Constraint-based Computational Semantics:

LTAG and LRS

Frank Richter, Laura Kallmeyer
University of Tübingen

July 2006, Varna, Bulgaria

The Topic

- Model-theoretic semantics in constraint-based grammar frameworks
- Constraint-based techniques of semantic composition
  - properties
  - advantages
- Consequences for research in model-theoretic semantics

Structure of the Tutorial

Part I: Introduction to the Frameworks

- Constraint-based Model-theoretic Semantics
- Introduction to LTAG and LTAG Semantics
- Introduction to Lexical Resource Semantics (LRS)

Part II: Analyses

- Quantifier Scope
- Restrictions on Quantifier Scope
- Negative Concord
- Negative Polarity Items
- Conclusions

Constraint-based Model-theoretic Semantics

Constraint-based Model-theoretic Semantics (1)

We will discuss constraint-based semantics in 2 different but mathematically explicit frameworks. The semantic systems have important properties in common:

- Ty2 as the language of semantic representations
- Semantic composition *not* based on the lambda calculus
- Feature logical specification of semantic composition
- Application of underspecification techniques
- Some fundamental notions about semantic representations in syntactic structures are very similar

These common properties make the 2 systems comparable in a meaningful way.

Constraint-based Model-theoretic Semantics (2)

The 2 systems also exhibit important differences. They can be traced to the fundamentally different grammar architectures of LTAG and HPSG:

LTAG

- Based on Context free grammars (CFGs)
- More precisely: Mild extension of CFGs
- The categories of the nodes are non-atomic: Described by statements in a (very weak) feature logic
- The feature logic will be important for LTAG semantics

HPSG

- Model-theoretic grammar framework: Grammars as systems of constraints with a logical interpretation
- LRS defined alongside syntax in the same system

Lexicalized Tree Adjoining Grammars:

Syntax and Semantics

Overview

1. Motivation for LTAG
2. Tree Adjoining Grammars (TAG)
3. LTAG and natural languages
4. LTAG Semantics with Semantic Unification

Motivation for LTAG

1. CFG and natural languages
2. Lexicalizing CFGs

CFG and natural languages (1)

Idea of mildly context-sensitive grammar formalisms: keep the grammar formalism as simple as possible in terms of generative power, i.e., just as complex as needed, nothing more.

Question: is CFG enough?

Answer: No.

Example: cross-serial dependencies in Dutch and in Swiss German cannot be adequately described within CFG.

(1)

... dat Wim Jan Marie de kinderen zag helpen leren zwemmen

... that Wim Jan Marie the children saw help teach swim

'... that Wim saw Jan help Marie teach the children to swim'

CFG and natural languages (2)

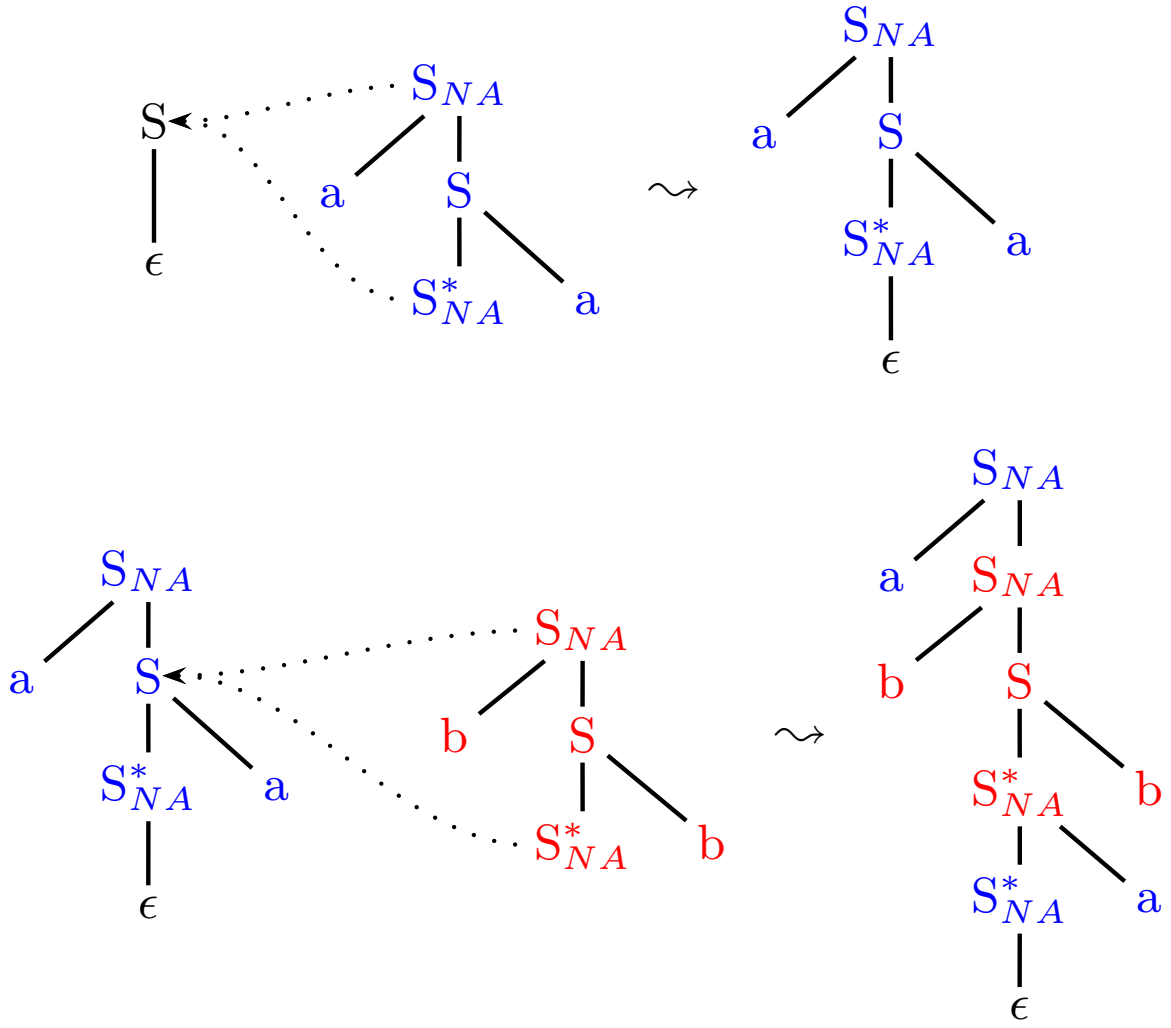
A formalism that can generate cross-serial dependencies must be able to generate the copy language $\{ww \mid w \in \{a, b\}^*\}$.

The copy language is not context-free.

Idea of Tree Adjoining Grammars (TAG): replacing not only leaves with new trees (substitution) as in CFG but replacing also internal nodes with new trees (adjunction).

CFG and natural languages (3)

Example: TAG derivation of *abab*:



Lexicalizing CFGs (1)

In a lexicalized grammar, each element of the grammar contains at least one lexical item (terminal symbol), and the operation used to put these elements together is non-erasing.

Lexicalized grammars are

- computationally interesting since in a lexicalized grammar the number of analyses for a sentence is finite (if the grammar is finite of course).
- linguistically interesting: each lexical item comes with the possibility of certain partial syntactic constructions, therefore one would like to associate it to a set of substructures.

Lexicalizing CFGs (2)

Question: can CFGs be lexicalized such that the set of trees remains the same (strong lexicalization)?

Answer: No. Only weak lexicalization (same string language) possible for CFGs (Greibach Normal Form).

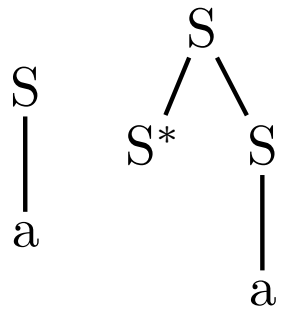
Example: the CFG $S \to SS$, $S \to a$ cannot be strongly lexicalized.

Problem: minimal length of a path from root to a leaf can be arbitrarily large.

But: the distance between two nodes on the same path cannot increase in CFG.

Lexicalizing CFGs (3)

Strongly equivalent LTAG:



LTAG

- strongly lexicalize CFGs, and
- are closed under lexicalization

Tree Adjoining Grammars (TAG)

1. Adjunction and substitution
2. Adjunction constraints
3. Derivation trees

Adjunction and substitution (1)

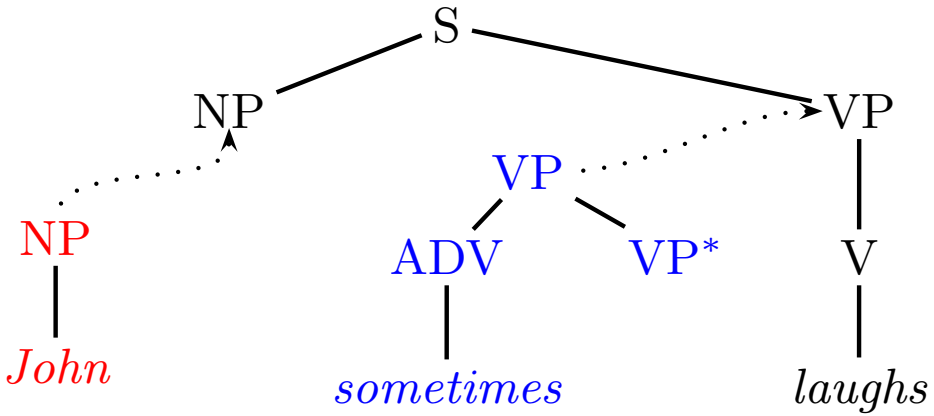
Tree Adjoining Grammars (TAG): Tree-rewriting system: set of *elementary* trees with two operations:

- adjunction: replacing an internal node with a new tree.
  The new tree is an auxiliary tree and has a special leaf, the foot node.
- substitution: replacing a leaf with a new tree.
  The new tree is an initial tree

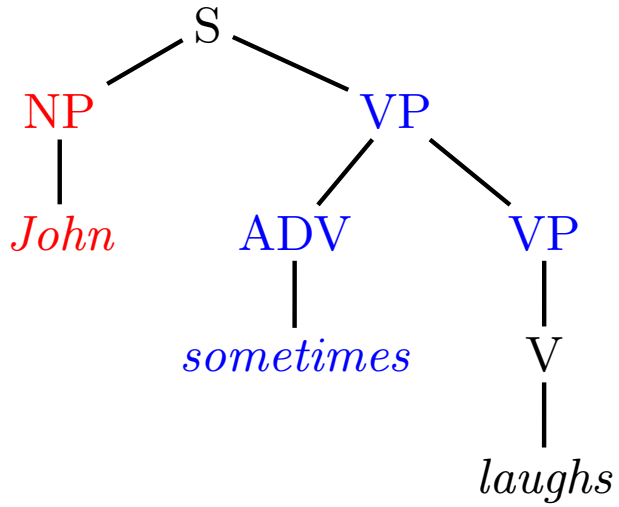
A good LTAG introduction: Joshi & Schabes (1997)

Adjunction and substitution (2)

(2) John sometimes laughs



derived tree:

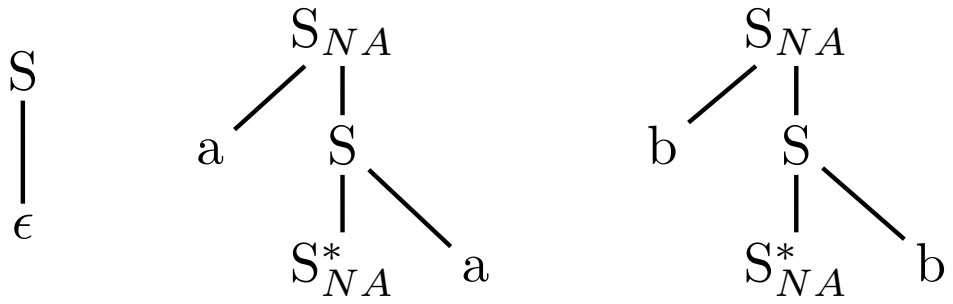


Adjunction constraints (1)

Additionally, adjunction constraints specify for each node

1. whether adjunction is mandatory and
2. which trees can be adjoined.

