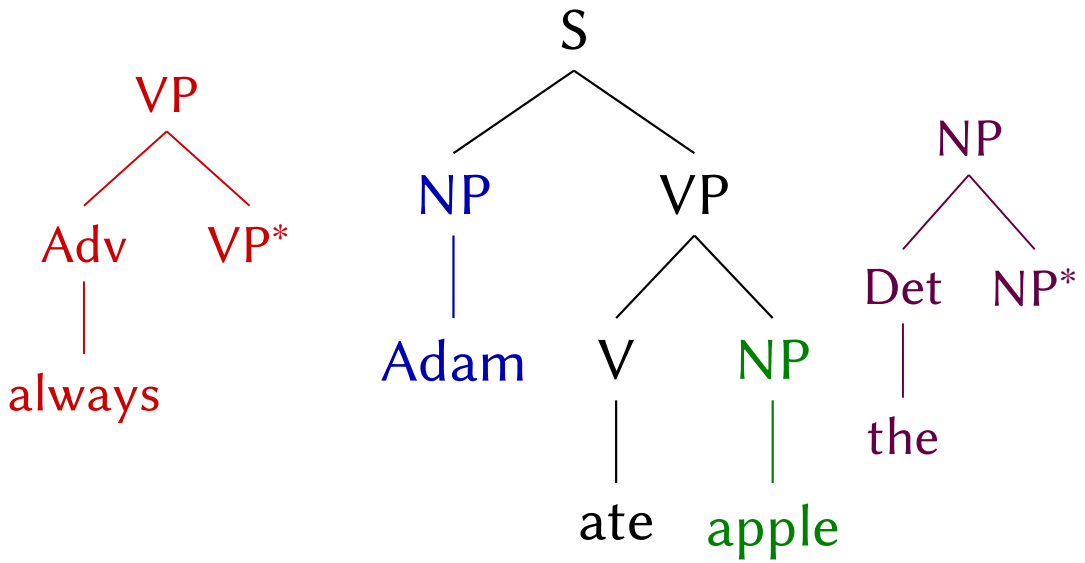
Example derivation



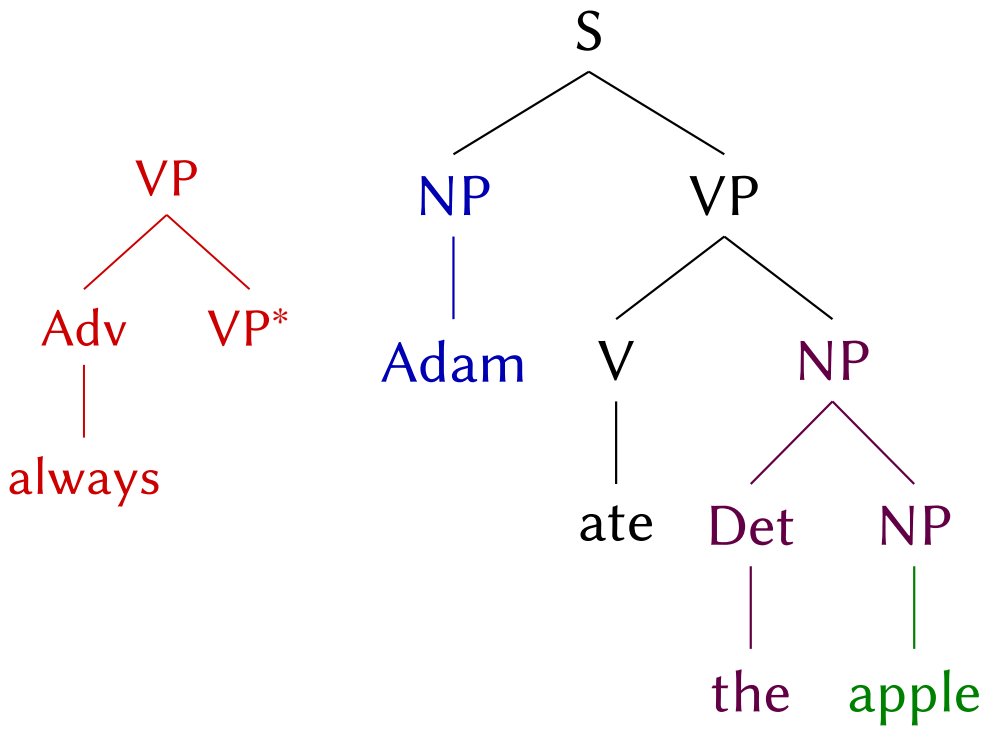
Example derivation



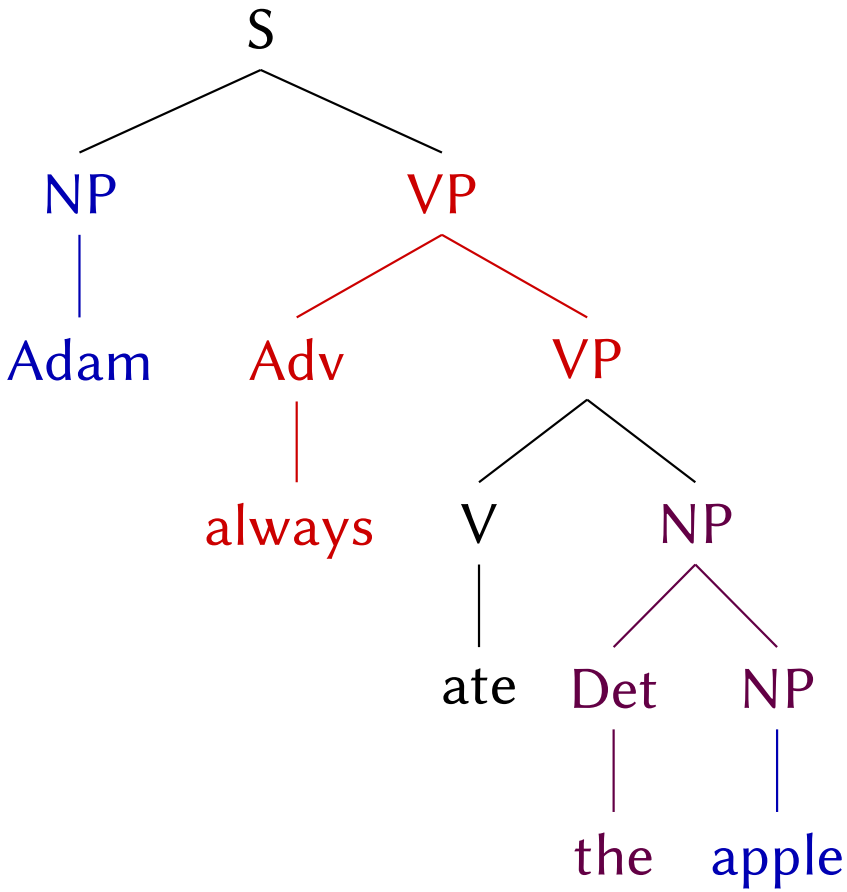
Example derivation



Example derivation



Example derivation



Derivation trees

TAG derivations are uniquely described by **derivation trees**. The derivation tree contains:

Derivation trees

TAG derivations are uniquely described by **derivation trees**. The derivation tree contains:

- **nodes** for all elementary trees used in the derivation, and

Derivation trees

TAG derivations are uniquely described by **derivation trees**. The derivation tree contains:

- **nodes** for all elementary trees used in the derivation, and
- **edges** for all adjunctions and substitutions performed throughout the derivation, and

Derivation trees

TAG derivations are uniquely described by **derivation trees**. The derivation tree contains:

- **nodes** for all elementary trees used in the derivation, and
- **edges** for all adjunctions and substitutions performed throughout the derivation, and
- **edge labels** indicating the target node of the rewriting operation.

Derivation trees

TAG derivations are uniquely described by **derivation trees**. The derivation tree contains:

- **nodes** for all elementary trees used in the derivation, and
- **edges** for all adjunctions and substitutions performed throughout the derivation, and
- **edge labels** indicating the target node of the rewriting operation.

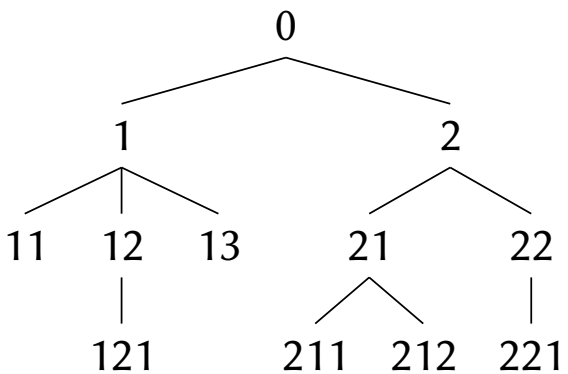
Whenever an elementary tree γ rewrites the node at Gorn address p in the elementary tree γ', there is an edge from γ' to γ labeled with p.

Note that in principle derivation trees are unordered. As a convention, daughters are ordered according to their addresses.

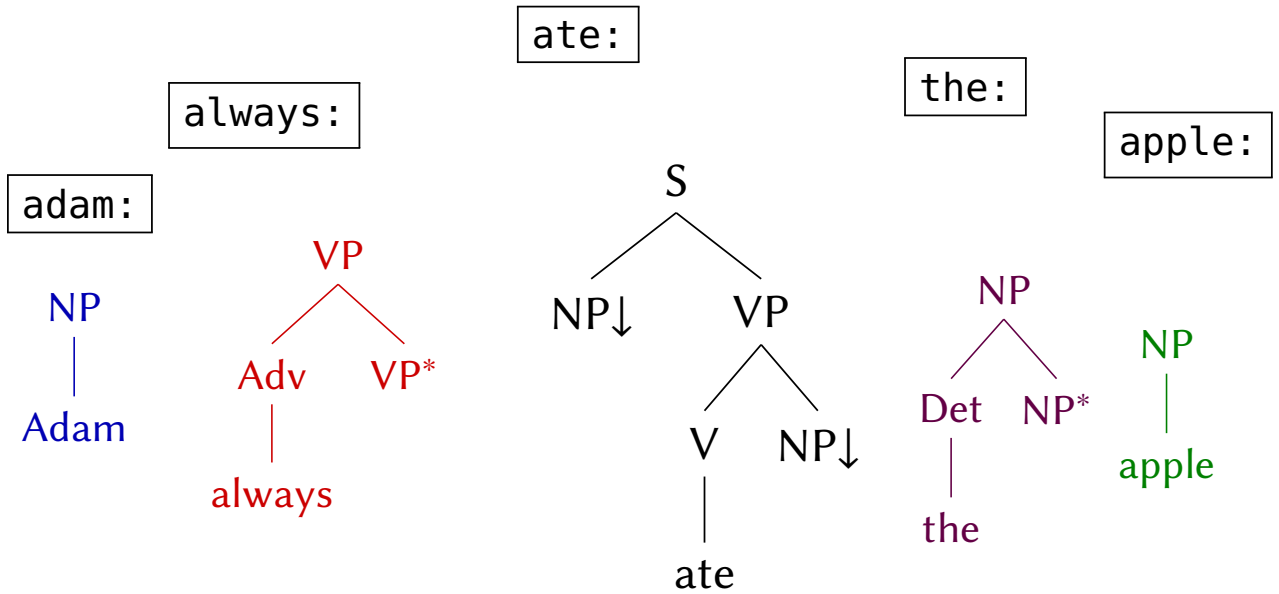
Derivation trees

For the node addresses of elementary trees, **Gorn addresses** are used:

- the root has address ϵ (or 0)
- the ith daughter of the node with address p has address pi.

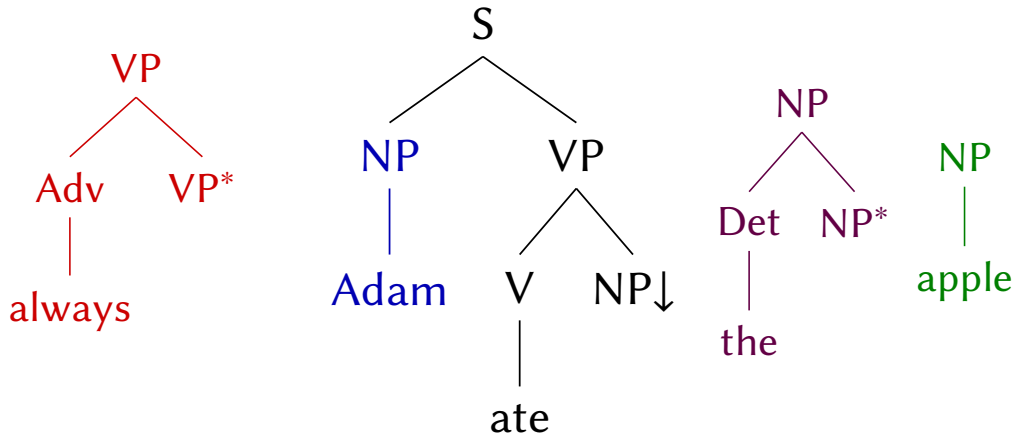


Derivation trees: example

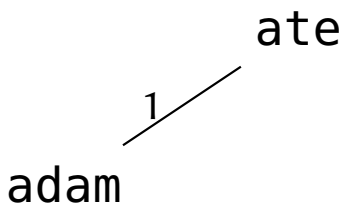


Derivation tree:

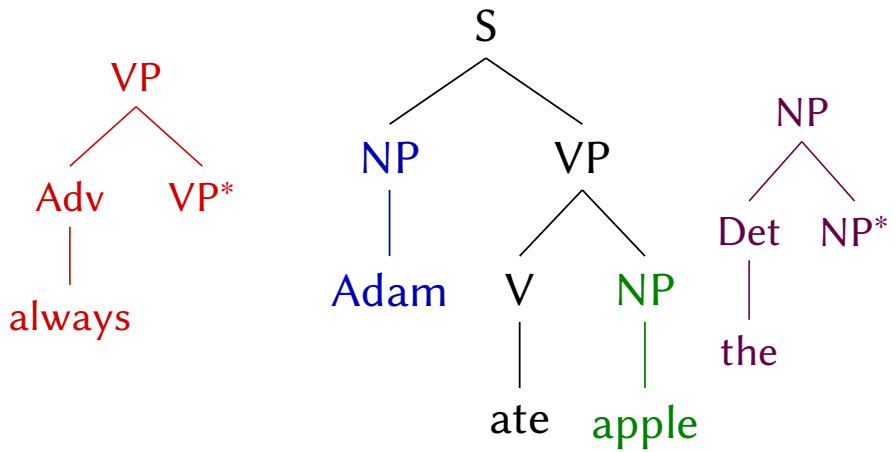
Derivation trees: example



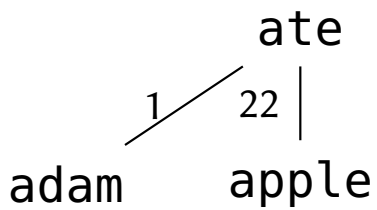
Derivation tree:



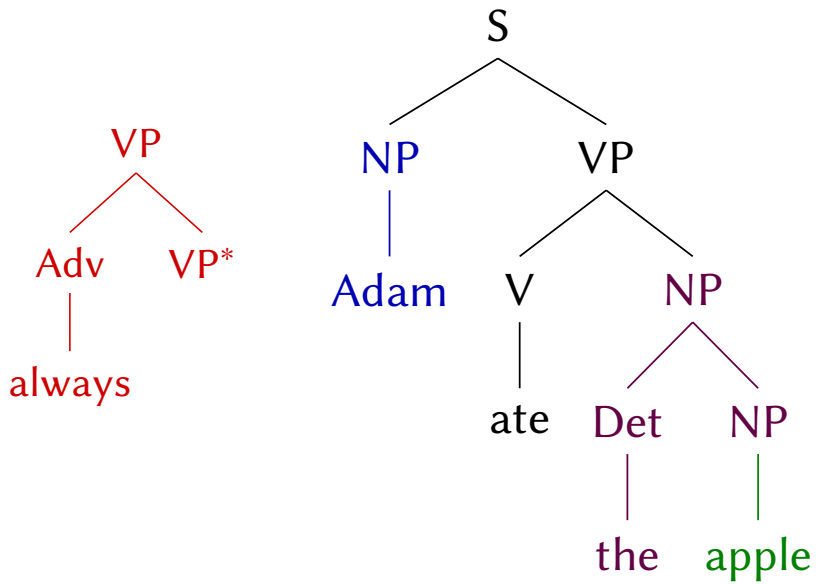
Derivation trees: example



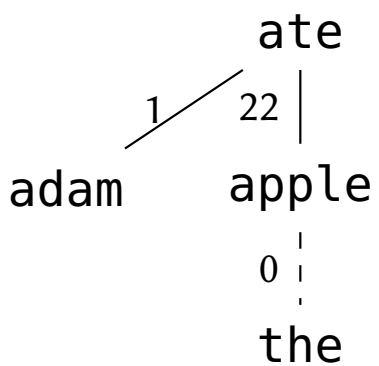
Derivation tree:



Derivation trees: example

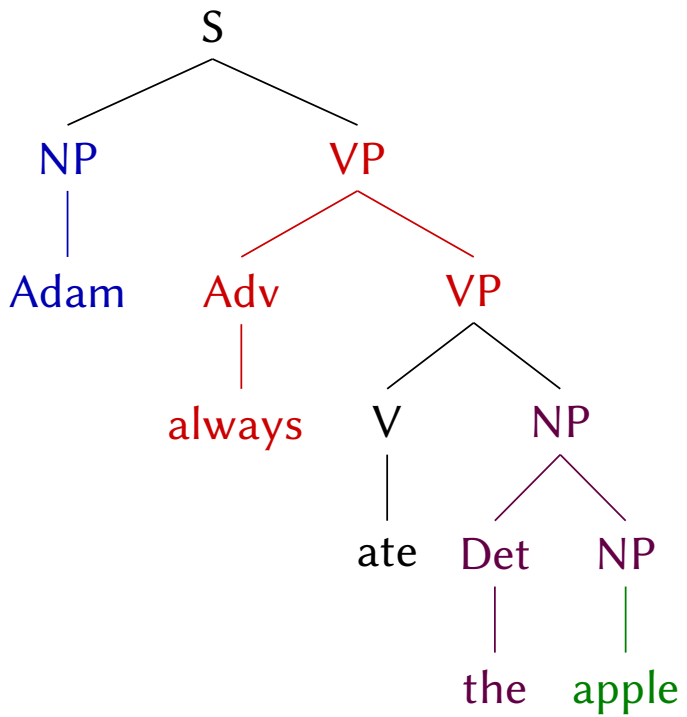


Derivation tree:

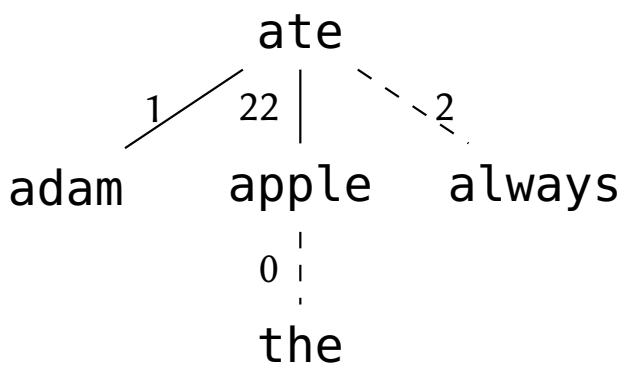


Derivation trees: example

Derived tree:



Derivation tree:



Outline of today's course

1. The derivation tree
2. Design principles for elementary trees
3. Sample derivations
   - NP and PP complements
   - Sentential complements and long-distance dependencies
   - Modifiers
4. Feature based TAG
5. Summary and outlook

Linguistic analyses with LTAG

What is an elementary tree, and what is its shape?

Linguistic analyses with LTAG

What is an elementary tree, and what is its shape?

elementary trees $\overset{?}{\Longleftarrow}$ syntactic/semantic properties of linguistic objects

Linguistic analyses with LTAG

What is an elementary tree, and what is its shape?

elementary trees $\overset{?}{\Longleftarrow}$ syntactic/semantic properties of linguistic objects

$\Rightarrow$ Syntactic design principles from Frank (2002):

- Lexicalization
- Fundamental TAG Hypothesis (FTH)
- Condition on Elementary Tree Minimality (CETM)
- θ-Criterion for TAG

Linguistic analyses with LTAG

What is an elementary tree, and what is its shape?

elementary trees $\overset{?}{\Longleftarrow}$ syntactic/semantic properties of linguistic objects

$\Rightarrow$ Syntactic design principles from Frank (2002):

- Lexicalization
- Fundamental TAG Hypothesis (FTH)
- Condition on Elementary Tree Minimality (CETM)
- θ-Criterion for TAG

$\Rightarrow$ Semantic design principles [Abeillé & Rambow (2000)]

Linguistic analyses with LTAG

What is an elementary tree, and what is its shape?

elementary trees $\overset{?}{\Longleftarrow}$ syntactic/semantic properties of linguistic objects

$\Rightarrow$ Syntactic design principles from Frank (2002):

- Lexicalization
- Fundamental TAG Hypothesis (FTH)
- Condition on Elementary Tree Minimality (CETM)
- θ-Criterion for TAG

$\Rightarrow$ Semantic design principles [Abeillé & Rambow (2000)]

$\Rightarrow$ Design principle of economy

Syntactic design principles (1): Lexicalization

Each elementary tree has at least one non-empty lexical item, its lexical **anchor**.

Syntactic design principles (1): Lexicalization

Each elementary tree has at least one non-empty lexical item, its lexical **anchor**.

⇒ All widely used grammar formalisms support some kind of lexicalization!

Syntactic design principles (1): Lexicalization

Each elementary tree has at least one non-empty lexical item, its lexical **anchor**.

⇒ All widely used grammar formalisms support some kind of lexicalization!

⇒ TAG → LTAG: Lexicalized Tree-Adjoining Grammar

Syntactic design principles (1): Lexicalization

Each elementary tree has at least one non-empty lexical item, its lexical **anchor**.

⇒ All widely used grammar formalisms support some kind of lexicalization!

⇒ TAG → LTAG: Lexicalized Tree-Adjoining Grammar

[Schabes & Joshi (1990); Joshi & Schabes (1991)]

Recall: reasons for lexicalization

- **Formal properties:** A finite lexicalized grammar provides finitely many analyses for each string (finitely ambiguous).
- **Linguistic properties:** Syntactic properties of lexical items can be accounted for more directly.
- **Parsing:** The search space during parsing can be delimited (grammar filtering).

Syntactic design principles (2): FTH

Fundamental TAG Hypothesis (FTH); [Frank (2002)]

Every syntactic dependency is expressed locally within an elementary tree.

Syntactic design principles (2): FTH

Fundamental TAG Hypothesis (FTH); [Frank (2002)]

Every syntactic dependency is expressed locally within an elementary tree.

“**syntactic dependency**”

- valency/subcategorization
- binding
- filler-gap constructions
- …

Syntactic design principles (2): FTH

Fundamental TAG Hypothesis (FTH); [Frank (2002)]

Every syntactic dependency is expressed locally within an elementary tree.

“syntactic dependency”

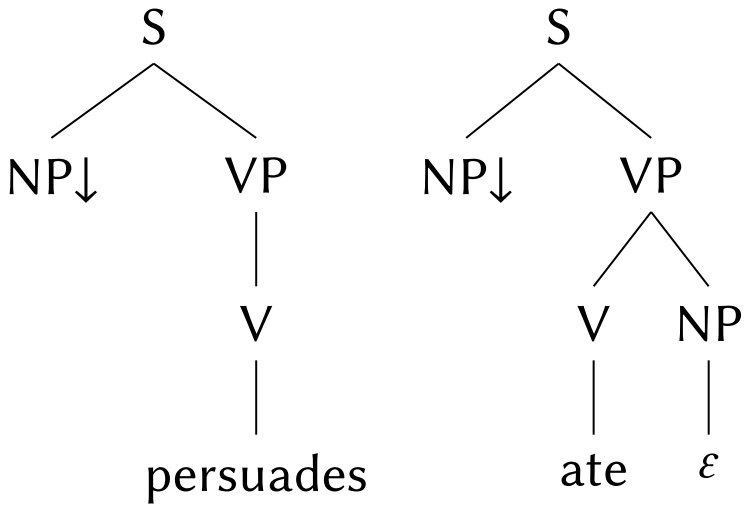
- valency/subcategorization
- binding
- filler-gap constructions
- …

“expressed within an elementary tree”

- terminal leaf (i.e. lexical anchor)
- nonterminal leaf (substitution node and footnode)
- marking an inner node for obligatory adjunction

Syntactic design principles (2): FTH

Examples of ill-formed elementary trees:



Complex primitives

Joshi (2004):

Complicate locally, simplify globally.

"[...] start with complex (more complicated) primitives, which capture directly some crucial linguistic properties and then introduce some general operations for composing these complex structures (primitive or derived). What is the nature of these complex primitives? In the conventional approach the primitive structures (or rules) are kept as simple as possible. This has the consequence that information (e.g., syntactic and semantic) about a lexical item (word) is distributed over more than one primitive structure. Therefore, the information associated with a lexical item is not captured locally, i.e., within the domain of a primitive structure."
[Joshi (2004)]

Syntactic design principles (3): CETM

Condition on Elementary Tree Minimality (CETM); ; [Frank (2002)]

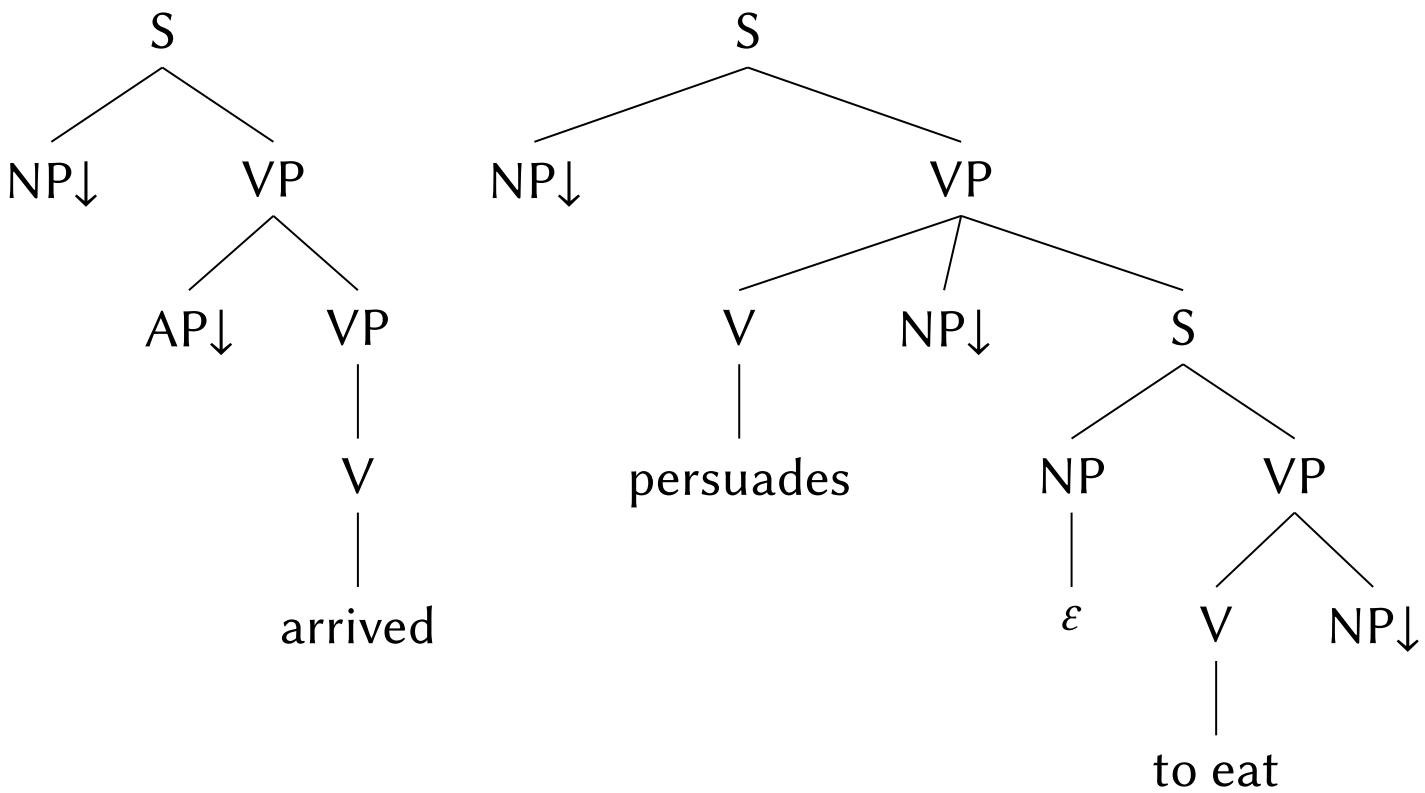
The syntactic heads in an elementary tree and their projections must form the extended projection of a single lexical head.