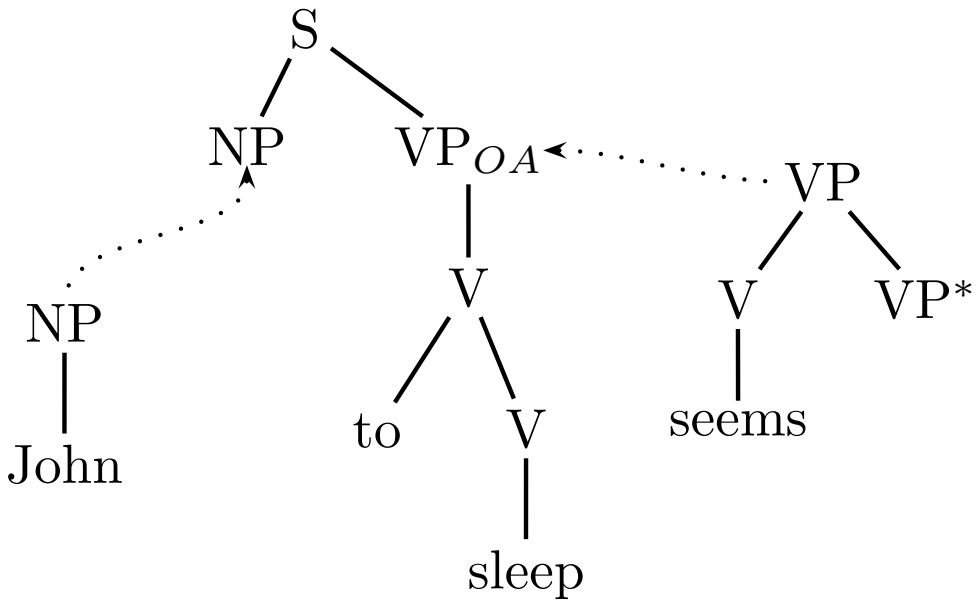
Example: TAG for the copy language:



Adjunction constraints (2)

Example:

(3) John seems to sleep



Derivation trees (1)

TAG derivations are described by derivation trees:

For each derivation in a TAG there is a corresponding derivation tree. This tree contains

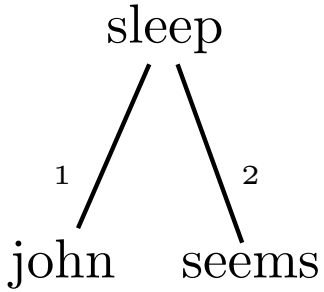
- nodes for all elementary trees used in the derivation, and
- edges for all adjunctions and substitutions performed throughout the derivation.

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

Derivation trees (2)

Example:

derivation tree for the derivation of (3) *John seems to sleep*



Derivation trees

- are context-free, and
- uniquely determine the derived tree.

⇒ TAG is a linear context-free rewriting system, LCFRS

LTAG and natural languages

1. The nature of elementary trees
2. Constituency and dependency
3. Feature-Structure Based TAG (FTAG)

Elementary trees (1)

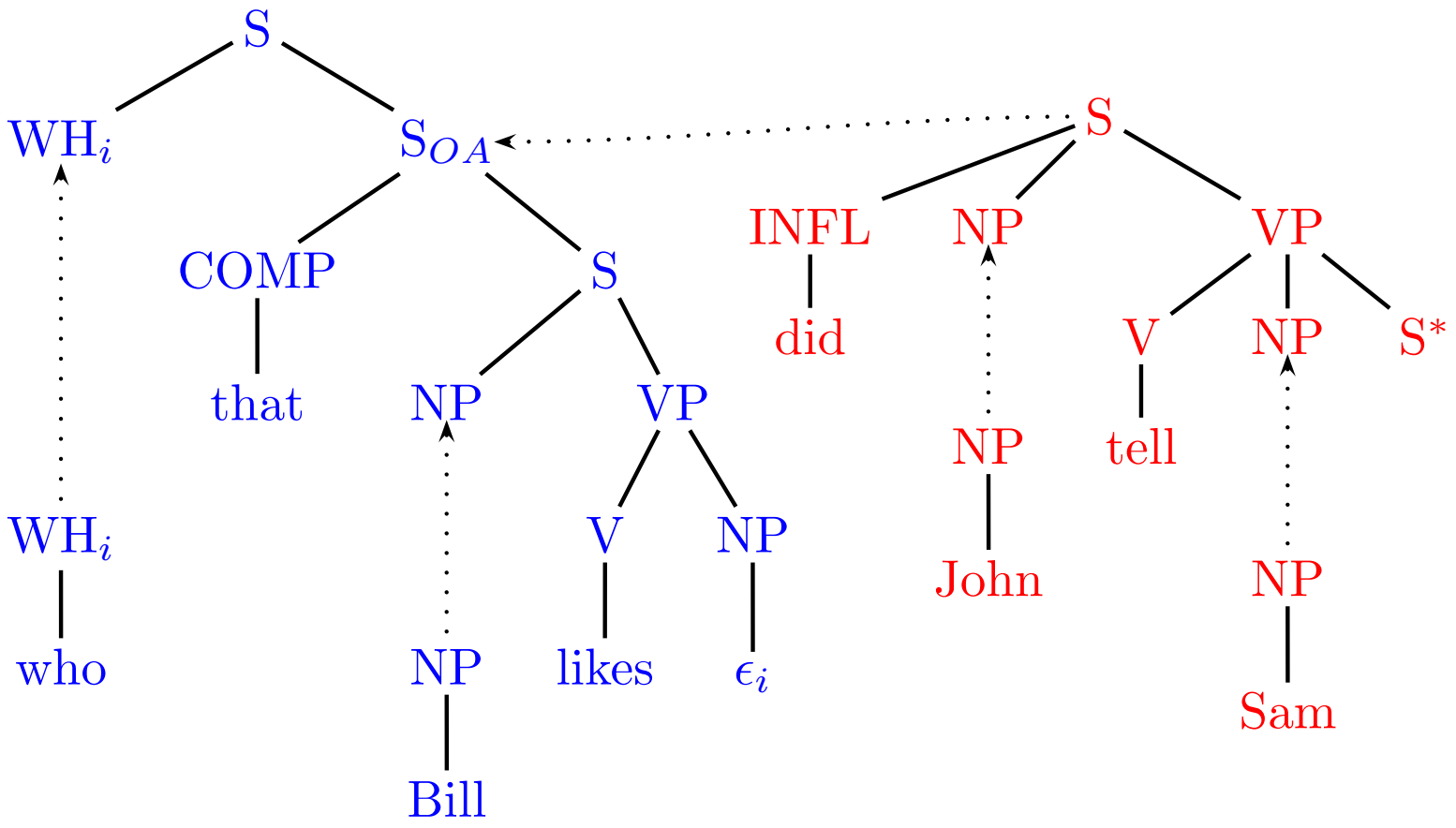
Important features of LTAG:

- Grammar is lexicalized
- Recursive parts are put into separate elementary trees that can be adjoined (Factoring of recursion, FR)
- Elementary trees can be arbitrarily large, in particular (because of FR) they can contain elements that are far apart in the final derived tree (Extended domain of locality)

Elementary trees (2)

(4) a. who$_i$ did John tell Sam that Bill likes t_i

b. who$_i$ did John tell Sam that Mary said that Bill likes t_i



Elementary trees (3)

Elementary trees are extended projections of lexical items (Frank 2002). Recursion is factored away ⇒ finite set of elementary trees.

The elementary tree of a lexical predicate contains slots for all arguments of the predicate, for nothing more.

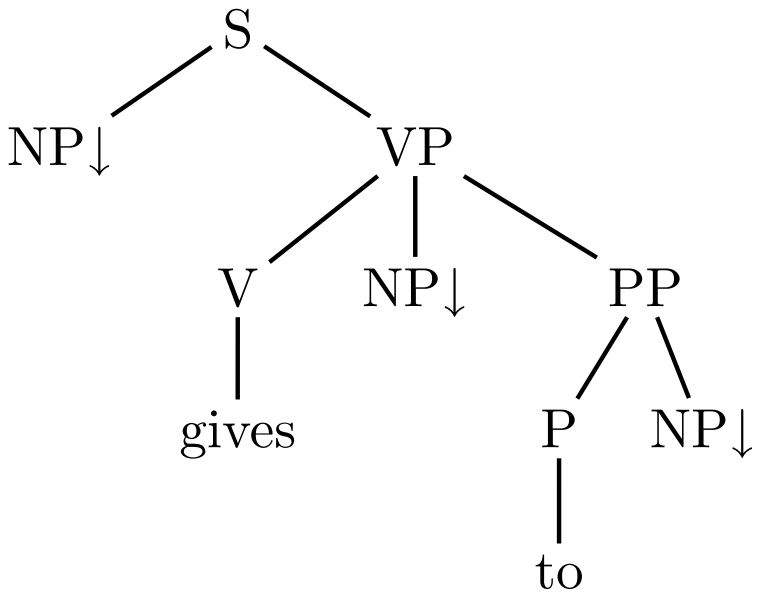
Besides lexical predicates, there are functional elements (complementizers, determiners, auxiliaries, negation) whose treatment in LTAG is less clear. They can be

- either in separate elementary trees (e.g., XTAG grammar)
- or in the elementary tree of the lexical item they are associated with.

Elementary trees (4)

Example:

(5) John gives a book to Mary

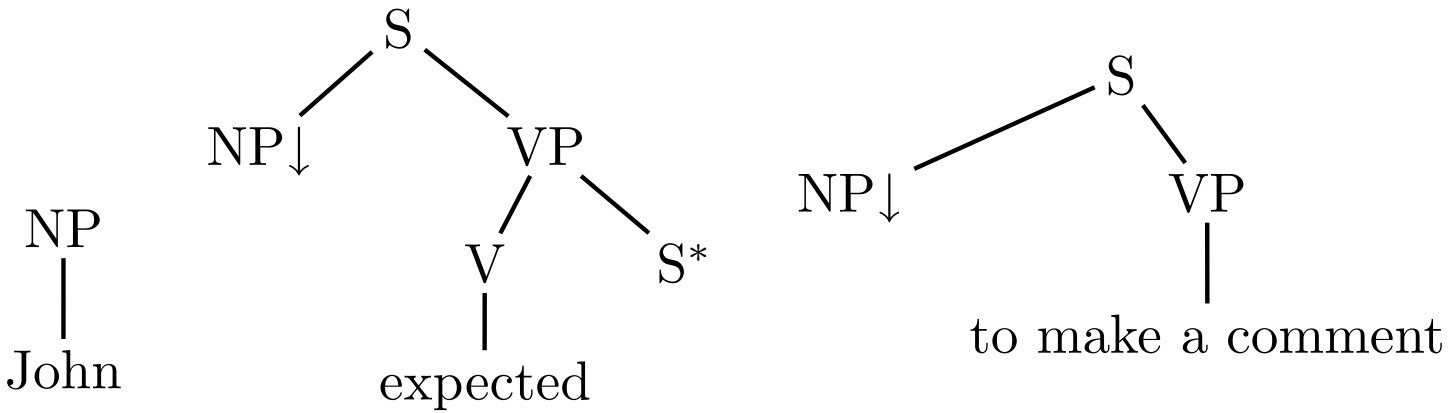


Elementary trees (5)

Example:

(6) John expected Mary to make a comment

expected selects for a subject NP and an infinitival sentence:

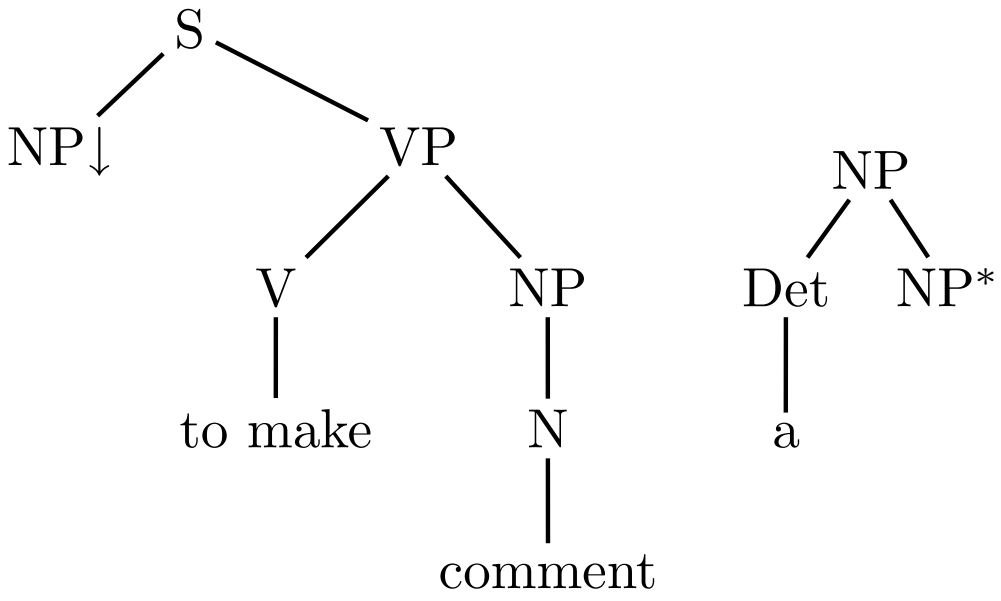


The sentential object is realised as a foot node in order to allow extractions:

(7) whom does John expect to come?

Elementary trees (6)

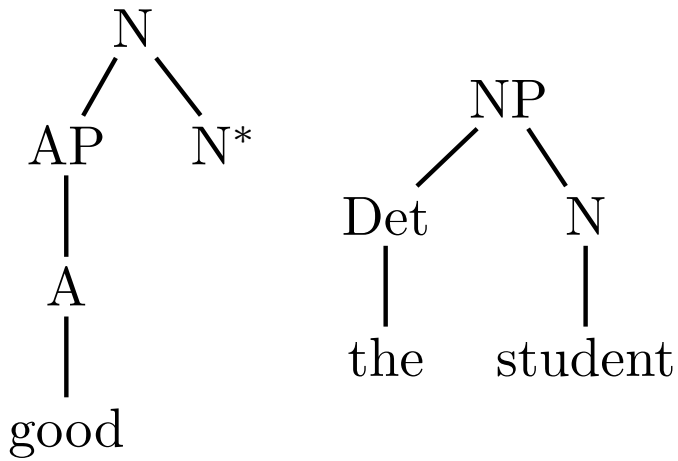
to make a comment: *make* and *comment* in the same elementary tree since they form a light verb construction:



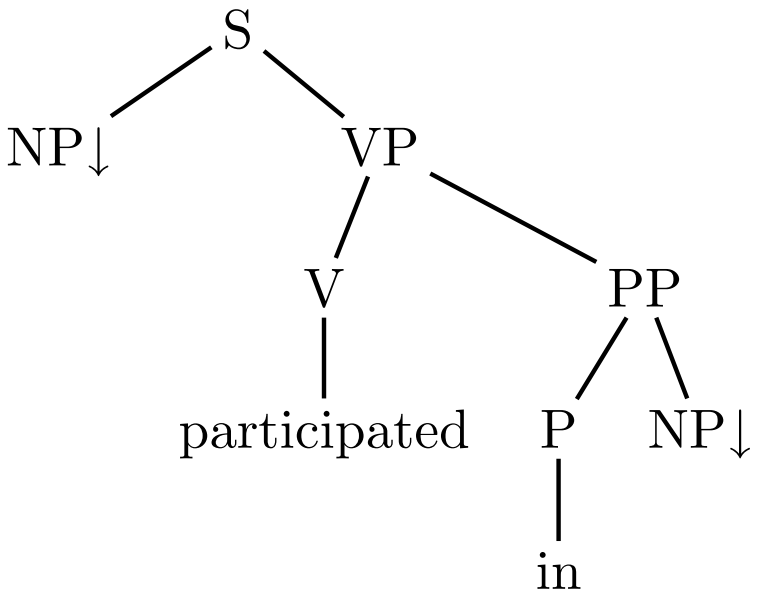
Elementary trees (7)

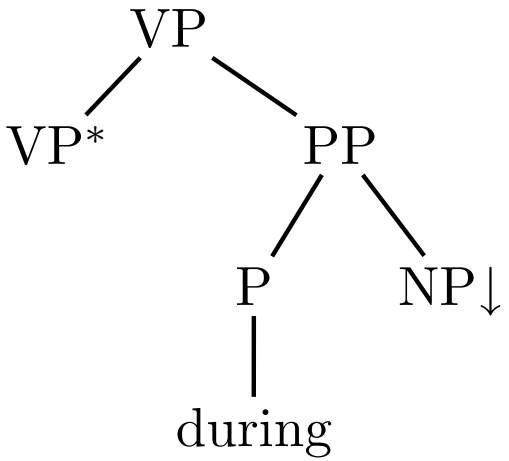
Example with modifiers:

(8) the good student participated in every course during the semester



Elementary trees (8)





Elementary trees (9)

Constraints on larger structures (constraints on "unbounded dependencies") need not be stipulated but follow from the possibilities of adjunction in the extended projections.

Fundamental LTAG hypothesis: Every syntactic dependency is expressed locally within a single elementary tree.

Non-local dependency corollary: Non-local dependencies always reduce to local ones once recursive structure is factored away.

Constituency and Dependency (1)

The derived tree gives the constituent structure.

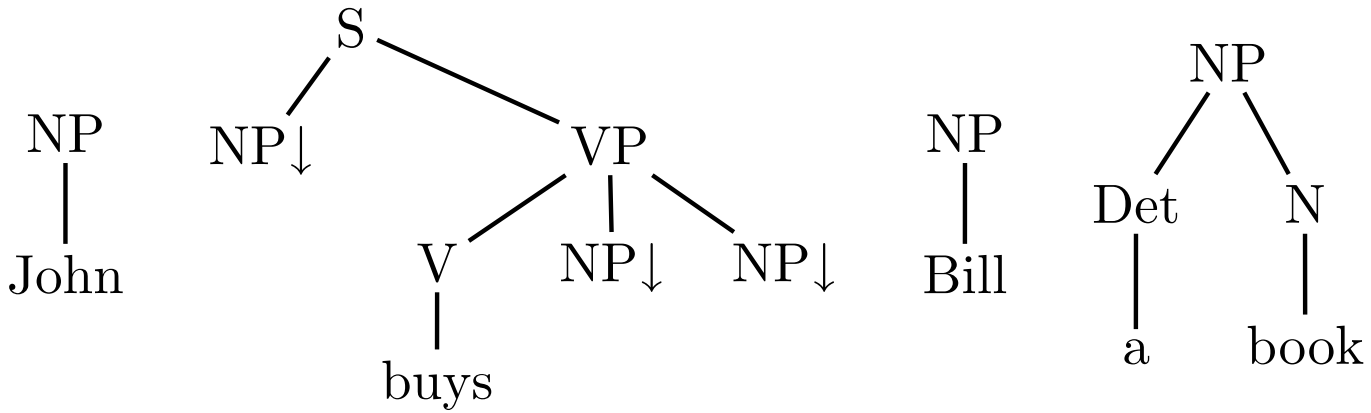
The derivation tree records the history of how the elementary trees are put together.
$\Rightarrow$ the edges in the derivation tree represent predicate-argument dependencies; the derivation tree is close to a semantic dependency graph.

$\Rightarrow$ compute semantics on derivation tree

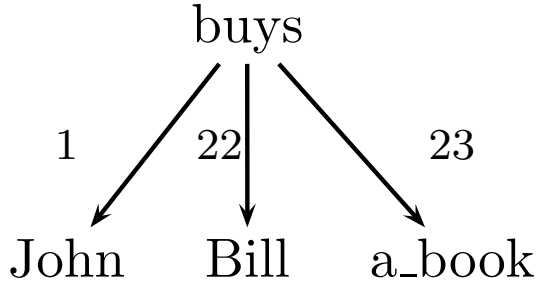
Constituency and Dependency (2)

(9) John buys Bill a book

Elementary trees:

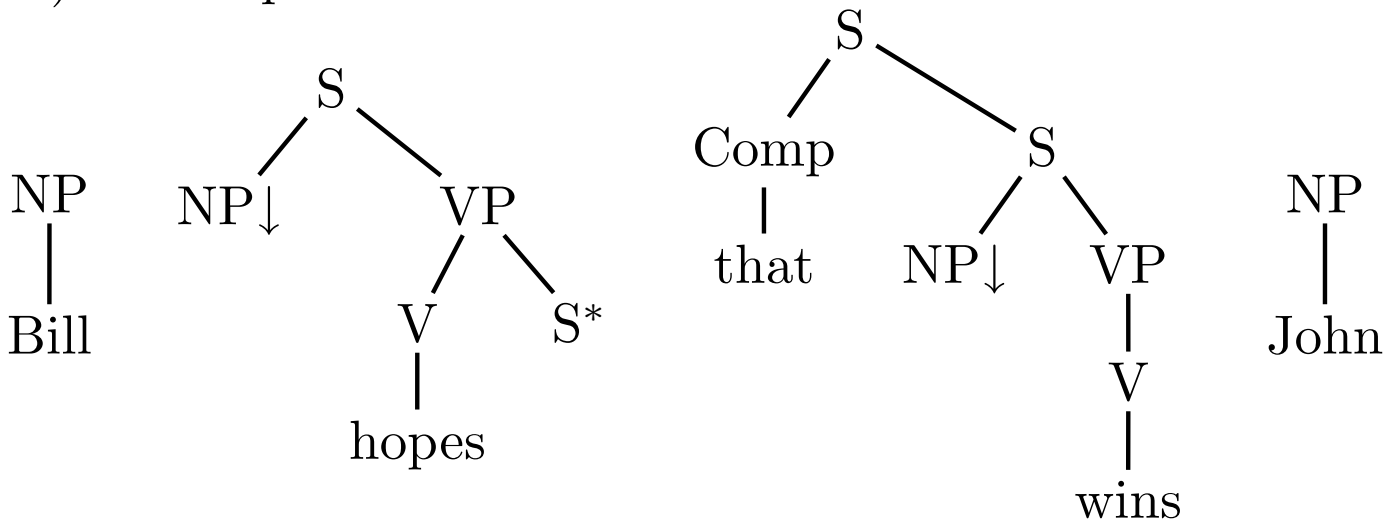


Derivation tree:

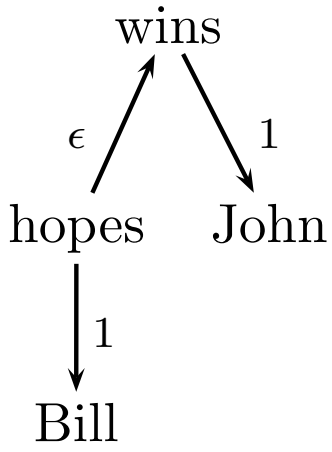


Constituency and Dependency (3)

(10) Bill hopes that John wins

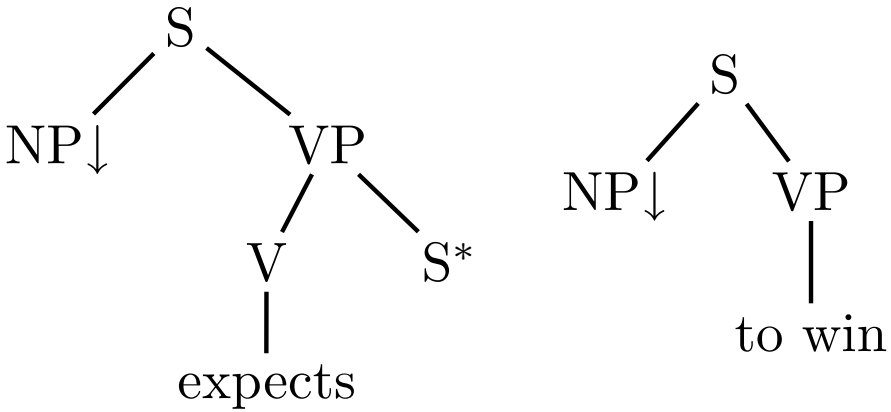


Derivation tree:

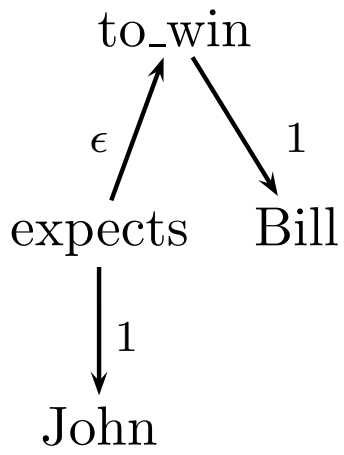


Constituency and Dependency (4)

(11) John expects [Bill to win]

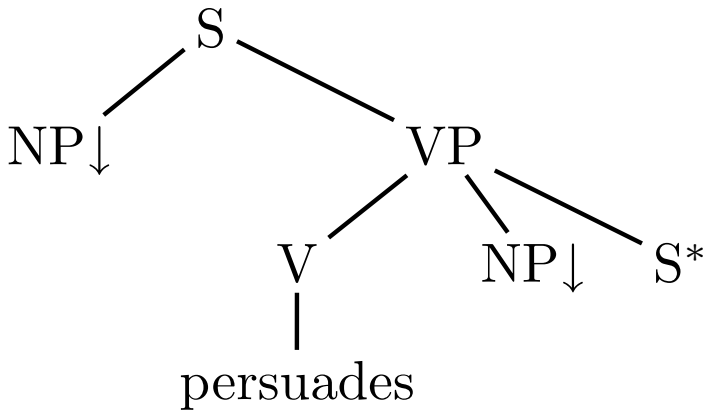


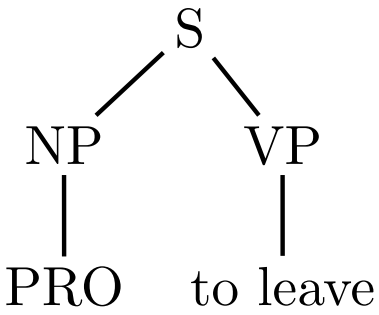
Derivation tree:



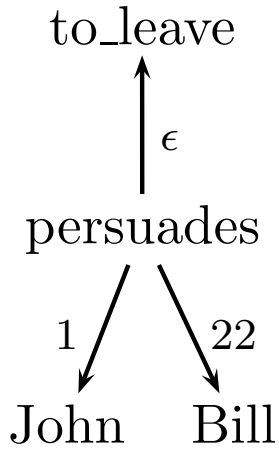
Constituency and Dependency (5)