Syntactic design principles (3): CETM

Condition on Elementary Tree Minimality (CETM); ; [Frank (2002)]

The syntactic heads in an elementary tree and their projections must form the extended projection of a single lexical head.

Examples of ill-formed elementary trees:



Syntactic design principles (4): θ-Criterion for TAG

Thematic role (θ-role)

the semantic relationship of an argument with its predicate is expressed through the assignment of a role by the predicate to the argument. Different theta-roles have different labels, such as AGENT, THEME, PATIENT, GOAL, SOURCE, EXPERIENCER etc.

Syntactic design principles (4): θ-Criterion for TAG

Thematic role (θ-role)

the semantic relationship of an argument with its predicate is expressed through the assignment of a role by the predicate to the argument. Different theta-roles have different labels, such as Agent, Theme, Patient, Goal, Source, Experiencer etc.

- example: *Bart kicked the ball.*
  - *kicked* $\rightsquigarrow$ predicate
  - *Bart* $\rightsquigarrow$ Agent
  - *ball* $\rightsquigarrow$ Theme/Patient

Syntactic design principles (4): θ-Criterion for TAG

Thematic role (θ-role)

the semantic relationship of an argument with its predicate is expressed through the assignment of a role by the predicate to the argument. Different theta-roles have different labels, such as AGENT, THEME, PATIENT, GOAL, SOURCE, EXPERIENCER etc.

- example: *Bart kicked the ball.*
  - *kicked* $\rightsquigarrow$ predicate
  - *Bart* $\rightsquigarrow$ AGENT
  - *ball* $\rightsquigarrow$ THEME/PATIENT
- *The ball was kicked by Bart.*
  - *kicked* $\rightsquigarrow$ predicate
  - *Bart* $\rightsquigarrow$ AGENT
  - *ball* $\rightsquigarrow$ THEME/PATIENT

Syntactic design principles (4): θ-Criterion for TAG

θ-Criterion (TAG version)

a. If H is the lexical head of an elementary tree T, H assigns all of its θ-roles in T.

b. If A is a frontier non-terminal of elementary tree T, A must be assigned a θ-role in T.

[Frank (2002)]

$\Longrightarrow$ Valency/subcategorization is expressed only with nonterminal leaves!

Syntactic design principles (4): θ-Criterion for TAG

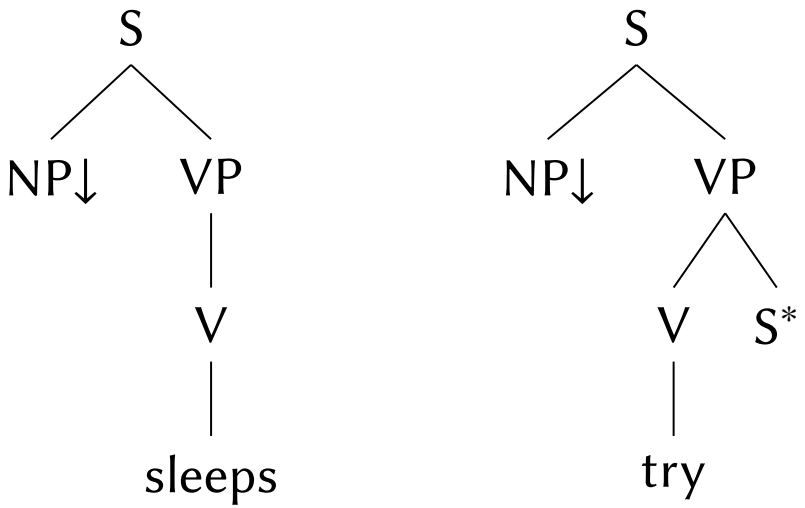
θ-Criterion (TAG version)

a. If H is the lexical head of an elementary tree T, H assigns all of its θ-roles in T.

b. If A is a frontier non-terminal of elementary tree T, A must be assigned a θ-role in T.

[Frank (2002)]

$\Longrightarrow$ Valency/subcategorization is expressed only with nonterminal leaves!



Further design principles

Semantic design principles

Predicate-argument co-occurrence:

Each elementary tree associated with a predicate contains a non-terminal leaf for each of its arguments.

Further design principles

Semantic design principles

Predicate-argument co-occurrence:

Each elementary tree associated with a predicate contains a non-terminal leaf for each of its arguments.

Semantic anchoring:

Elementary trees are not semantically void (to, that.)

Further design principles

Semantic design principles

Predicate-argument co-occurrence:

Each elementary tree associated with a predicate contains a non-terminal leaf for each of its arguments.

Semantic anchoring:

Elementary trees are not semantically void (to, that.)

Compositional principle:

An elementary tree corresponds to a single semantic unit.

Further design principles

Semantic design principles

Predicate-argument co-occurrence:

Each elementary tree associated with a predicate contains a non-terminal leaf for each of its arguments.

Semantic anchoring:

Elementary trees are not semantically void (to, that.)

Compositional principle:

An elementary tree corresponds to a single semantic unit.

Design principle of economy

The elementary trees are shaped in such a way, that the size of the elementary trees and the size of the grammar is minimal.

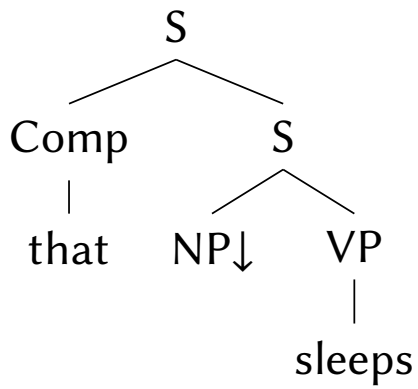
Modification and functional elements

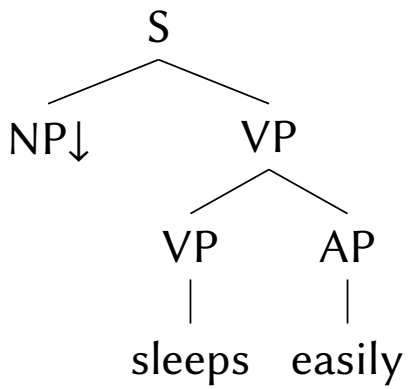
How to insert **modifiers** (e.g. *easily*) and **functional elements** (complementizers, determiners, do-auxiliaries, ...)?

Modification and functional elements

How to insert **modifiers** (e.g. *easily*) and **functional elements** (complementizers, determiners, do-auxiliaries, ...)?

- either as co-anchor in the elementary tree of the lexical item they are associated with

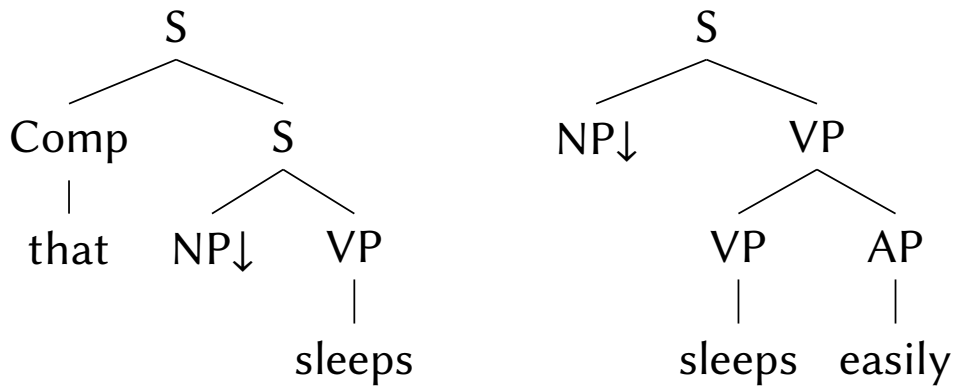




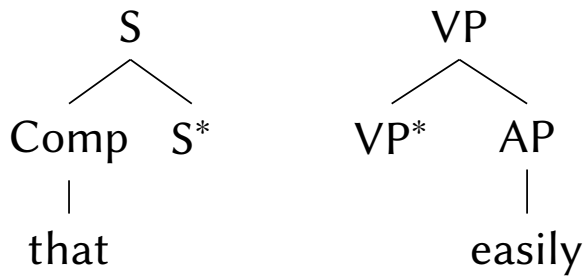
Modification and functional elements

How to insert **modifiers** (e.g. *easily*) and **functional elements** (complementizers, determiners, do-auxiliaries, ...)?

- either as co-anchor in the elementary tree of the lexical item they are associated with



- or by separate auxiliary trees (e.g., XTAG grammar)



⇒ Footnodes/Adjunctions indicate both complementation and modification.

⇒ Enhancement of the CETM: [see Abeillé & Rambow (2000)]

Outline of today's course

1. The derivation tree
2. Design principles for elementary trees
3. Sample derivations
   - NP and PP complements
   - Sentential complements and long-distance dependencies
   - Modifiers
4. Feature based TAG
5. Summary and outlook

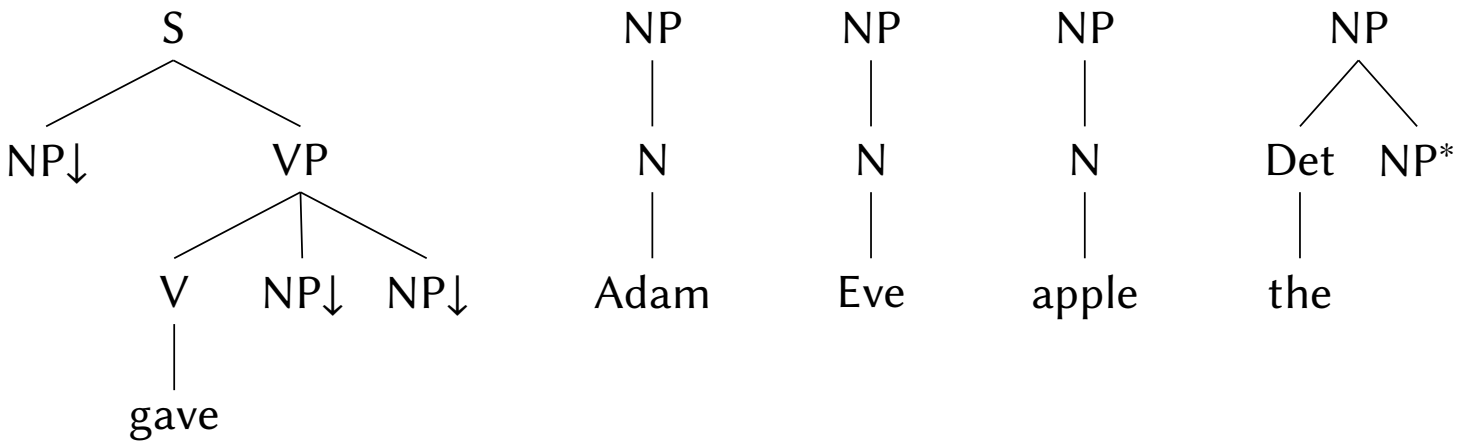
Sample derivations: NP and PP complements

(1) Adam gave Eve the apple.

Sample derivations: NP and PP complements

(1) Adam gave Eve the apple.