(12) John persuades Bill [PRO to leave]



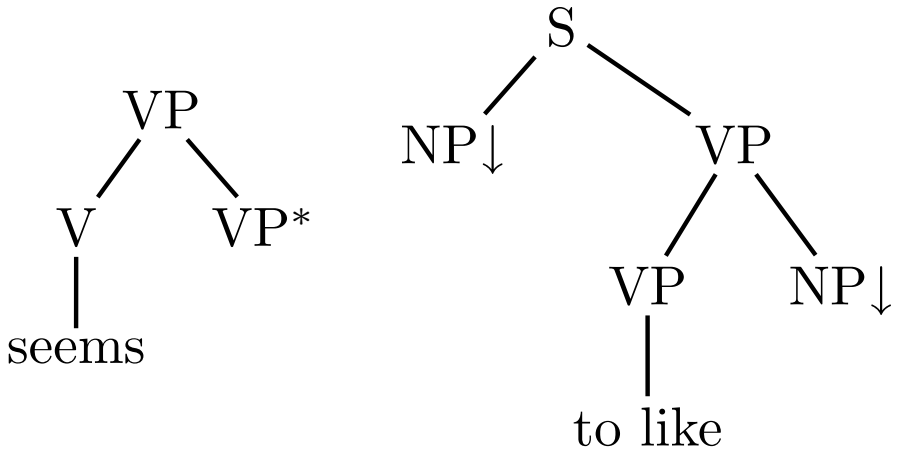


Derivation tree:

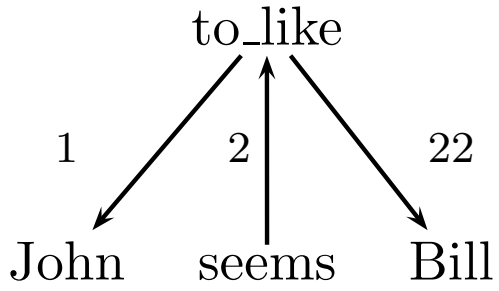


Constituency and Dependency (6)

(13) John seems to like Bill



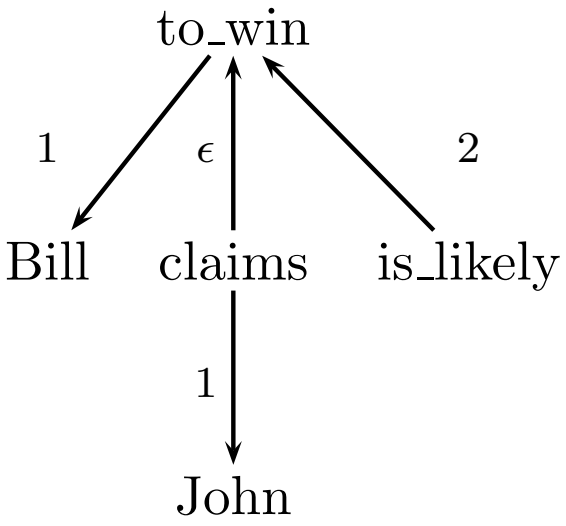
Derivation tree:



Constituency and Dependency (7)

The derivation tree is not always the semantic dependency structure:

(14) John claims Bill is likely to win



FTAG (1)

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

Each node has a top and a bottom feature structure (except substitution nodes that have only a top). Nodes in the same elementary tree can share features (extended domain of locality).

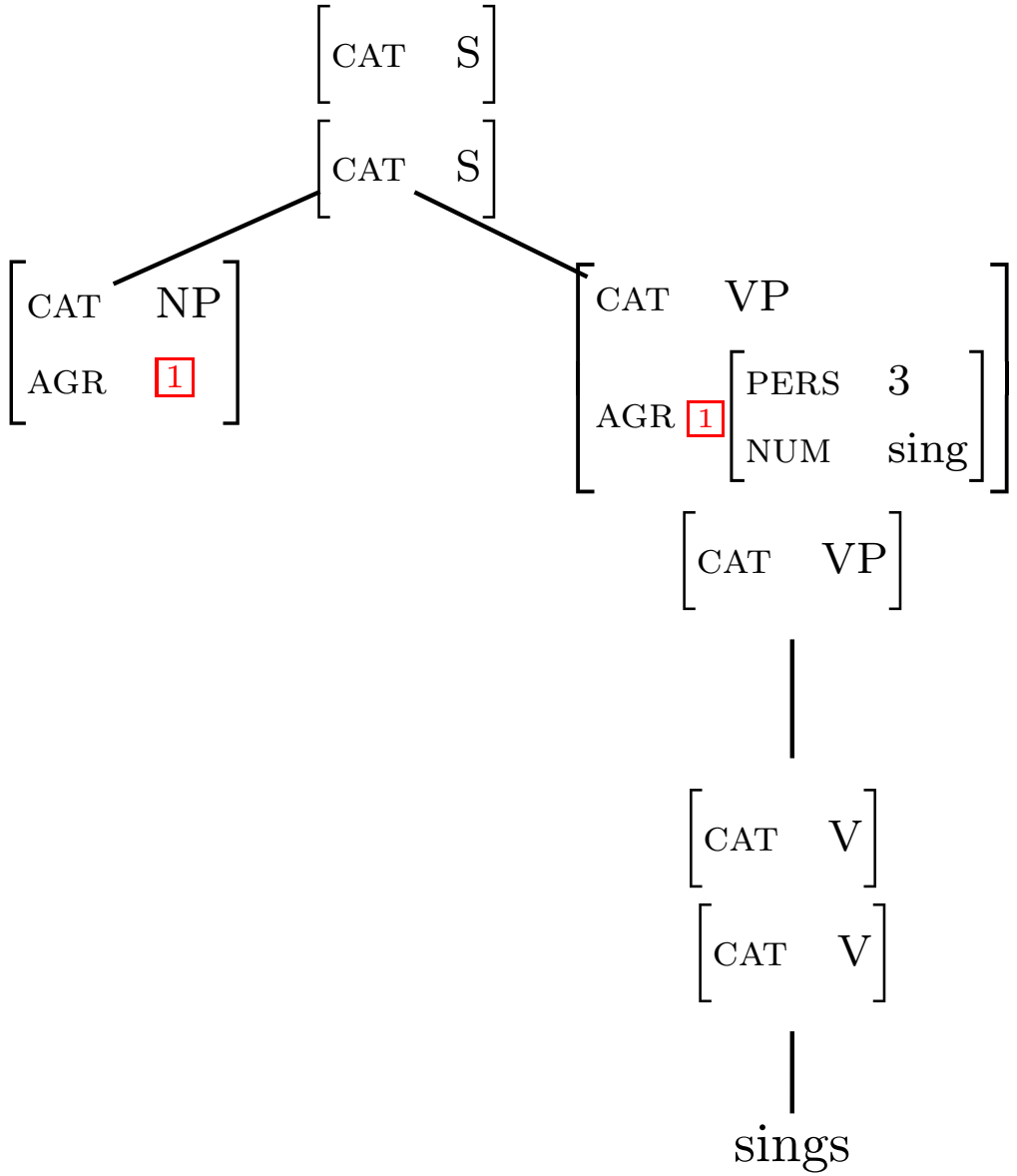
Intuition:

- The top feature structure tells us something about what the node presents within the surrounding structure, and
- the bottom feature structure tells us something about what the tree below the node represents.

In the final derived tree, both must be the same.

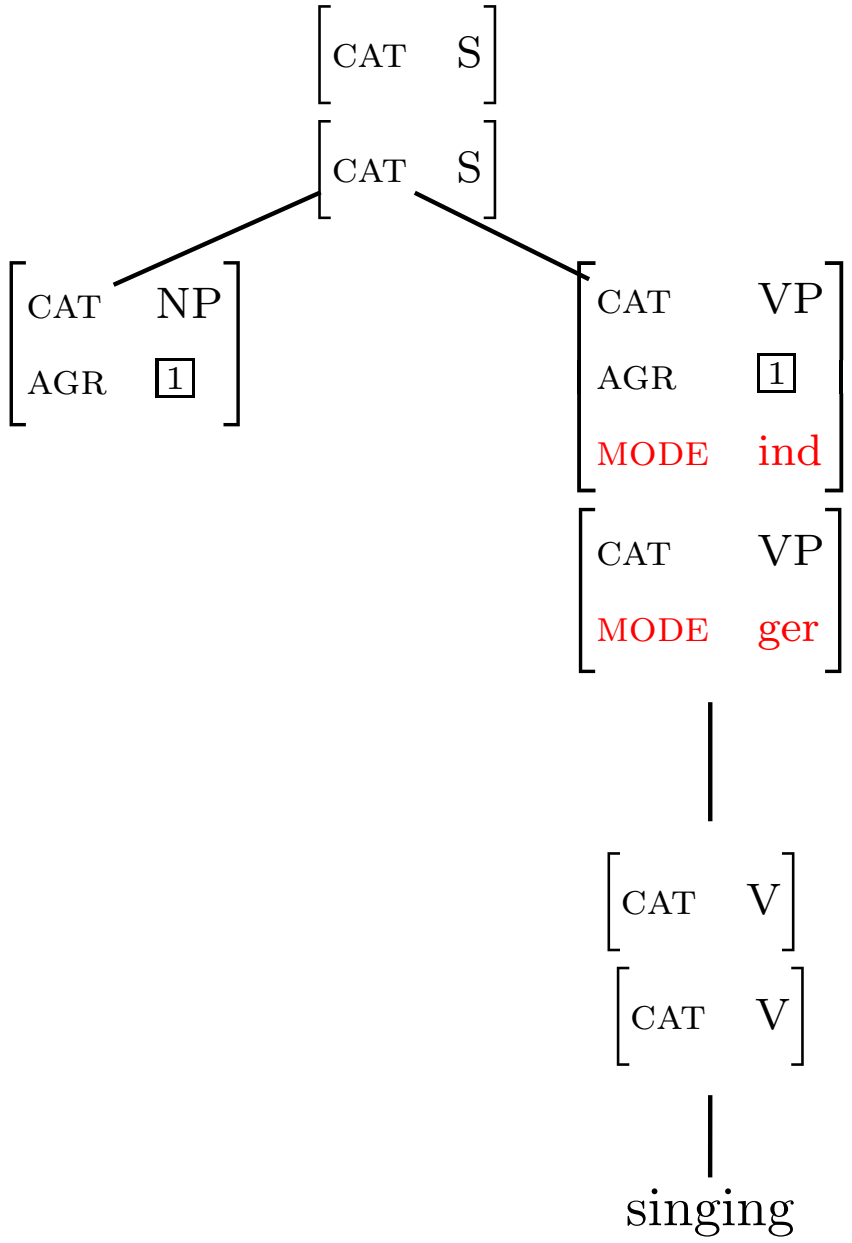
FTAG (2)

Example:



FTAG (3)

Example:



FTAG (4)

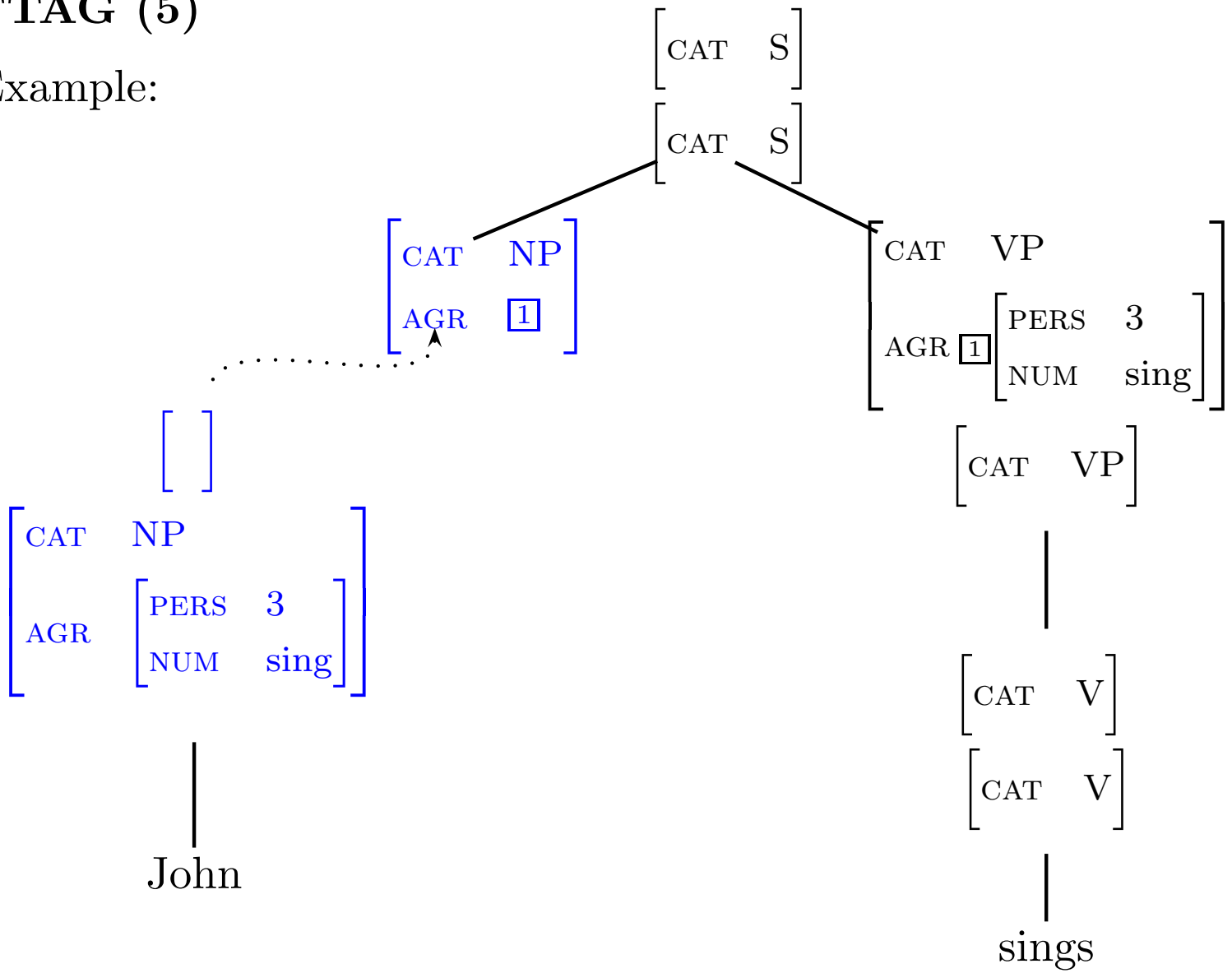
Unification during derivation:

- **Substitution**: the top of the root of the new initial tree unifies with the top of the substitution node
- **Adjunction**: the top of the root of the new auxiliary tree unifies with the top of the adjunction site,

  and the bottom of the foot of the new tree unifies with the bottom of the adjunction site.
- In the final derived tree, top and bottom unify for all nodes.

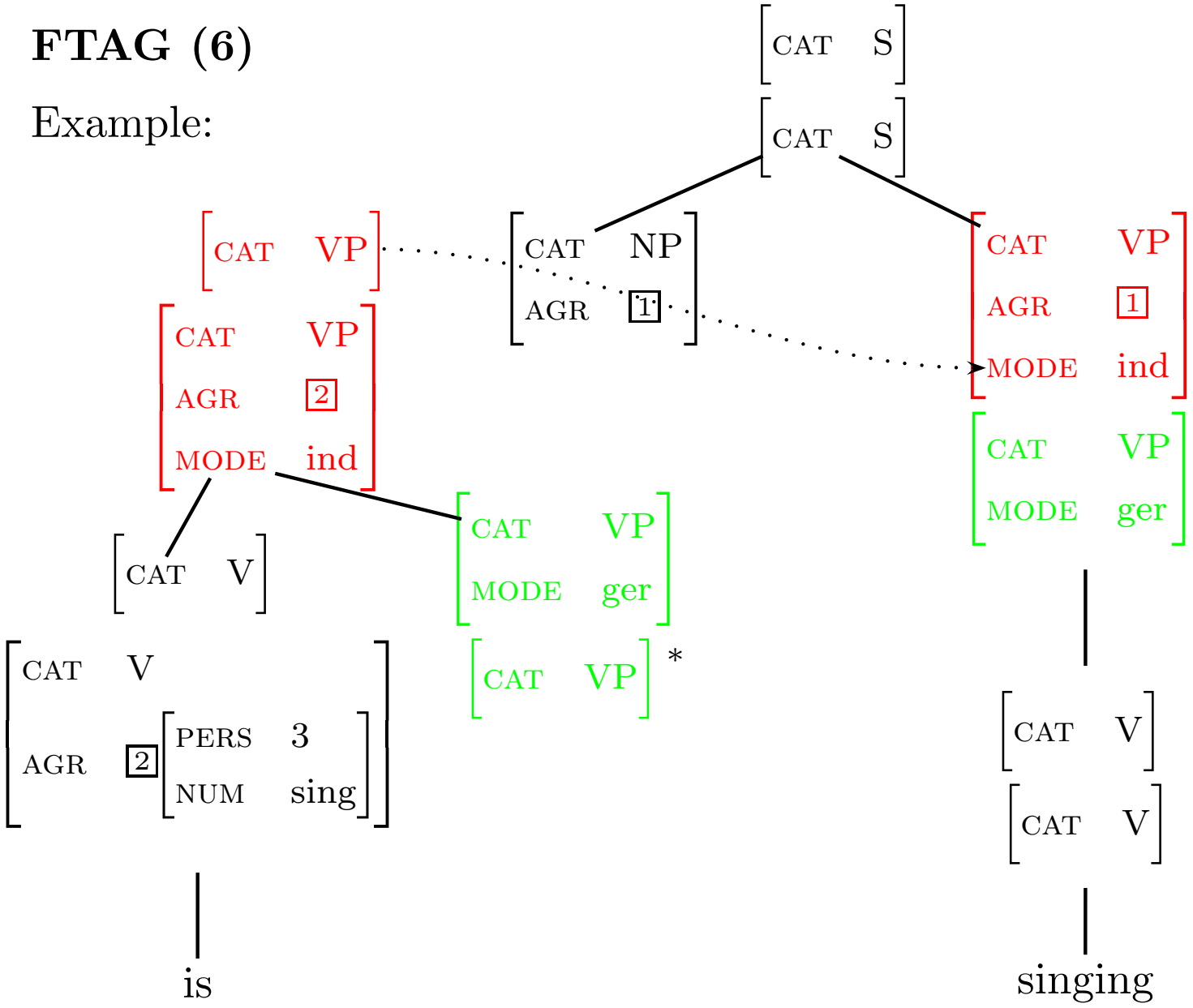
FTAG (5)

Example:



FTAG (6)

Example:



FTAG (7)

In FTAG, there are no explicit adjunction constraints. Instead, adjunction constraints are expressed via feature unification requirements.

Important: LTAG feature structures are restricted; there is only a finite set of possible feature structures.

Therefore, the following can be shown:

For each FTAG there exists a weakly equivalent TAG with adjunction constraints and vice versa. The two TAGs generate even the same sets of trees, only with different node labels.

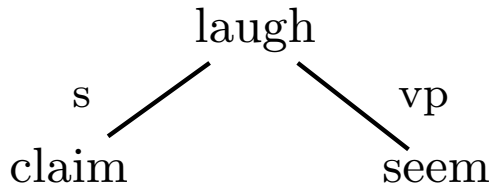
LTAG Semantics with Semantic Unification

1. Semantic Representations and Semantic Feature Structures
2. Semantic Unification
3. Disambiguation
4. Global Features

Semantic Representations (1)

Problems where derivation tree does not provide all links necessary for semantic computation:

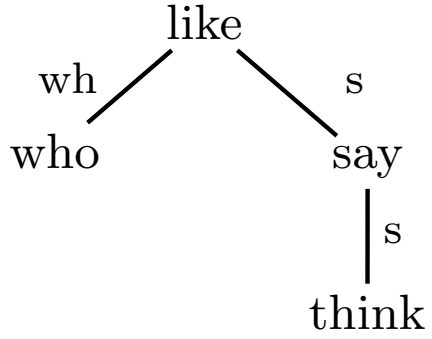
(15) Mary claims John seems to laugh



$claim'(m, (seem'(laugh'(j))))$

no link between *claim* and *seem*

(16) Who does Paul think John said Bill liked?



$who'(x, think'(p, say'(j, like'(b, x))))$

no link between *like* and *think*

(neither between *who* and *think*)

Semantic Representations (2)

Observation: whenever a semantic link in the derivation tree is missing, it is

1. either a link between trees attaching to different nodes in the same tree, i.e., attaching to nodes that can share features inside an elementary tree,
2. or a link between trees γ_1 and γ_2 such that γ_2 adjoins to the root of a tree that (adjoins to the root of a tree that ...) attaches to some node n in γ_1. In this case, indirectly, the top of n and the top of the root of γ_2 unify and thereby features can be shared.

$\Rightarrow$ the feature unification in FTAG provides the necessary links (Gardent & Kallmeyer 2003)

Semantic Representations (3)

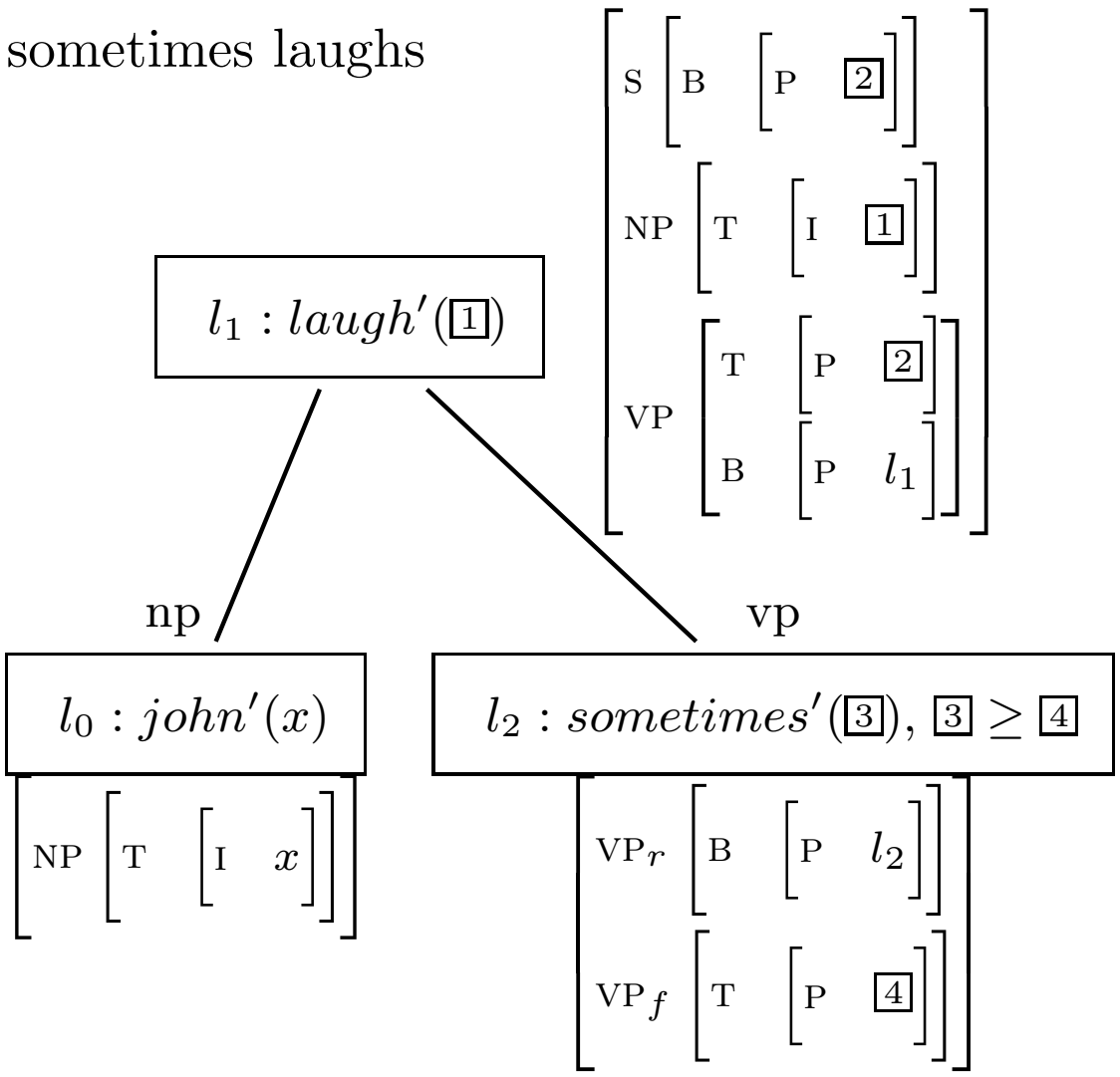
To separate more neatly between syntax and semantics, augment the derivation tree with semantic feature structure descriptions and do semantic unification on the derivation tree (Kallmeyer & Romero 2006):

- each elementary tree is linked to a semantic representation and a semantic feature structure description
- semantic composition = conjunction and additional feature value equations between these feature structure descriptions.

The feature structures link top and bottom features to the nodes in the elementary tree. Feature value identifications proceed parallel to syntactic feature structure unification.

Semantic Representations (4)

(17) John sometimes laughs



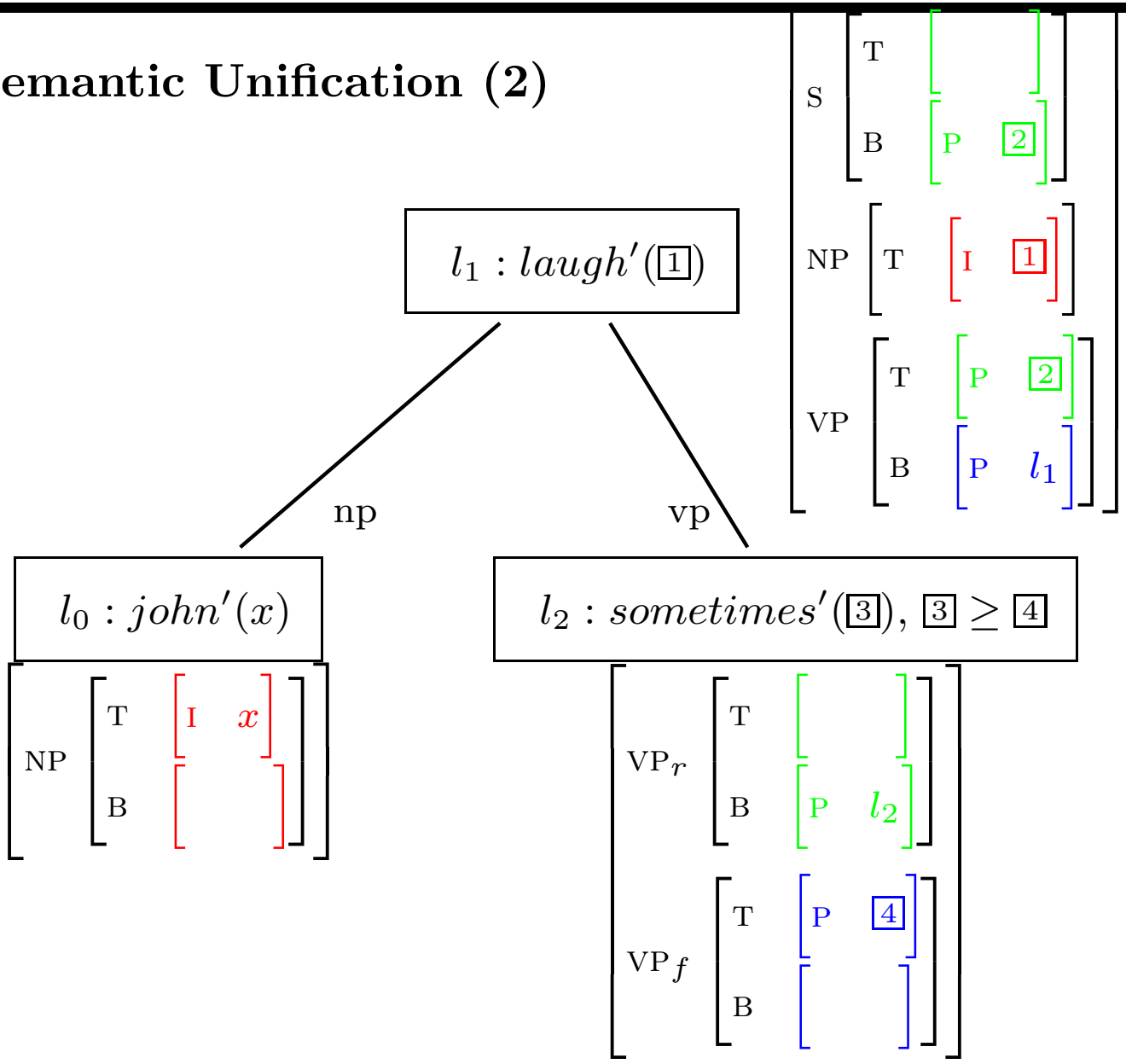
Semantic Unification (1)

for each edge in the derivation tree from γ_1 to γ_2 with position p:

1. the top of p in γ_1 and the top of the root in γ_2 are identified
2. and if γ_2 is an auxiliary tree, then the bottom of the foot node of γ_2 and the bottom of p in γ_1 are identified.

Furthermore, for all γ and all p in γ such that there is no edge with position p from γ to some other tree: the top and bottom of $\gamma.p$ are identified.

Semantic Unification (2)



Semantic Unification (3)

After unification the union of the semantic representations is built and the assignments obtained from the unifications is applied to it.

Assignment in our example: $\boxed{1} \rightarrow x$, $\boxed{4} \rightarrow l_1$, $\boxed{2} \rightarrow l_2$

Result:

$$l_1 : laugh'(x),\ l_0 : john'(x),\ l_2 : sometimes'(\boxed{3}),$$
$$\boxed{3} \geq l_1$$

Underspecified representation.

Disambiguation (1)

Disambiguation: Function assigning propositional labels to the remaining propositional metavariables while respecting the scope constraints.

$$l_1 : laugh'(x),\ l_0 : john'(x),\ l_2 : sometimes'(\boxed{3}),\ \boxed{3} \geq l_1$$

Only one disambiguation: $\boxed{3} \rightarrow l_1$. Leads to

$$l_0 : john'(x),\ l_2 : sometimes'(l_1 : laugh'(x))$$

The resulting set is interpreted conjunctively.

This yields $john'(x) \wedge sometimes'(laugh'(x))$

Global features (1)

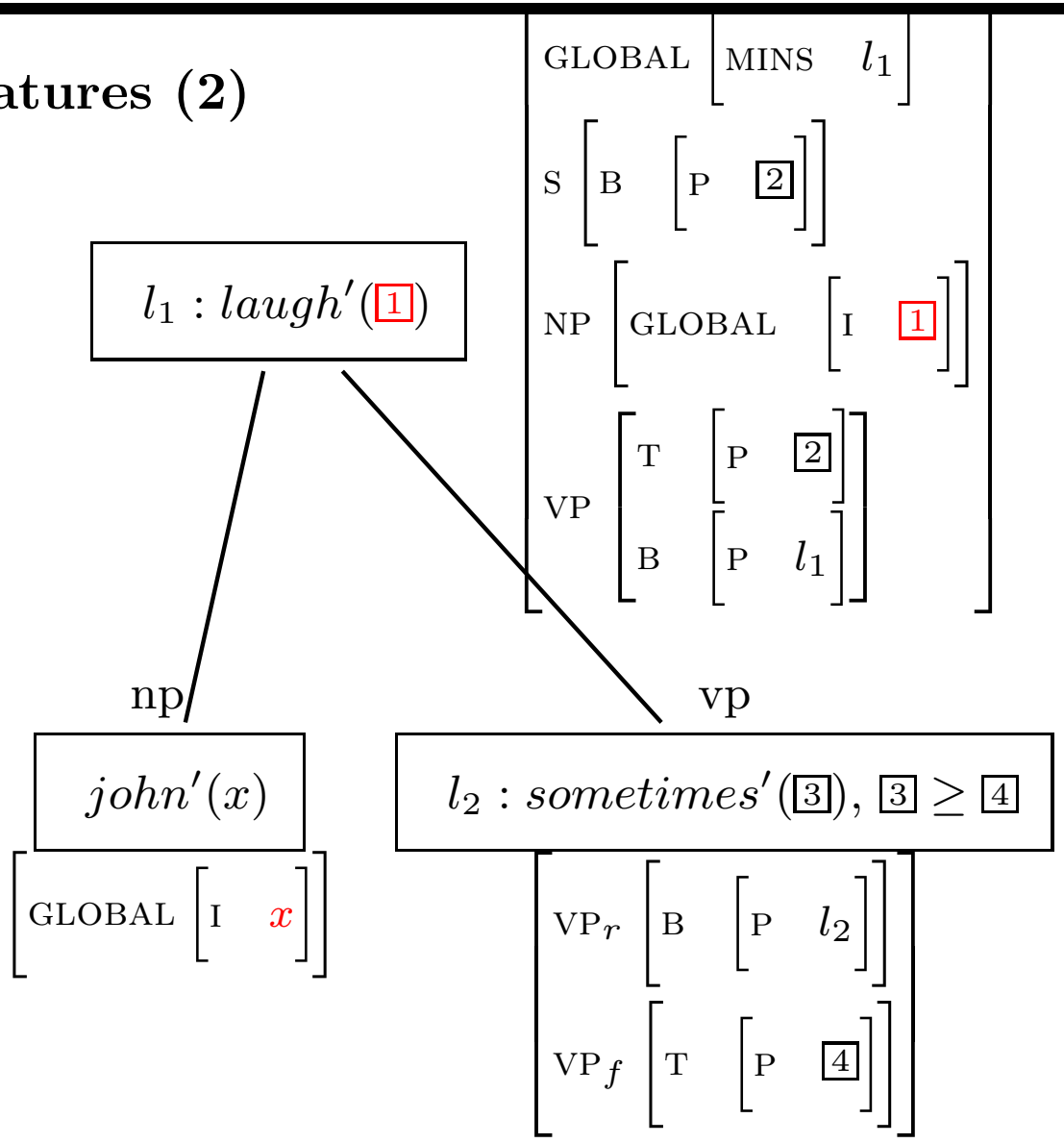
Intuition: Some features are linked to elementary trees as a whole.

Examples:

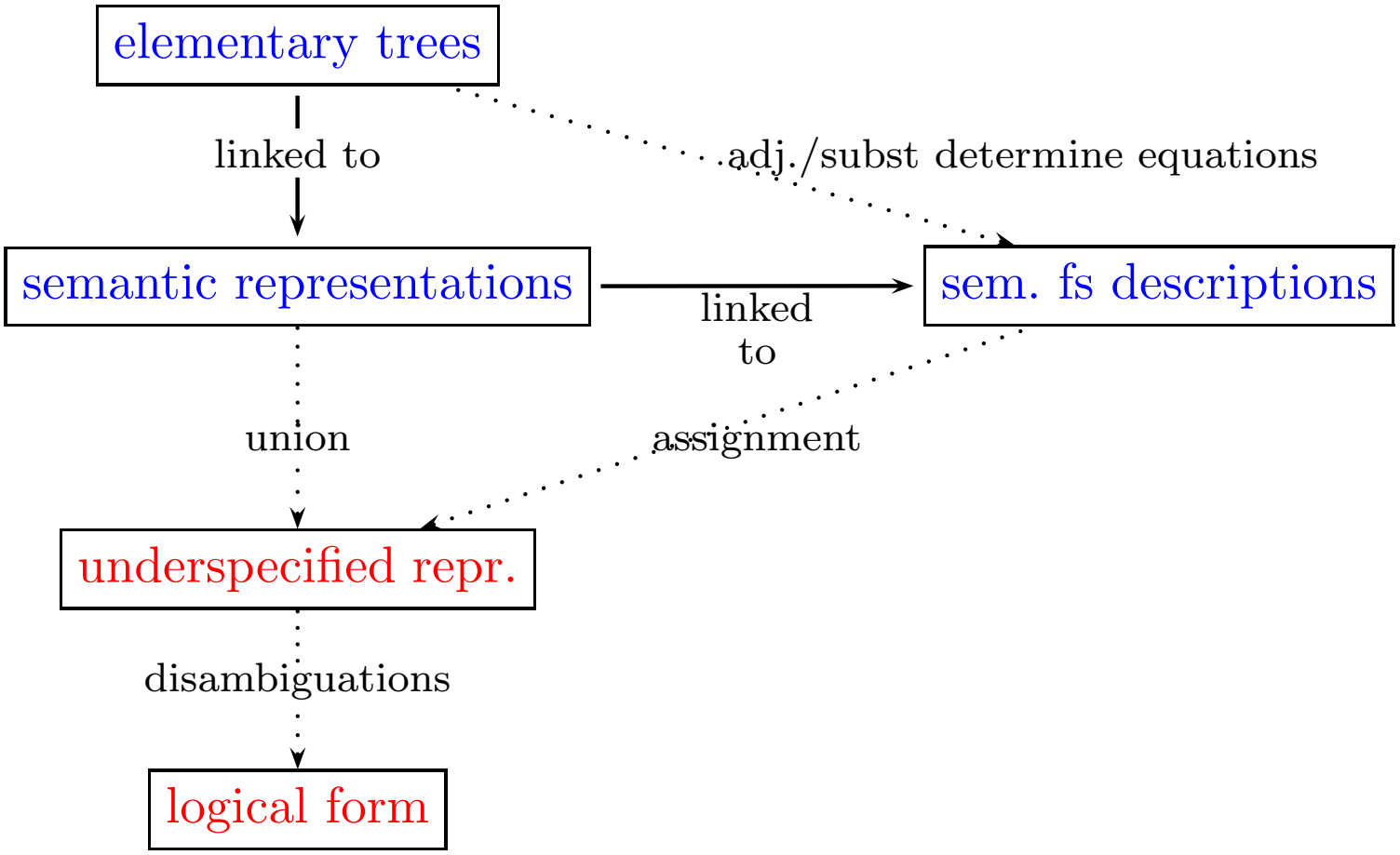
- Feature I in NP trees.
- Minimal proposition contributed by a verb.