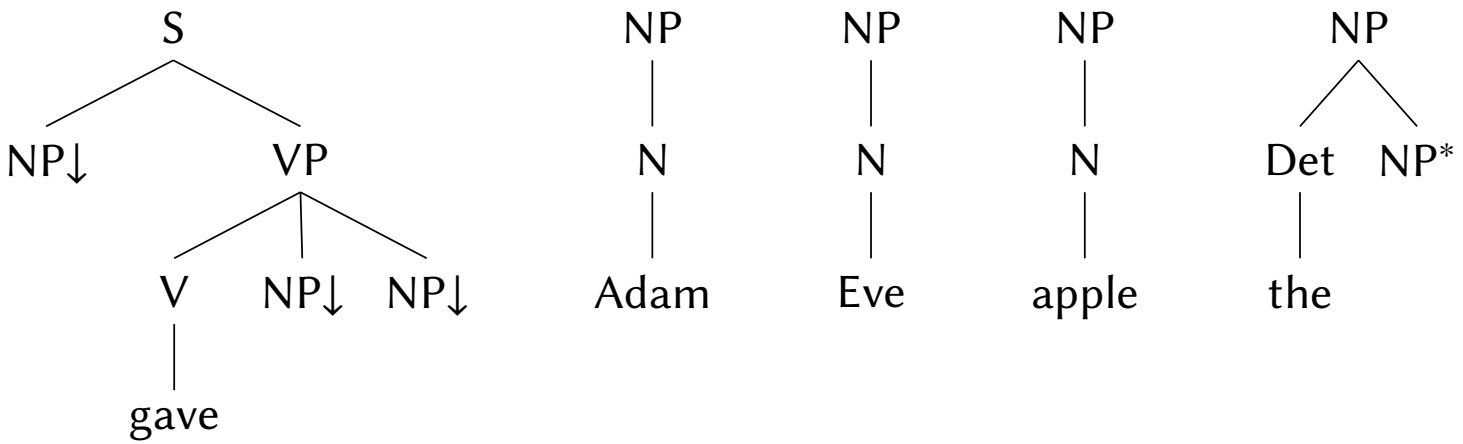
Elementary trees:



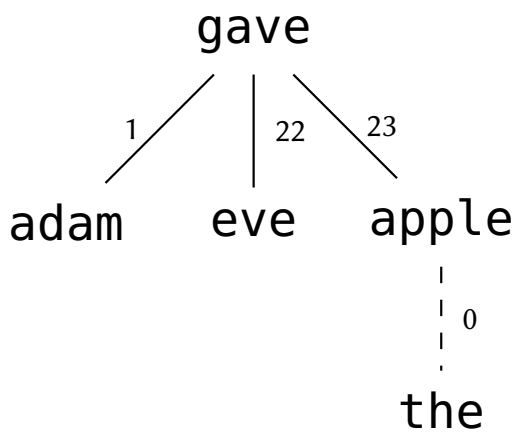
Sample derivations: NP and PP complements

(1) Adam gave Eve the apple.

Elementary trees:



Derivation tree:



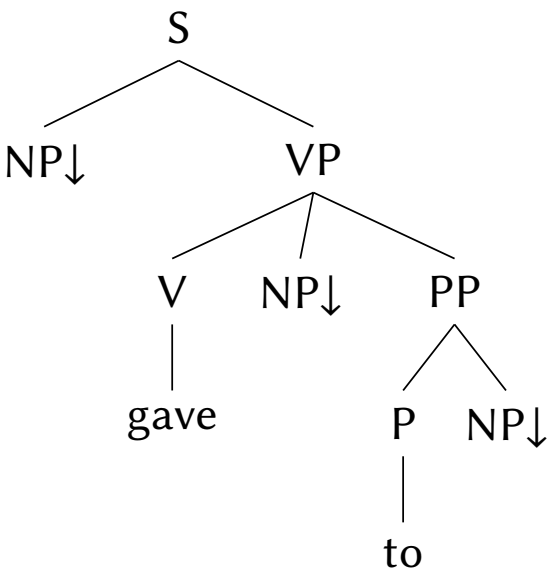
Sample derivations: NP and PP complements

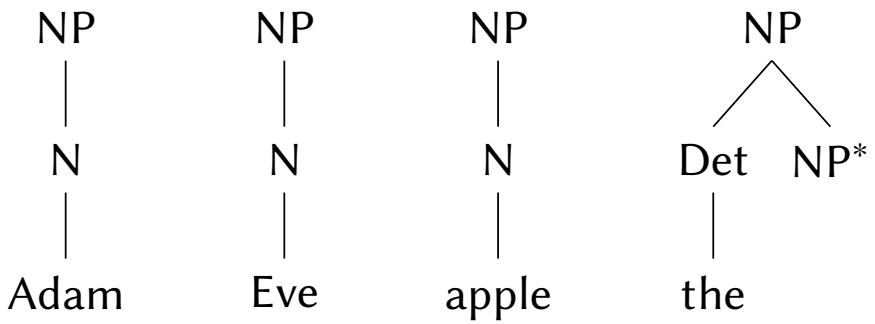
(2) Adam gave the apple to Eve.

Sample derivations: NP and PP complements

(2) Adam gave the apple to Eve.

Elementary trees:

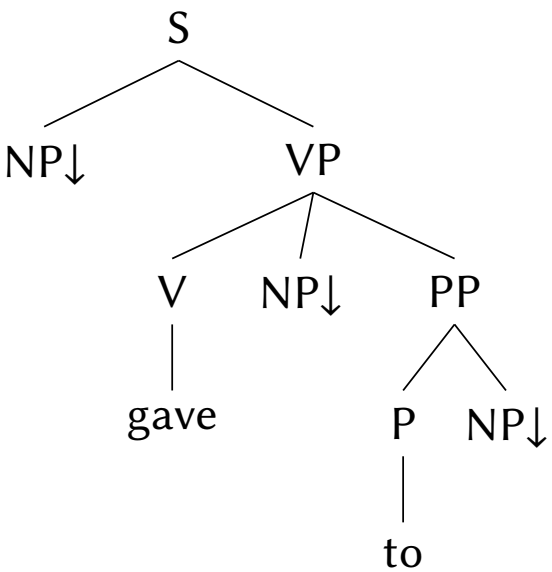


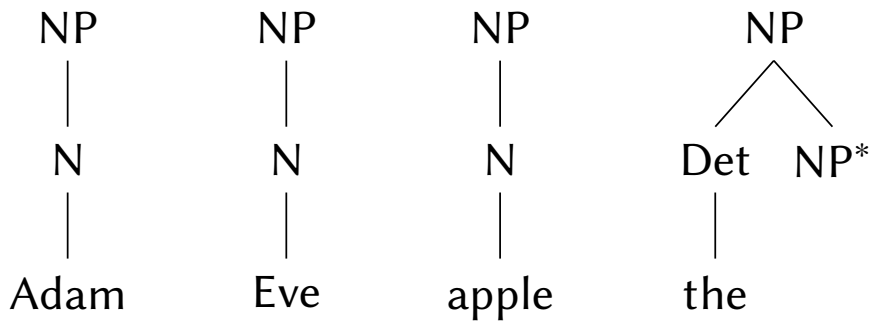


Sample derivations: NP and PP complements

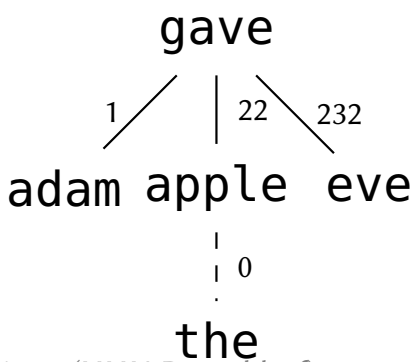
(2) Adam gave the apple to Eve.

Elementary trees:





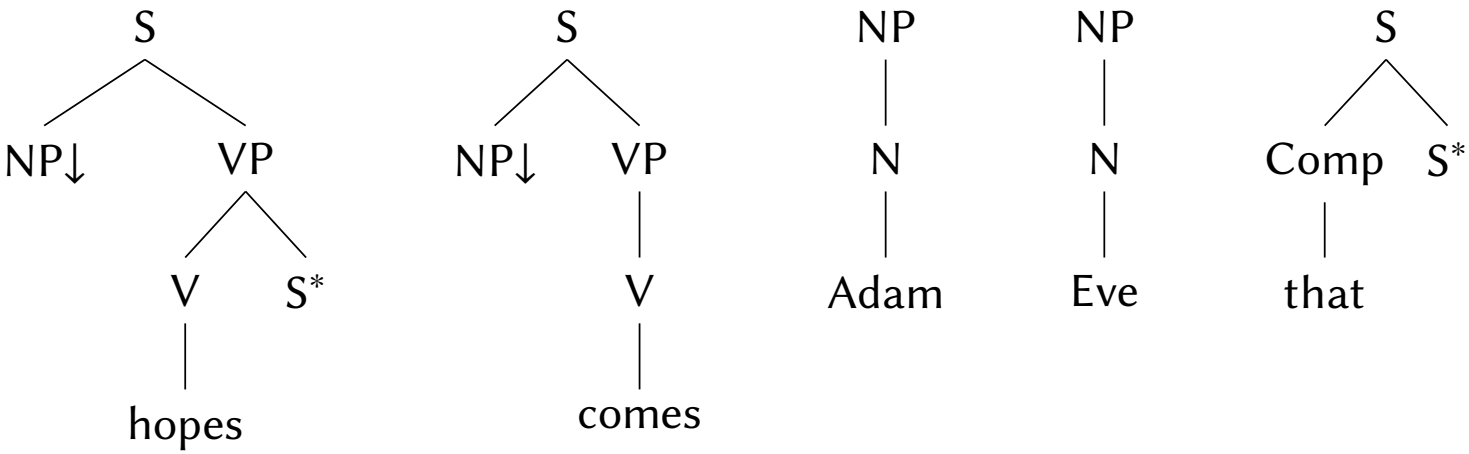
Derivation tree:



Sample derivations: Sentential complements

(3) Adam hopes that Eve comes.

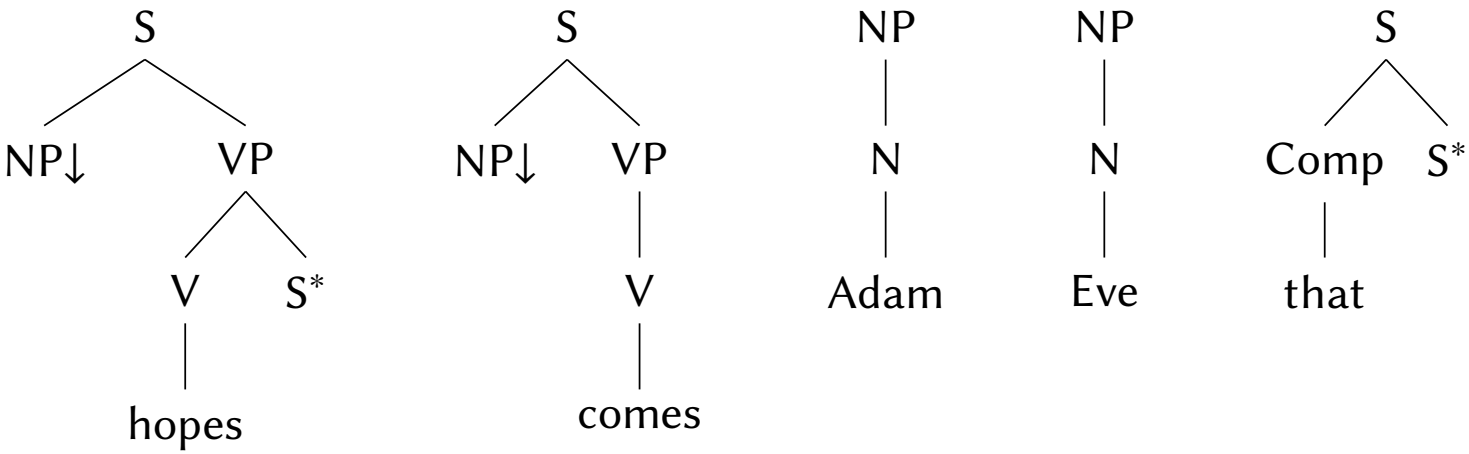
Elementary trees:



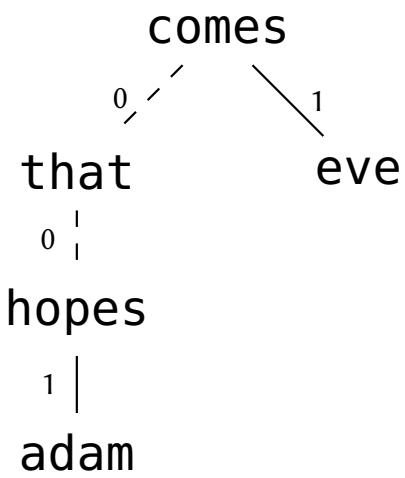
Sample derivations: Sentential complements

(3) Adam hopes that Eve comes.

Elementary trees:

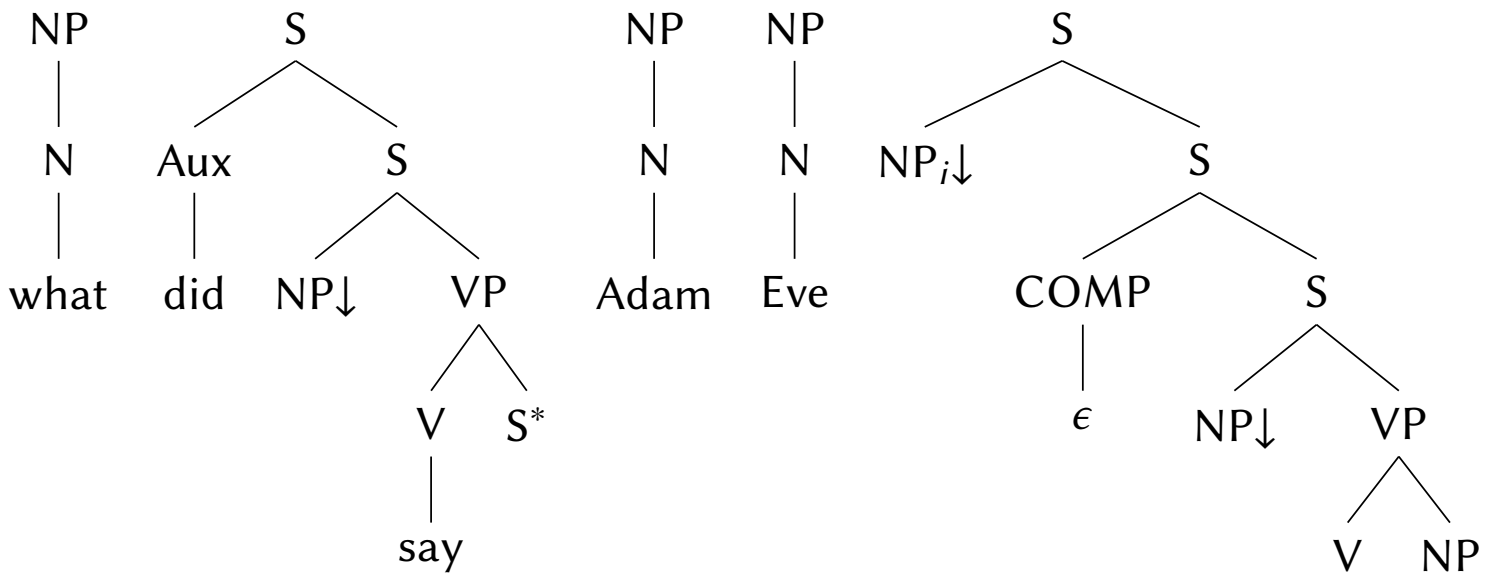


Derivation tree:

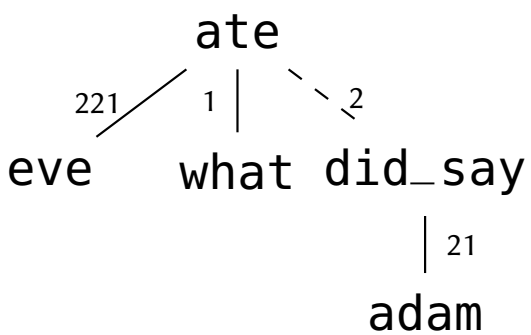


Sample derivations: long-distance dependency

(4) What$_i$ did Adam say (that) Eve ate $\_i$?



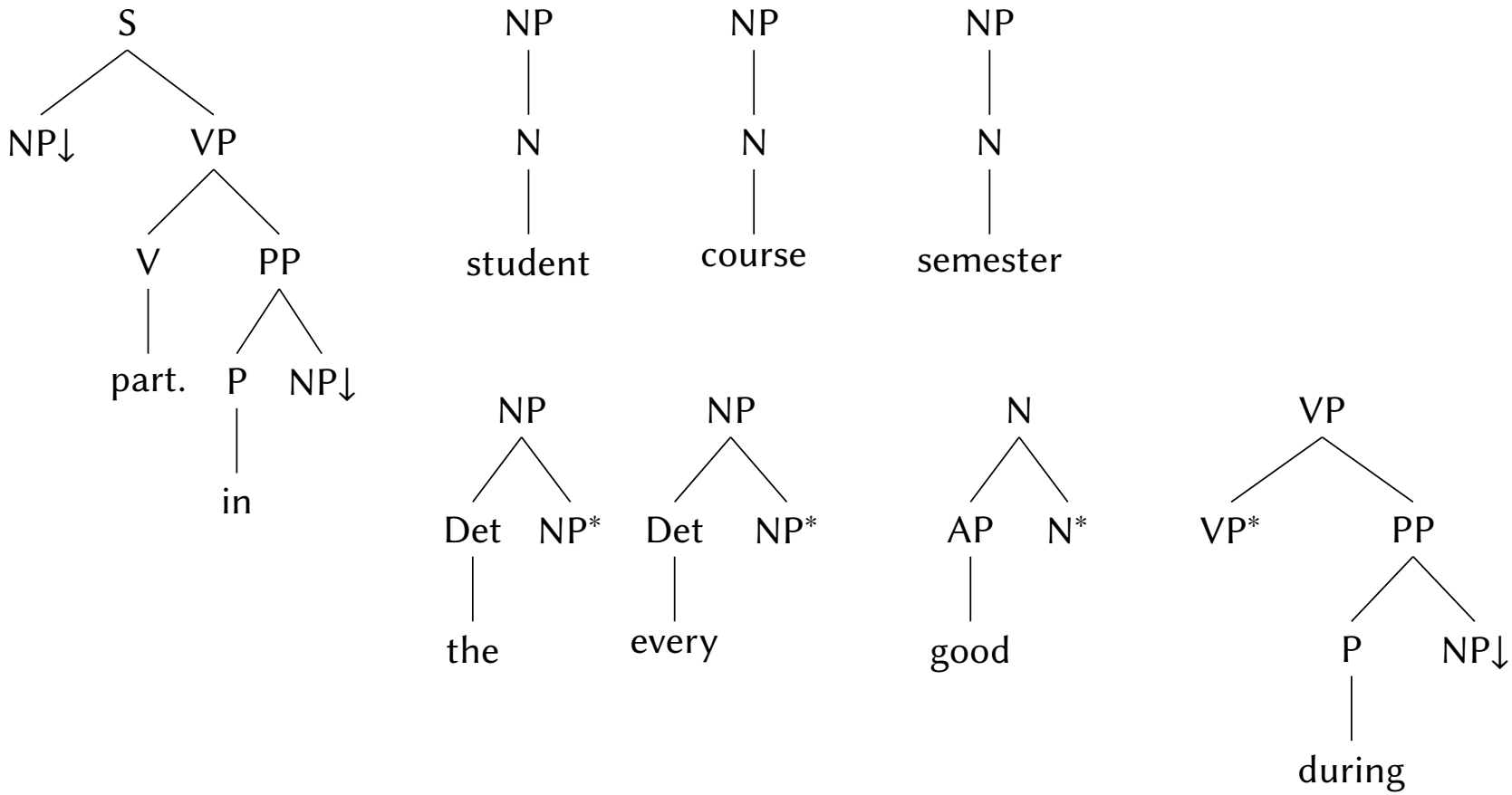
Derivation tree:



Sample derivations: Modifiers

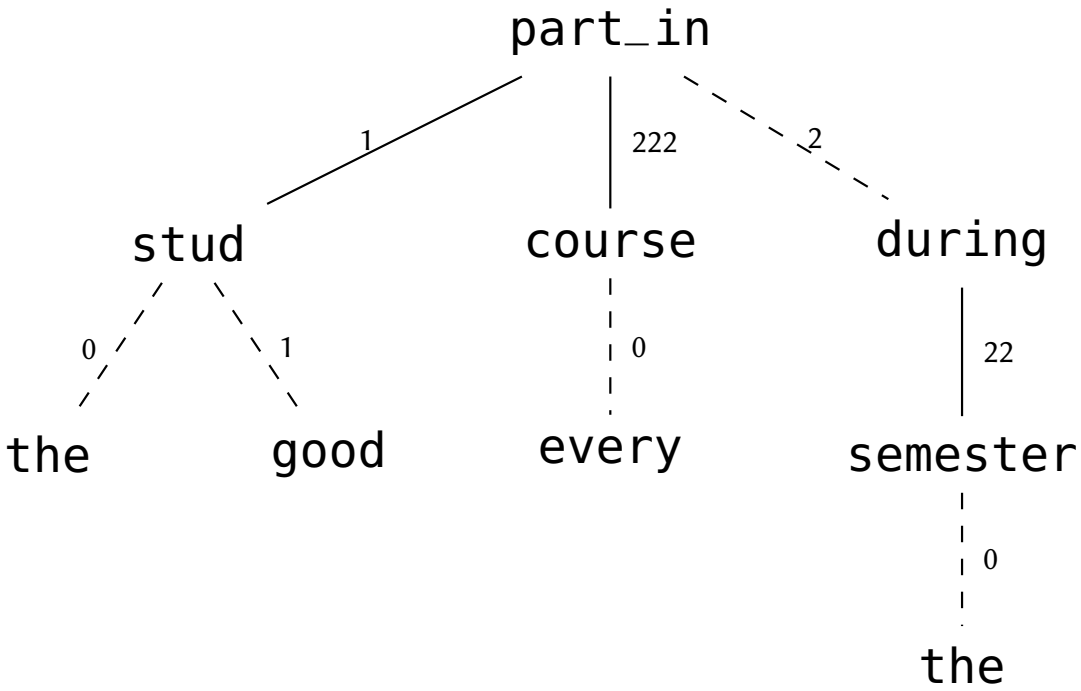
(5) The good student participated in every course during the semester.

Elementary trees:



Sample derivations: Modifiers

Derivation tree:



Outline of today's course

1. The derivation tree
2. Design principles for elementary trees
3. Sample derivations
   - NP and PP complements
   - Sentential complements and long-distance dependencies
   - Modifiers
4. Feature based TAG
5. Summary and outlook

Feature structures

Idea: Instead of atomic categorial symbols, feature structures are used as non-terminal nodes.

Feature structures

Idea: Instead of atomic categorial symbols, feature structures are used as non-terminal nodes.

Two reasons:

- generalizing agreement and case marking (via underspecification)
- modelling adjunction constraints (TAG specific)

$\Rightarrow$ smaller grammars that are easier to maintain

Feature structures

Idea: Instead of atomic categorial symbols, feature structures are used as non-terminal nodes.

Two reasons:

- generalizing agreement and case marking (via underspecification)
- modelling adjunction constraints (TAG specific)

$\Rightarrow$ smaller grammars that are easier to maintain

- case assignment:

  Joe saw her. / **Joe saw she.*
  Joe expected her to come. (ECM) / **Joe expected she to come.*

Feature structures

Idea: Instead of atomic categorial symbols, feature structures are used as non-terminal nodes.

Two reasons:

- generalizing agreement and case marking (via underspecification)
- modelling adjunction constraints (TAG specific)

$\Rightarrow$ smaller grammars that are easier to maintain

- case assignment:

  Joe saw her. / **Joe saw she.*
  Joe expected her to come. (ECM) / **Joe expected she to come.*

- person/number agreement:

  You sing. / **You sings.*
  She sings. / **She sing.*
  This woman sings. / **This woman sing.*
  These women sing. / **These women sings.*

Feature structures

Idea: Instead of atomic categorial symbols, feature structures are used as non-terminal nodes.

Two reasons:

- generalizing agreement and case marking (via underspecification)
- modelling adjunction constraints (TAG specific)

$\Rightarrow$ smaller grammars that are easier to maintain

- case assignment:

  Joe saw her. / **Joe saw she.*
  Joe expected her to come. (ECM) / **Joe expected she to come.*

- person/number agreement:

  You sing. / **You sings.*
  She sings. / **She sing.*
  This woman sings. / **This woman sing.*
  These women sing. / **These women sings.*

Features structures

- a list of features (e.g. CASE) and values (e.g. nom)

Features structures

- a list of features (e.g. CASE) and values (e.g. nom)
- feature structures are often represented as **attribute value matrices (AVM)**

Features structures

- a list of features (e.g. CASE) and values (e.g. nom)
- feature structures are often represented as **attribute value matrices (AVM)**

  sings:

$$\begin{bmatrix} \text{CAT} & \text{V} \\ \text{VFORM} & \text{finite} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PERS} & 3 \end{bmatrix} \end{bmatrix}$$

Features structures

- a list of features (e.g. CASE) and values (e.g. nom)
- feature structures are often represented as **attribute value matrices (AVM)**

  sings:

$$\begin{bmatrix} \text{CAT} & \text{V} \\ \text{VFORM} & \text{finite} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PERS} & 3 \end{bmatrix} \end{bmatrix}$$

- feature values:
  - atomic (e.g. for VFORM)
  - feature structures (e.g. for AGR)

Features structures

- a list of features (e.g. CASE) and values (e.g. nom)
- feature structures are often represented as **attribute value matrices (AVM)**

  sings:

$$\begin{bmatrix} \text{CAT} & \text{V} \\ \text{VFORM} & \text{finite} \\ \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PERS} & 3 \end{bmatrix} \end{bmatrix}$$

- feature values:
  - atomic (e.g. for VFORM)
  - feature structures (e.g. for AGR)
- combining constituents ⇒ **unify** their feature structures

Unification

- unification is a (partial) operation on feature structures
  - intuitively: the operation of combining two feature structures such that the new feature structure contains all the information of the original two, and nothing more

Unification

- unification is a (partial) operation on feature structures
  - intuitively: the operation of combining two feature structures such that the new feature structure contains all the information of the original two, and nothing more

e.g. $\begin{bmatrix} \textsc{cat} & \textsc{vp} \\ \textsc{agr} & \begin{bmatrix} \textsc{num} & \textsc{pl} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \textsc{cat} & \textsc{vp} \\ \textsc{agr} & \begin{bmatrix} \textsc{pers} & 3 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \textsc{cat} & \textsc{vp} \\ \textsc{agr} & \begin{bmatrix} \textsc{num} & \textsc{pl} \\ \textsc{pers} & 3 \end{bmatrix} \end{bmatrix}$

Unification

- unification is a (partial) operation on feature structures
  - intuitively: the operation of combining two feature structures such that the new feature structure contains all the information of the original two, and nothing more

e.g. $\begin{bmatrix} \textsc{cat} & \text{vp} \\ \textsc{agr} & \begin{bmatrix} \textsc{num} & \text{pl} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \textsc{cat} & \text{vp} \\ \textsc{agr} & \begin{bmatrix} \textsc{pers} & 3 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \textsc{cat} & \text{vp} \\ \textsc{agr} & \begin{bmatrix} \textsc{num} & \text{pl} \\ \textsc{pers} & 3 \end{bmatrix} \end{bmatrix}$

- partial operation $\Rightarrow$ unification can fail

e.g. $\begin{bmatrix} \textsc{cat} & \text{np} \\ \textsc{num} & \text{sg} \end{bmatrix} \sqcup \begin{bmatrix} \textsc{cat} & \text{np} \\ \textsc{num} & \text{pl} \end{bmatrix} = \text{FAIL}$

Unification

- unification is a (partial) operation on feature structures
  - intuitively: the operation of combining two feature structures such that the new feature structure contains all the information of the original two, and nothing more

e.g. $\begin{bmatrix} \textsc{cat} & \text{vp} \\ \textsc{agr} & \begin{bmatrix} \textsc{num} & \text{pl} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \textsc{cat} & \text{vp} \\ \textsc{agr} & \begin{bmatrix} \textsc{pers} & 3 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \textsc{cat} & \text{vp} \\ \textsc{agr} & \begin{bmatrix} \textsc{num} & \text{pl} \\ \textsc{pers} & 3 \end{bmatrix} \end{bmatrix}$

- partial operation $\Rightarrow$ unification can fail

e.g. $\begin{bmatrix} \textsc{cat} & \text{np} \\ \textsc{num} & \text{sg} \end{bmatrix} \sqcup \begin{bmatrix} \textsc{cat} & \text{np} \\ \textsc{num} & \text{pl} \end{bmatrix} = \text{FAIL}$

- underspecified feature values

e.g. $\begin{bmatrix} \textsc{cat} & \text{np} \\ \textsc{case} & \text{nom} \mid \text{acc} \end{bmatrix} \sqcup \begin{bmatrix} \textsc{cat} & \text{np} \\ \textsc{case} & \text{acc} \end{bmatrix} = \begin{bmatrix} \textsc{cat} & \text{np} \\ \textsc{case} & \text{acc} \end{bmatrix}$

Unification: definition

Unification ($F \sqcup G$)

The unification of two feature structures F and G is (if it exists) the smallest feature structure that is subsumed by both F and G.