Each tree has a feature GLOBAL and "requests" for global features of other trees can be put on specific node positions.

Global features (2)



LTAG Semantics: Summary



Interface structure: derivation tree, determines locally the compositions of elementary trees and of fs descriptions.

Short Introduction to

Lexical Resource Semantics (LRS)

A Short Introduction to LRS

- HPSG: Grammar = $\langle$Signature, Set of Principles$\rangle$
- Model theoretic: Linguistic expressions as sets of structures
- Consequences for LRS in HPSG:
  - LRS principles describe Ty2 terms; do not contain Ty2 terms
  - Underspecification on the description level

LRS (Framework 2)

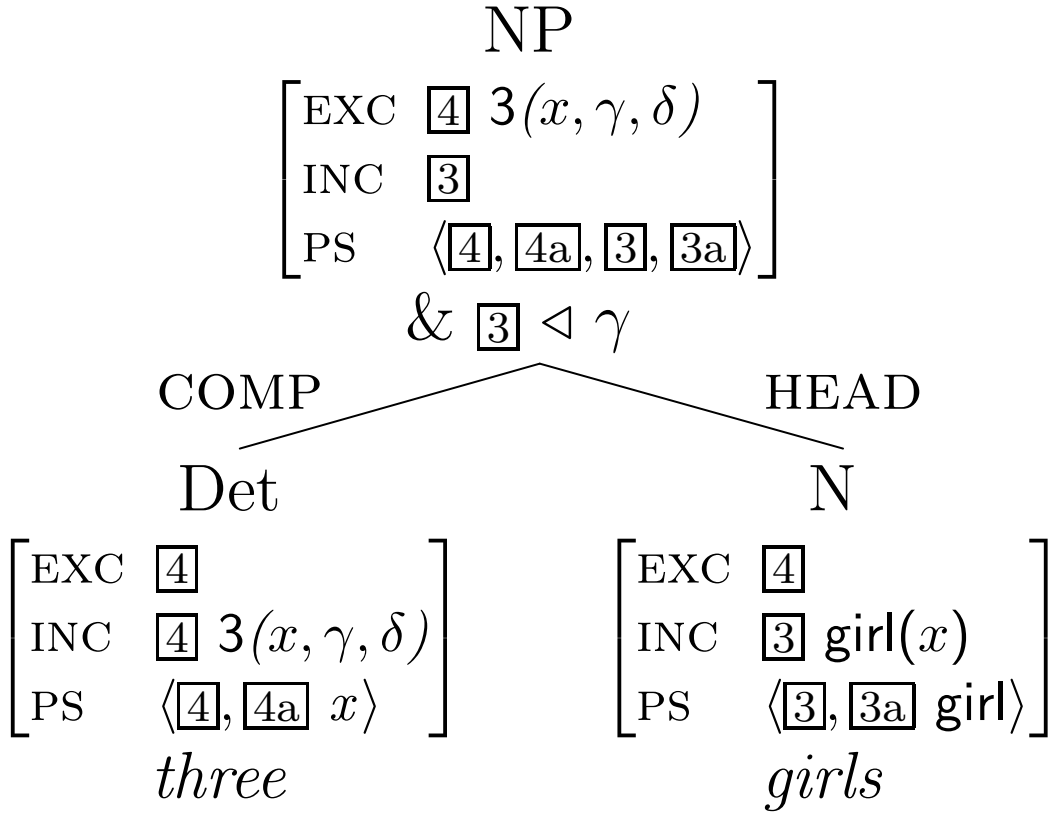
- Grammar (signature and principles) of Ty2
- Feature geometry for LRS with separaton of local content and an LF attribute on signs with *lrs* value:

$$sign\left[\begin{array}{ll} \textsc{synsem loc content} & \left[\begin{array}{ll}\textsc{var} & me\\ \textsc{main} & me\end{array}\right]\\ \textsc{lf} \quad lrs & \left[\begin{array}{ll}\textsc{external content} & me\\ \textsc{internal content} & me\\ \textsc{parts} & list\end{array}\right]\end{array}\right]$$

- Principles
  - excont Principle, incont Principle
  - lrs Projection Principle
  - Semantics Principle

LRS: Quantified NP

$3(x, \ldots \mathsf{girl}'(x) \ldots, \ldots)$



The Core Principles (1)

The INCONT PRINCIPLE (IContP):
In each *lrs*, the INCONT value is an element of the PARTS list and a component of the EXCONT value.

The EXCONT PRINCIPLE (EContP):

Clause (a):
In every phrase, the EXCONT value of the non-head daughter is an element of the non-head daughter's PARTS list.

Clause (b):
In every utterance, every subexpression of the EXCONT value of the utterance is an element of its PARTS list, and every element of the utterance's PARTS list is a subexpression of the EXCONT value.

The Core Principles (1)

LRS Projection Principle: In each *headed-phrase*,

the EXCONT value of the head and the mother are identical,

the INCONT value of the head and the mother are identical,

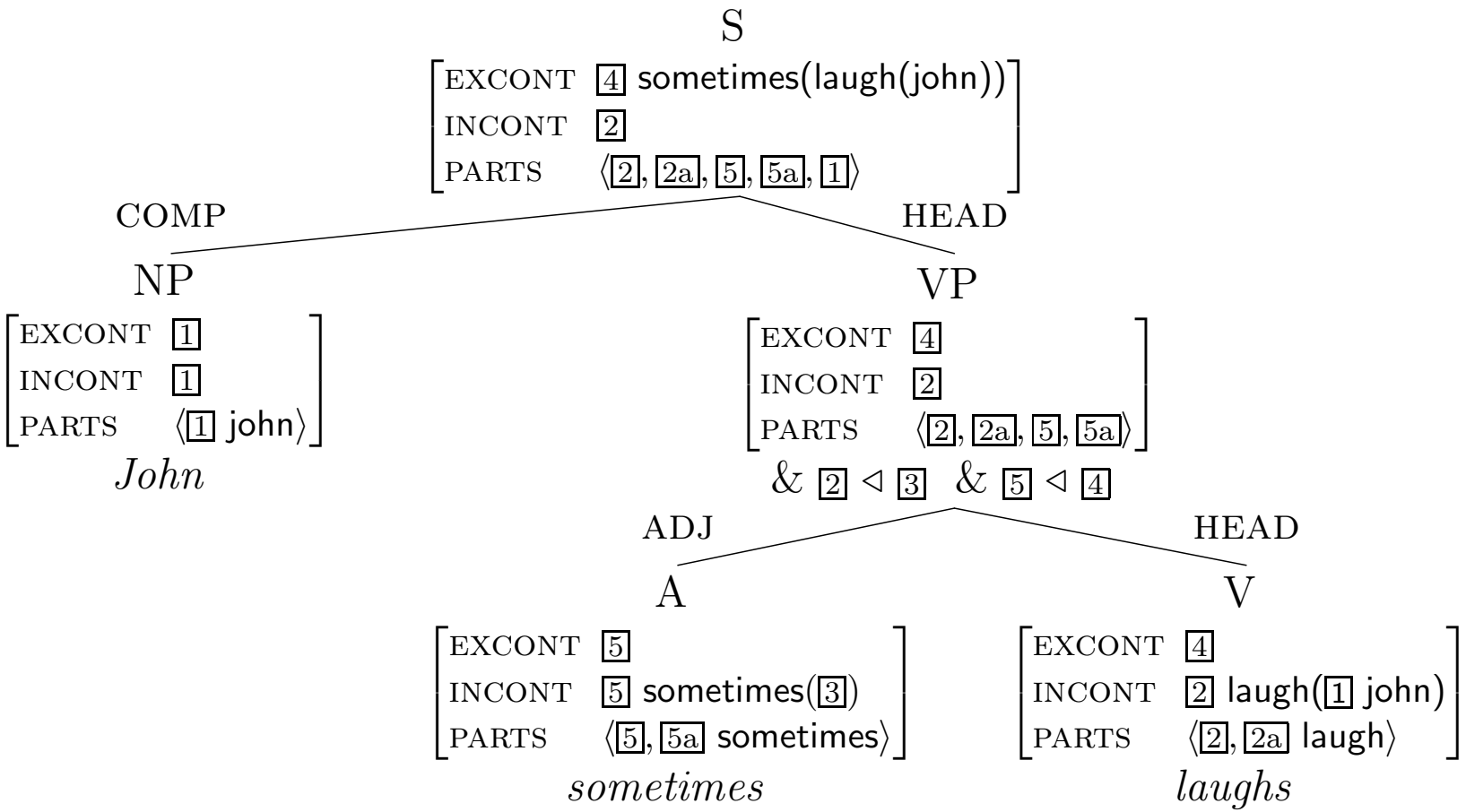
the PARTS value contains all and only the elements of the PARTS values of the daughters.

Semantics Principle: In each *headed-phrase*,

if the non-head is a quantifier then its INCONT value is of the form $Qx[\rho \circ \nu]$, the INCONT value of the head is a component of ρ, and the INCONT value of the non-head daughter is identical with the EXCONT value of the head daughter;

if the non-head is a quantified NP with an EXCONT value of the form $Qx[\rho \circ \nu]$, then the INCONT value of the head is a component of ν.

LRS: Simple Sentence



LRS: Lexical Entry

$$\left[\begin{array}{ll}
\textsc{phon} & \langle \textit{laughs} \rangle \\
\textsc{ss loc} & \left[\begin{array}{ll}
\textsc{cat} & \left[\begin{array}{ll} \textsc{head} & \textit{verb} \\ \textsc{subcat} & \left\langle \mathrm{NP}_{\boxed{\mathrm{X}}} \right\rangle \end{array}\right] \\
\textsc{cont} & \left[\begin{array}{ll} \textsc{main} & \mathsf{walk'} \\ \textsc{var} & \mathsf{e} \end{array}\right]
\end{array}\right] \\
\textsc{lf} & \left[\begin{array}{ll}
\textit{lrs} & \\
\textsc{excont} & \textit{me} \\
\textsc{incont} & \boxed{2}\ \mathsf{laugh'}_{@}(\mathsf{e}, \boxed{\mathrm{X}}) \\
\textsc{parts} & \left\langle \boxed{2}\ \mathsf{laugh'}_{@}(\mathsf{e}, \boxed{\mathrm{X}}),\ \boxed{2a}\ @,\ \boxed{2b}\ \mathsf{e},\ \boxed{2c},\ \boxed{3}\ \exists \mathsf{e}\ \phi \right\rangle
\end{array}\right]
\end{array}\right]$$

$\bigwedge\ \boxed{2} \lhd \boxed{3}$

The LRS Architecture (Summary)

- Typed feature logic for syntax and semantics
- Interleaved specification of syntax and semantics
- Underspecification on the level of the feature logic (subterm constraints)
- Components of an LRS theory:
  1. A grammar (signature and principles) of Ty2
  2. LRS principles for semantic composition on the basis of
     - EXCONT, INCONT and PARTS, using
     - subterm constraints
  3. Lexical entries for each word

End of Part I

Part II

Overview

1. Constraint-based computational semantics
2. Specific analyses in LRS and LTAG
   (a) Quantifier Scope
   (b) Restrictions for Quantifier Scope
   (c) Negative Concord
   (d) Negative Polarity Items
3. Compositionality
4. Comparing the frameworks

1. Constraint-based computational semantics
2. Specific analyses in LRS and LTAG
   (a) Quantifier Scope
   (b) Restrictions for Quantifier Scope
   (c) Negative Concord
   (d) Negative Polarity Items
3. Compositionality
4. Comparing the frameworks

Constraint-based computational semantics (1)

Summary of the characteristics of LRS and LTAG.

Things LRS and LTAG have in common:

- Ty2 language for semantics
- scope constraints $\geq$ in LTAG and component-of constraints $\lhd$ in LRS for scope ambiguities
- feature logic for specifying and computing semantic representations

Constraint-based computational semantics (2)

Differences between LRS and LTAG:

- Extended domain of locality in LTAG
- General principles in LRS vs. lexicalization in LTAG
- Interleaved specification of syntax and semantics in LRS vs. modular architecture in LTAG
- LRS: (partial) descriptions of fully specified models vs
  LTAG: underspecified representations in the style of Bos with subsequent disambiguation (pluggings)

Specific analyses in LRS and LTAG

1. Quantifier Scope
2. Restrictions on Quantifier Scope
3. Negative Concord
4. Negative Polarity Items

Quantifier Scope (1)

Quantificational NPs: can in principle scope freely; their scope is not directly linked to their surface position.