Unification: definition

Unification ($F \sqcup G$)

The unification of two feature structures F and G is (if it exists) the smallest feature structure that is subsumed by both F and G.

That is, (if it exists) $F \sqcup G$ is the feature structure with the following three properties:

Unification: definition

Unification ($F \sqcup G$)

The unification of two feature structures F and G is (if it exists) the smallest feature structure that is subsumed by both F and G.

That is, (if it exists) $F \sqcup G$ is the feature structure with the following three properties:

(1) $F \sqsubseteq (F \sqcup G)$

Unification: definition

Unification ($F \sqcup G$)

The unification of two feature structures F and G is (if it exists) the smallest feature structure that is subsumed by both F and G.

That is, (if it exists) $F \sqcup G$ is the feature structure with the following three properties:

(1) $F \sqsubseteq (F \sqcup G)$

(2) $G \sqsubseteq (F \sqcup G)$

Unification: definition

Unification ($F \sqcup G$)

The unification of two feature structures F and G is (if it exists) the smallest feature structure that is subsumed by both F and G.

That is, (if it exists) $F \sqcup G$ is the feature structure with the following three properties:

(1) $F \sqsubseteq (F \sqcup G)$

(2) $G \sqsubseteq (F \sqcup G)$

(3) If H is a feature structure such that $F \sqsubseteq H$ and $G \sqsubseteq H$, then $(F \sqcup G) \sqsubseteq H$. If there is no smallest feature structure that is subsumed by both F and G, then we say that $F \sqcup G$ is undefined.

Unification: definition

Unification ($F \sqcup G$)

The unification of two feature structures F and G is (if it exists) the smallest feature structure that is subsumed by both F and G.

That is, (if it exists) $F \sqcup G$ is the feature structure with the following three properties:

(1) $F \sqsubseteq (F \sqcup G)$

(2) $G \sqsubseteq (F \sqcup G)$

(3) If H is a feature structure such that $F \sqsubseteq H$ and $G \sqsubseteq H$, then $(F \sqcup G) \sqsubseteq H$. If there is no smallest feature structure that is subsumed by both F and G, then we say that $F \sqcup G$ is undefined.

Subsumption ($F_1 \sqsubseteq F_2$)

A feature structure F_1 subsumes ($\sqsubseteq$) another feature structure F_2, iff all the information that is contained in F_1 is also contained in F_2.

Reentrancies

- the paths that both lead to the same node ⇒ to the same value

Reentrancies

- the paths that both lead to the same node ⇒ to the same value
  ⇒ hence, they share that value

Reentrancies

- the paths that both lead to the same node ⇒ to the same value ⇒ hence, they share that value
- this property of sharing value(s) is called **reentrancy**

Reentrancies

- the paths that both lead to the same node ⇒ to the same value ⇒ hence, they share that value
- this property of sharing value(s) is called **reentrancy**
- in AVMs: expressed by coindexing the shared values (boxed numbers)

Reentrancies

- the paths that both lead to the same node ⇒ to the same value ⇒ hence, they share that value
- this property of sharing value(s) is called **reentrancy**
- in AVMs: expressed by coindexing the shared values (boxed numbers)
- within feature structures:

$$\begin{bmatrix} \text{ATTR}_1 & \boxed{1} \\ \text{ATTR}_2 & \boxed{1} \end{bmatrix} \qquad \begin{bmatrix} \text{ATTR}_1 & \boxed{1}\text{val}_1 \\ \text{ATTR}_2 & \boxed{1} \end{bmatrix} \qquad \begin{bmatrix} \text{ATTR}_1 & \boxed{1}\begin{bmatrix} \text{ATTR}_2 & \boxed{1} \end{bmatrix} \end{bmatrix}$$

Reentrancies

FTAG uses acyclic reentrancies!

Reentrancies

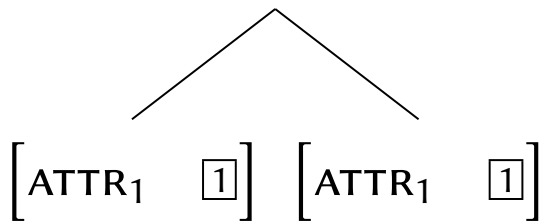
FTAG uses acyclic reentrancies!

- between feature structures (in a tree):

Reentrancies

FTAG uses acyclic reentrancies!

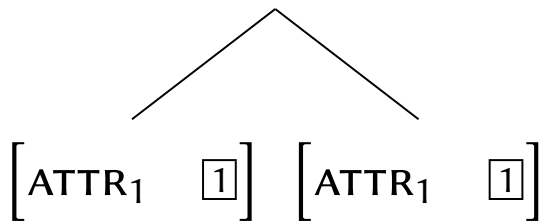
- between feature structures (in a tree):



Reentrancies

FTAG uses acyclic reentrancies!

- between feature structures (in a tree):



Note that

- feature structues in FTAG are untyped. (In contrast to frames, see tomorrow's course.)
- the feature geometry is such that there is only a finite number of possible feature structures
- therefore, FTAG can be shown to be strongly equivalent to TAG without feature structures

Unification: examples

- $$\begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{PERS} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PERS} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Unification: examples

- $$\begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{PERS} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PERS} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$
- $$\begin{bmatrix} \text{AGR} & \boxed{1} \begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \boxed{1} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{PERS} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \text{AGR} & \boxed{1} \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PERS} & 3 \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

Unification: examples

- $$\begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{PERS} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PERS} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$
- $$\begin{bmatrix} \text{AGR} & \boxed{1}\begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \boxed{1} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{PERS} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \text{AGR} & \boxed{1}\begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PERS} & 3 \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \boxed{1} \end{bmatrix} \end{bmatrix}$$
- for any feature structure F: $F \sqcup [\,] = [\,] \sqcup F = F$

Unification: examples

- $$\begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{PERS} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PERS} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$
- $$\begin{bmatrix} \text{AGR} & \boxed{1}\begin{bmatrix} \text{NUM} & \text{sg} \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \boxed{1} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{PERS} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \text{AGR} & \boxed{1}\begin{bmatrix} \text{NUM} & \text{sg} \\ \text{PERS} & 3 \end{bmatrix} \\ \text{SUBJ} & \begin{bmatrix} \text{AGR} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

- for any feature structure F: $F \sqcup [\,] = [\,] \sqcup F = F$
- the empty feature structure is the **identity element** for unification

TAG with feature structures

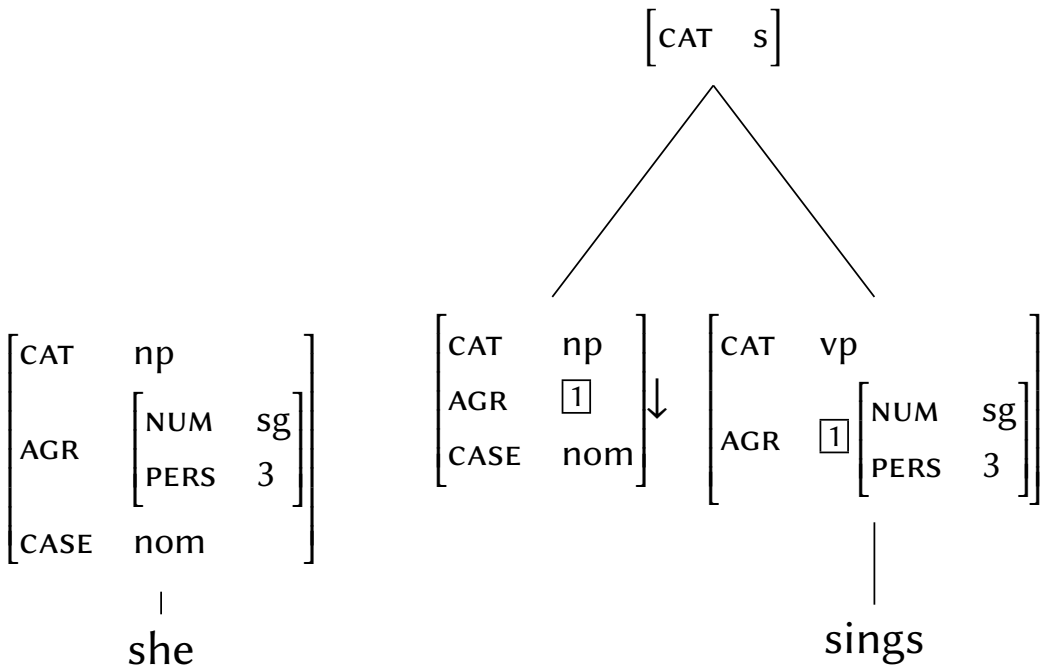
- **idea:** feature structures as non-terminal nodes

TAG with feature structures

- **idea:** feature structures as non-terminal nodes
- at substitution/adjunction the feature structures of the participating nodes are *unified*

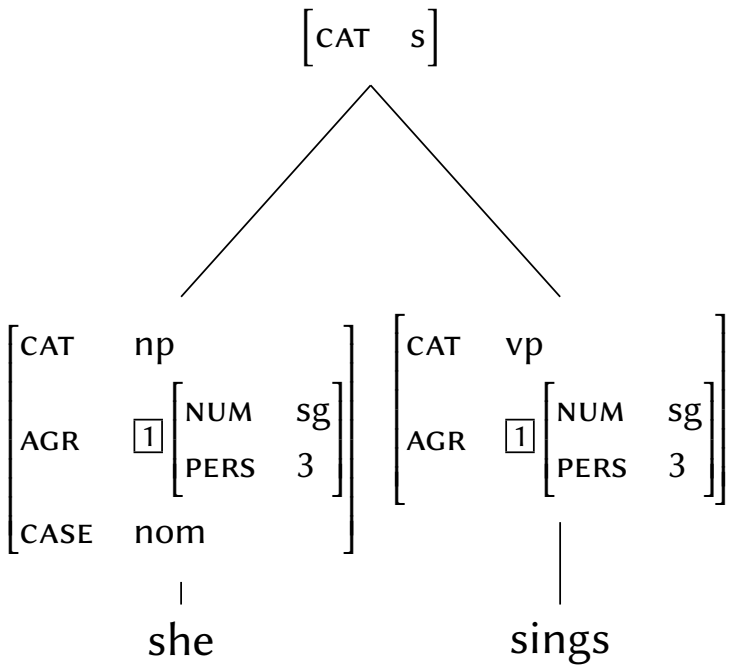
TAG with feature structures

- **idea:** feature structures as non-terminal nodes
- at substitution/adjunction the feature structures of the participating nodes are *unified*



TAG with feature structures

- **idea:** feature structures as non-terminal nodes
- at substitution/adjunction the feature structures of the participating nodes are *unified*



FTAG

Feature-structure based TAG (FTAG Vijay-Shanker & Joshi 1988):

FTAG

Feature-structure based TAG (FTAG Vijay-Shanker & Joshi 1988):

- annotate each substitution node with one and each other node with two feature structures

FTAG

Feature-structure based TAG (FTAG Vijay-Shanker & Joshi 1988):

- annotate each substitution node with one and each other node with two feature structures
- adjunction splits the feature structures
  - top features: the relation of the node to the tree above it
  - bottom features: the relation of the node to the tree below it

FTAG

Feature-structure based TAG (FTAG Vijay-Shanker & Joshi 1988):

- annotate each substitution node with one and each other node with two feature structures
- adjunction splits the feature structures
  - top features: the relation of the node to the tree above it
  - bottom features: the relation of the node to the tree below it

FTAG description of node η

1. The relation of η to its supertree is called feature structure t_η.
2. The relation of η to its descendants is called feature structure b_η.

FTAG

Feature-structure based TAG (FTAG Vijay-Shanker & Joshi 1988):

- annotate each substitution node with one and each other node with two feature structures
- adjunction splits the feature structures
  - top features: the relation of the node to the tree above it
  - bottom features: the relation of the node to the tree below it

FTAG description of node η

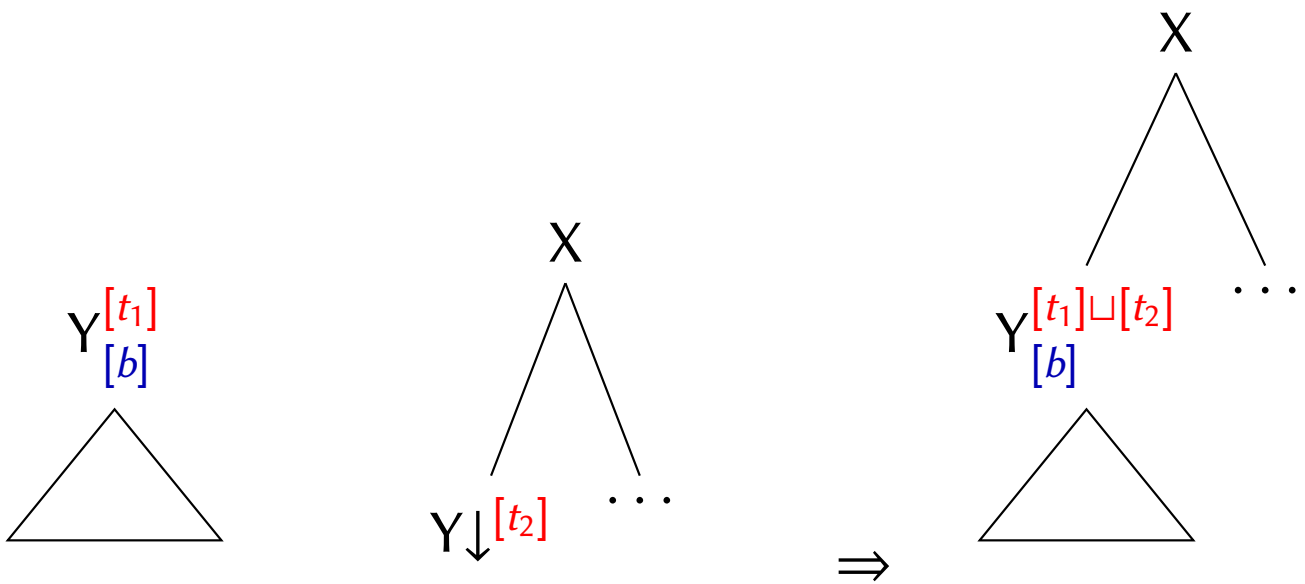
1. The relation of η to its supertree is called feature structure t_η.
2. The relation of η to its descendants is called feature structure b_η.

- at the final derived tree top and bottom features are unified for all nodes

FTAG: Substitution

Substitution in FTAG

The top features of the root of the tree to substitute unify with the top features of the substitution node.

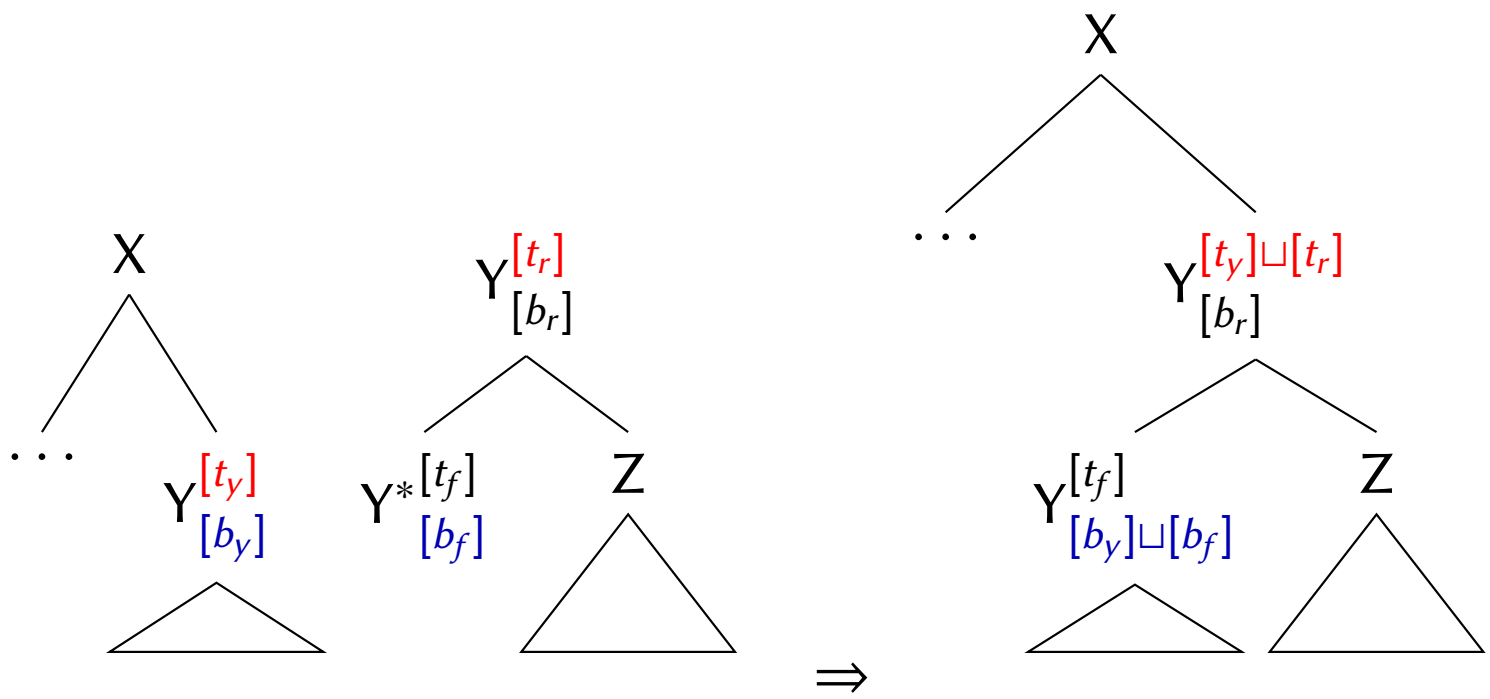


- substitution nodes (Y↓) have only top features

FTAG: Adjunction

Adjunction in FTAG

The top features of the root of the auxiliary tree unify with the top features of the adjunction node, and the bottom features of the footnode of the auxiliary tree unify with the bottom features of the adjunction node.



Adjunction constraints

Modeling adjunction constraints with features:

Adjunction constraints

Modeling adjunction constraints with features:

- SA: top and bottom are unifiable

$$\begin{bmatrix}\textsc{cat} & \text{vp}\end{bmatrix}$$
$$\begin{bmatrix}\textsc{cat} & \text{vp}\end{bmatrix}$$

Adjunction constraints

Modeling adjunction constraints with features:

- SA: top and bottom are unifiable

$$\begin{bmatrix} \textsc{cat} & \text{vp} \end{bmatrix}$$
$$\begin{bmatrix} \textsc{cat} & \text{vp} \end{bmatrix}$$

- OA + SA: feature mismatch between top and bottom

$$\begin{bmatrix} \textsc{cat} & \text{vp} \\ \textsc{mode} & \text{ind} \end{bmatrix}$$
$$\begin{bmatrix} \textsc{cat} & \text{vp} \\ \textsc{mode} & \text{ger} \end{bmatrix}$$

Adjunction constraints

Modeling adjunction constraints with features:

- SA: top and bottom are unifiable

$$\begin{bmatrix} \text{CAT} & \text{vp} \end{bmatrix}$$
$$\begin{bmatrix} \text{CAT} & \text{vp} \end{bmatrix}$$

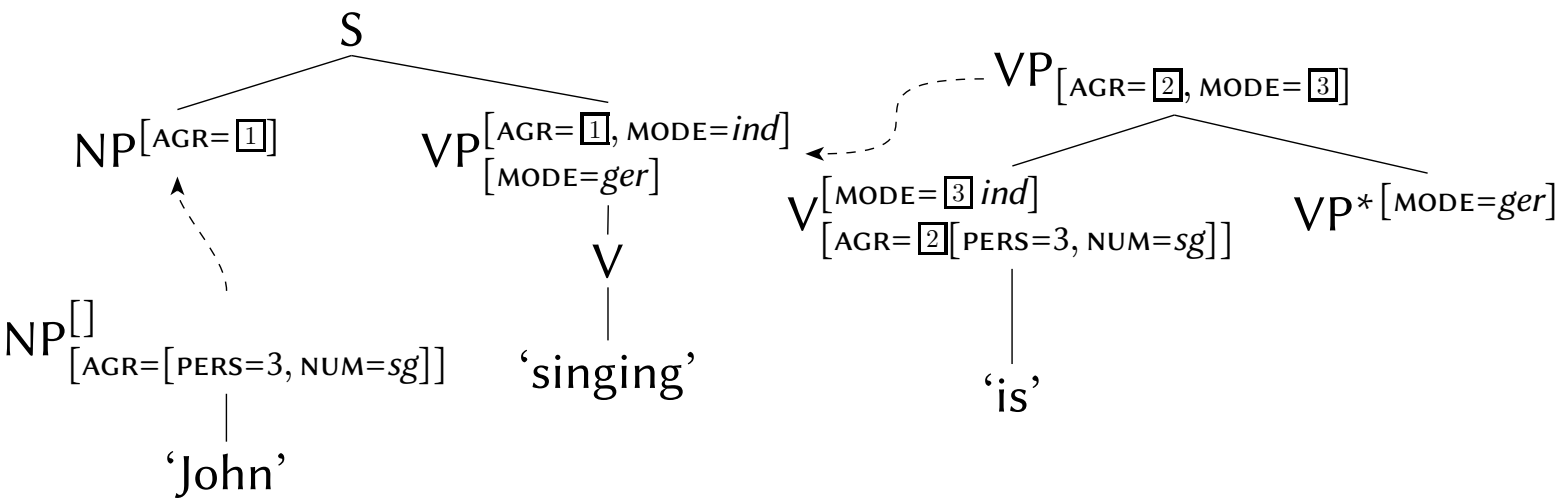
- OA + SA: feature mismatch between top and bottom

$$\begin{bmatrix} \text{CAT} & \text{vp} \\ \text{MODE} & \text{ind} \end{bmatrix}$$
$$\begin{bmatrix} \text{CAT} & \text{vp} \\ \text{MODE} & \text{ger} \end{bmatrix}$$

- NA: top and bottom are unifiable, but there is no auxiliary tree in the grammar that can be unified with them

FTAG example for OA

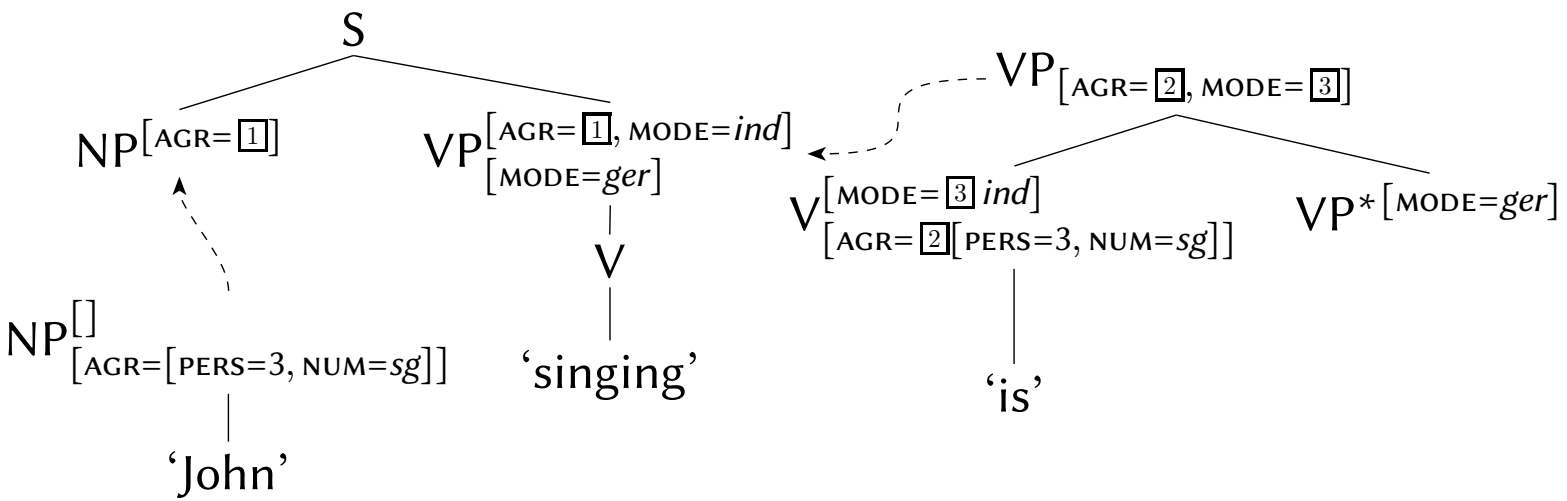
(6) John is singing.



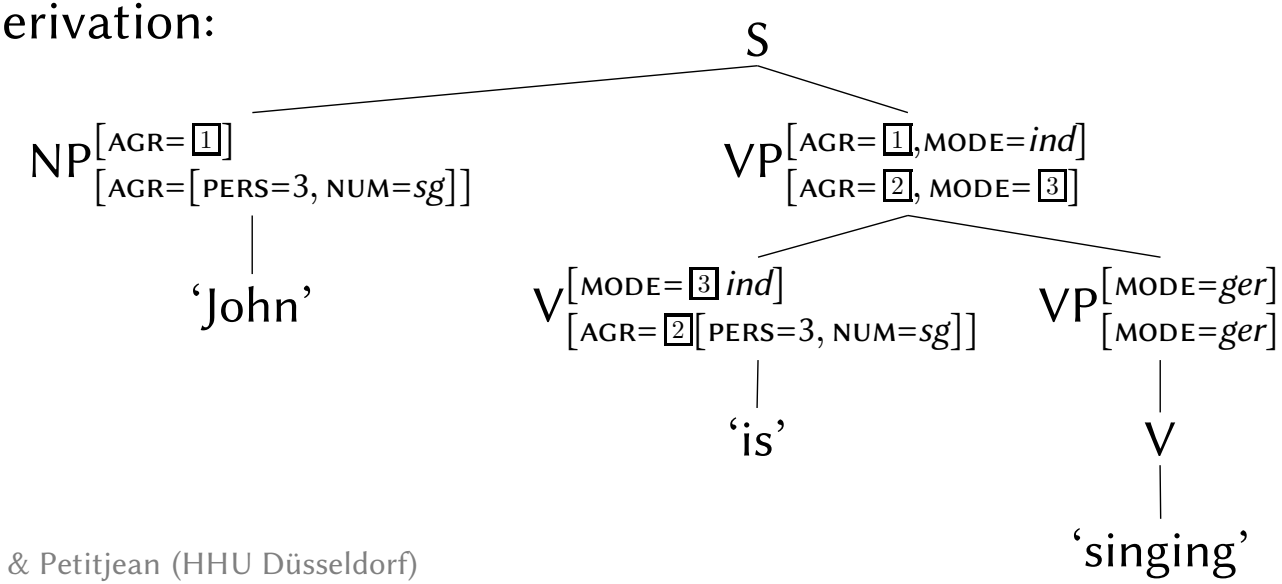
- The features are inspired by the XTAG grammar (XTAG Research Group 2001).
- The CAT feature is taken to be special, in particular it is usually the same in top and bottom. We therefore notate it as the main category of a node, outside the feature structures.

FTAG example for OA

(6) John is singing.

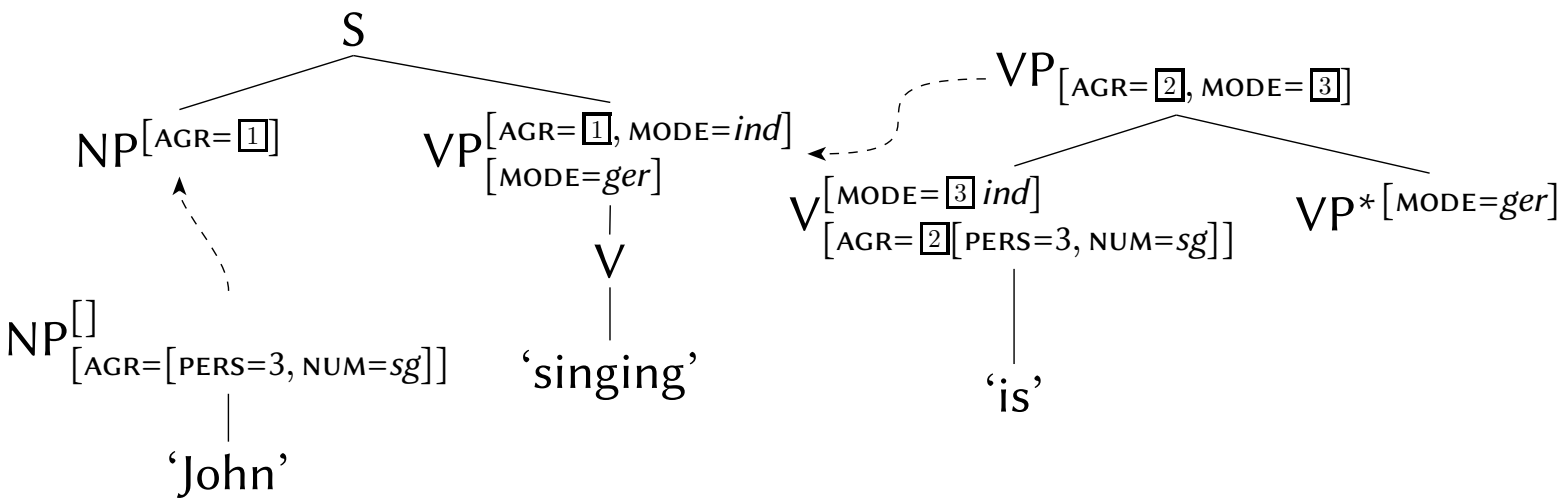


Result of derivation:

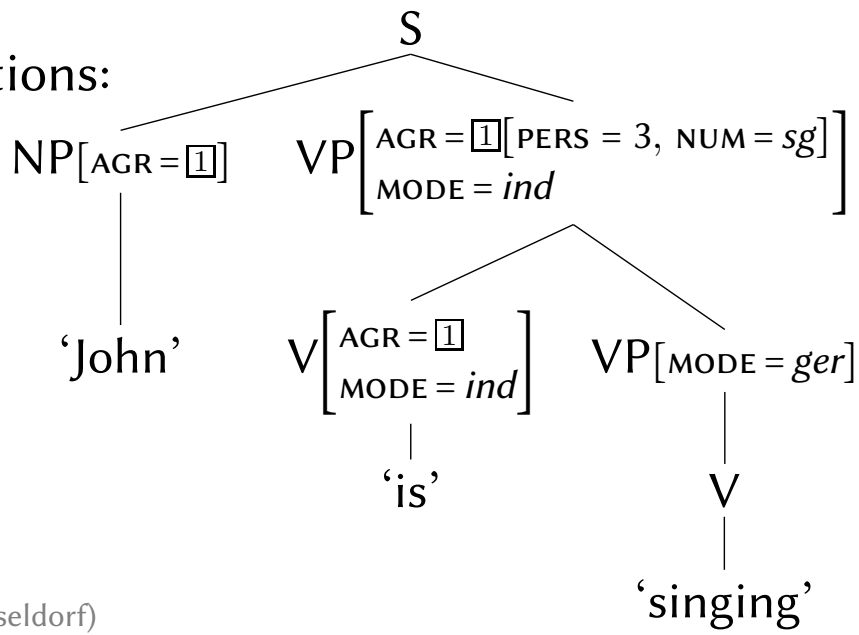


FTAG example for OA

(6) John is singing.



After top-bottom unifications:



Case assignment

- nouns carry the case, which is ‘checked’

Case assignment

- nouns carry the case, which is ‘checked’
- noun case is checked against the case value assigned by the verb during the unification

Case assignment

- nouns carry the case, which is ‘checked’
- noun case is checked against the case value assigned by the verb during the unification
- features of case-assignment:

Case assignment

- nouns carry the case, which is ‘checked’
- noun case is checked against the case value assigned by the verb during the unification
- features of case-assignment:
  - CASE with values: nom | acc | gen | none
    ⇒ Ns, NPs

Case assignment

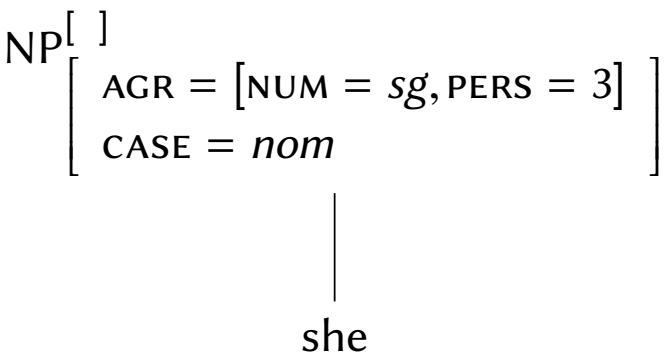
- nouns carry the case, which is 'checked'
- noun case is checked against the case value assigned by the verb during the unification
- features of case-assignment:
  - CASE with values: nom | acc | gen | none
    ⇒ Ns, NPs
  - ASSIGN-CASE with values: nom | acc | none
    ⇒ case assigners (prepositions, verbs) and S, VP, PP nodes that dominate them

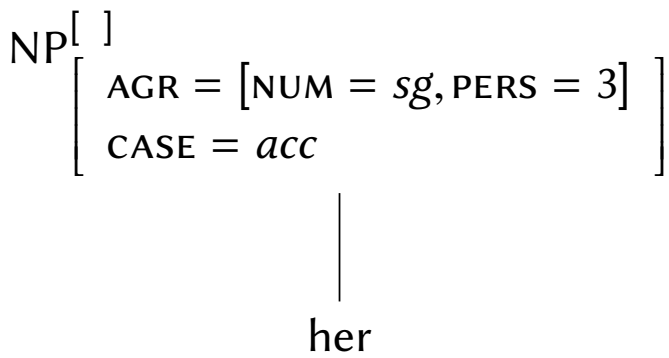
Case assignment

(7) a. she laughed
b. *her laughed

Case assignment

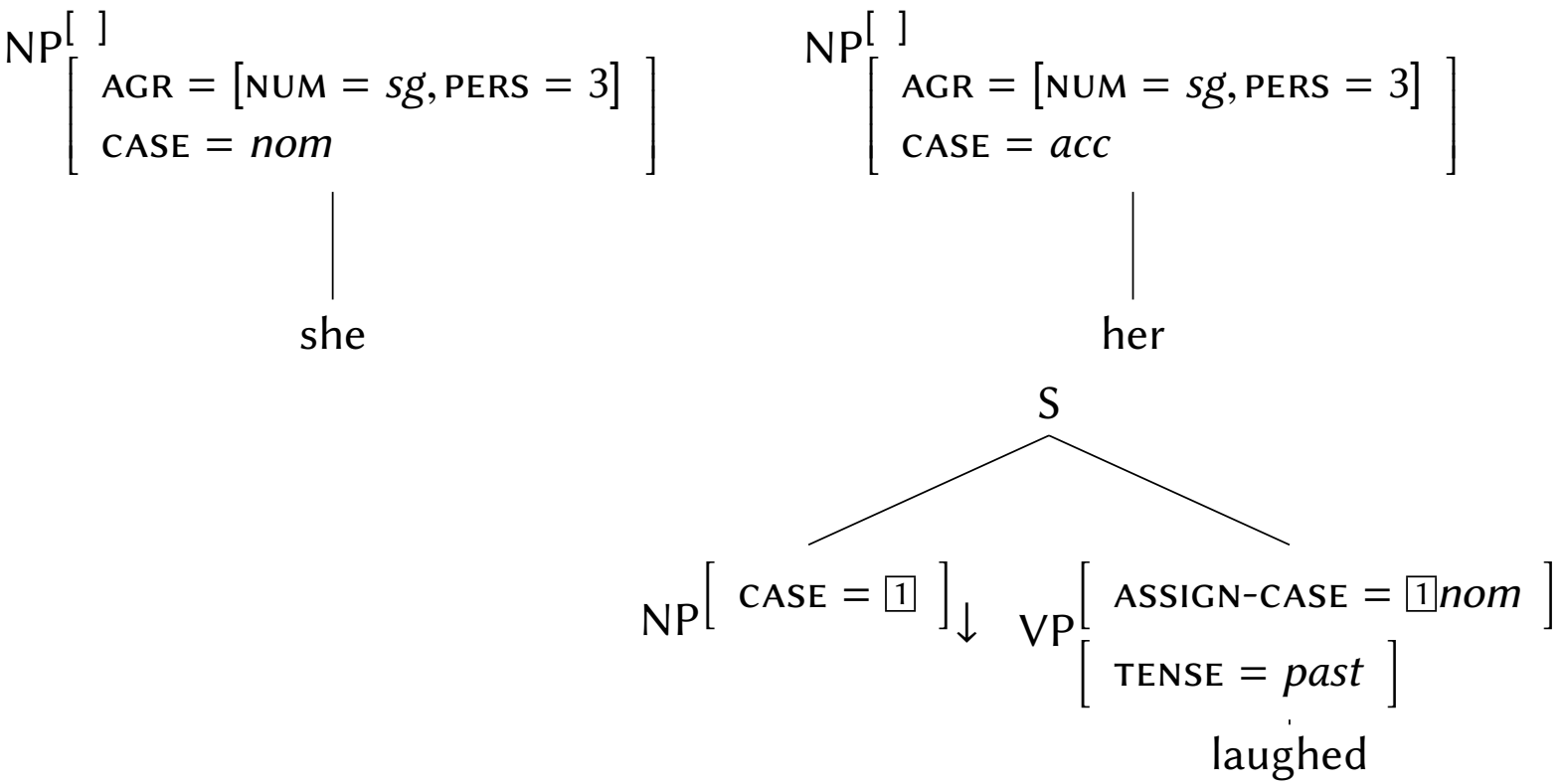
(7) a. she laughed
b. *her laughed





Case assignment

(7) a. she laughed
b. *her laughed



Outline of today's course

1. The derivation tree
2. Design principles for elementary trees
3. Sample derivations
   - NP and PP complements
   - Sentential complements and long-distance dependencies
   - Modifiers
4. Feature based TAG
5. Summary and outlook

Summary & outlook

Summary

- derivation tree vs. derived tree
- design principles: syntactic / semantic / economy
- example derivations: the base cases
- TAG + features structures

Summary & outlook

Summary

- derivation tree vs. derived tree
- design principles: syntactic / semantic / economy
- example derivations: the base cases
- TAG + features structures

Tomorrow

- further linguistic applications (syntax)
- in particular: extraction and long-distance dependencies
- grammar factorization
- the syntax-semantics interface

References

Abeillé, Anne & Owen Rambow. 2000. Tree adjoining grammar: An overview. In Anne Abeillé & Owen Rambow (eds.), **Tree Adjoining Grammars: Formalisms, linguistic analyses and processing** (CSLI Lecture Notes 107), 1–68. Stanford, CA: CSLI Publications.

Frank, Robert. 2002. **Phrase structure composition and syntactic dependencies**. Cambridge,MA: MIT Press.

Joshi, Aravind K. 2004. Starting with complex primitives pays off: complicate locally, simplify globally. **Cognitive Science** 28. 637–668.

Joshi, Aravind K. & Yves Schabes. 1991. Tree-Adjoining Grammars and lexicalized grammars. Tech. Rep. MS-CIS-91-22 Department of Computer and Information Science, University of Pennsylvania. `http://repository.upenn.edu/cis_reports/445/`.

Schabes, Yves & Aravind K. Joshi. 1990. Parsing with lexicalized tree adjoining grammar. Tech. Rep. MS-CIS-90-11 Department of Computer and Information Science, University of Pennsylvania. `http://repository.upenn.edu/cis_reports/542/`.

Vijay-Shanker, K. & Aravind K. Joshi. 1988. Feature structures based tree adjoining grammar. In **Proceedings of coling**, 714–719. Budapest.

XTAG Research Group. 2001. A Lexicalized Tree Adjoining Grammar for English. Tech. rep. Institute for Research in Cognitive Science, University of Pennsylvania Philadelphia, PA.