(18) Exactly one student admires every professor
$\exists > \forall, \forall > \exists$

(19) Two policemen spy on someone from every city
$\forall > \exists > 2$ (among others)

(20) John seems to have visited everybody
$seem > \forall, \forall > seem$

(21) Three girls are likely to come
$three > likely, likely > three$

Quantifier Scope (2)

Two things must be guaranteed:

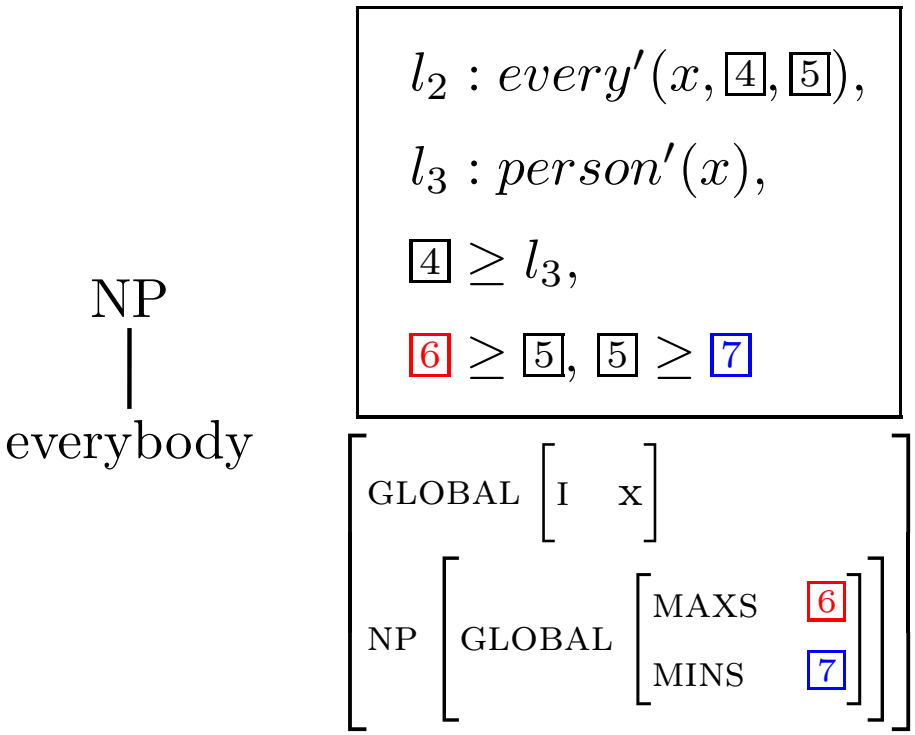
1. the proposition to which a quantifier attaches must be in its nuclear scope
2. a quantifier cannot scope higher than the next finite clause

Idea: scope window delimited by some maximal scope and some minimal scope for a quantifier.

Both, LTAG and LRS specify such scope windows for quantifiers.

Quantifier Scope (3)

LTAG analysis: global features MAXS and MINS.



Quantifier Scope (4)

(22) everybody laughs

$l_1 : laugh'(\boxed{1})$,
$\boxed{2} \geq \boxed{3}$

np

$l_2 : every'(x, \boxed{4}, \boxed{5})$,
$l_3 : person'(x)$,
$\boxed{4} \geq l_3$,
$\boxed{6} \geq \boxed{5}$, $\boxed{5} \geq \boxed{7}$

$$\begin{bmatrix} \text{GLOBAL} & \begin{bmatrix} \text{MINS} & l_1 \\ \text{MAXS} & \boxed{2} \end{bmatrix} \\ \text{S} & \begin{bmatrix} \text{B} & \begin{bmatrix} \text{P} & \boxed{3} \end{bmatrix} \end{bmatrix} \\ \text{VP} & \begin{bmatrix} \text{T} & \begin{bmatrix} \text{P} & \boxed{3} \end{bmatrix} \\ \text{B} & \begin{bmatrix} \text{P} & l_1 \end{bmatrix} \end{bmatrix} \\ \text{NP} & \begin{bmatrix} \text{GLOBAL} & \begin{bmatrix} \text{I} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{GLOBAL} & \begin{bmatrix} \text{I} & x \end{bmatrix} \\ \text{NP} & \begin{bmatrix} \text{GLOBAL} & \begin{bmatrix} \text{MAXS} & \boxed{6} \\ \text{MINS} & \boxed{7} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Quantifier Scope (5)

Result:

$l_1 : laugh'(x)$,

$l_2 : every'(x, \boxed{4}, \boxed{5})$, $l_3 : person'(x)$

$\boxed{2} \geq l_1$,

$\boxed{4} \geq l_3, \boxed{2} \geq \boxed{5}, \boxed{5} \geq l_1$

Disambiguation:

$\boxed{2} \rightarrow l_2$, $\boxed{4} \rightarrow l_3, \boxed{5} \rightarrow l_1$

yields $every'(x, person'(x), laugh'(x))$

Quantifier Scope (6)

LRS analysis:

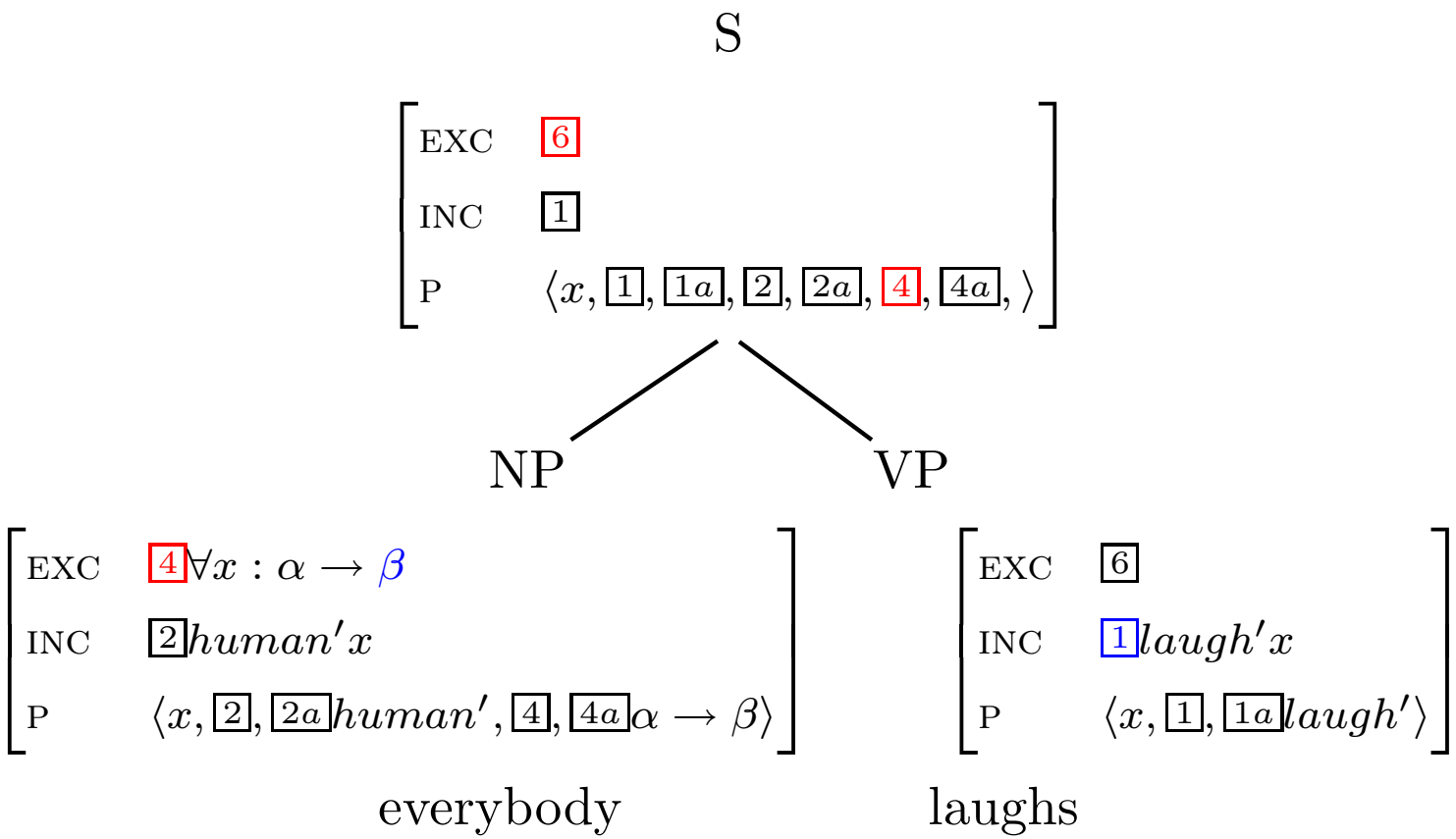
EXCONT PRINCIPLE:
a) The EXCONT of the non-head daughter is an element of the PARTS list of the non-head daughter.
b) In an utterance, everything on the PARTS list is a component of the EXCONT.

⇒ maximal scope of a quantifier: EXCONT of the next embedding utterance.

SEMANTICS PRINCIPLE:
If the non-head of a headed phrase is a quantified NP, then the INCONT of the head is a component of its nuclear scope.

⇒ minimal scope of a quantifier: INCONT of the verb it combines with.

Quantifier Scope (7)



Constraints: $\boxed{2} \lhd \alpha$ (from lexical entry of *everybody*), $\boxed{1} \lhd \beta$, $\boxed{4} \lhd \boxed{6}$

Quantifier Scope (8)

A simple example with scope ambiguity, illustrated in LRS:

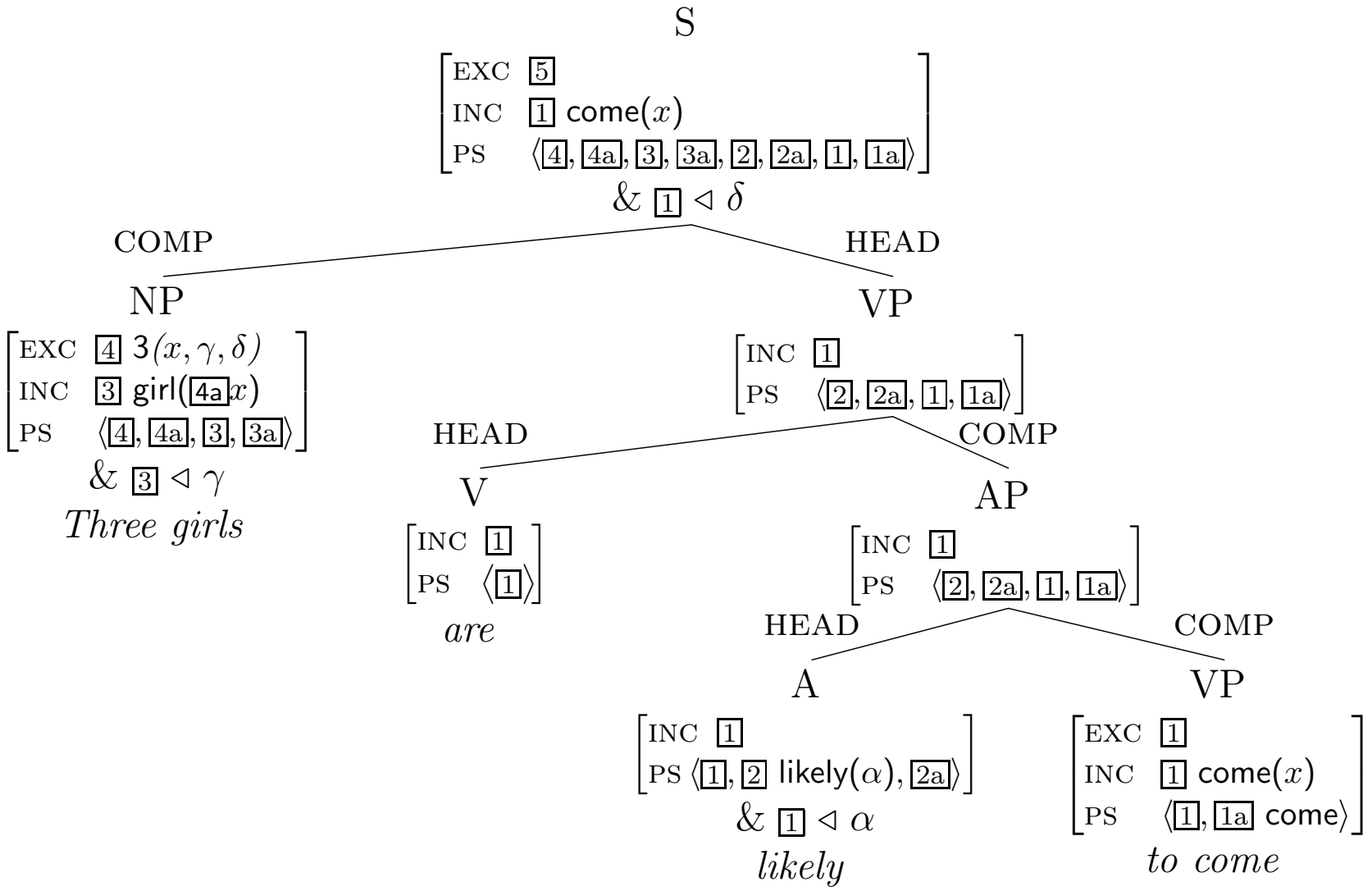
(23) Three girls are likely to come.

(24) Two readings:

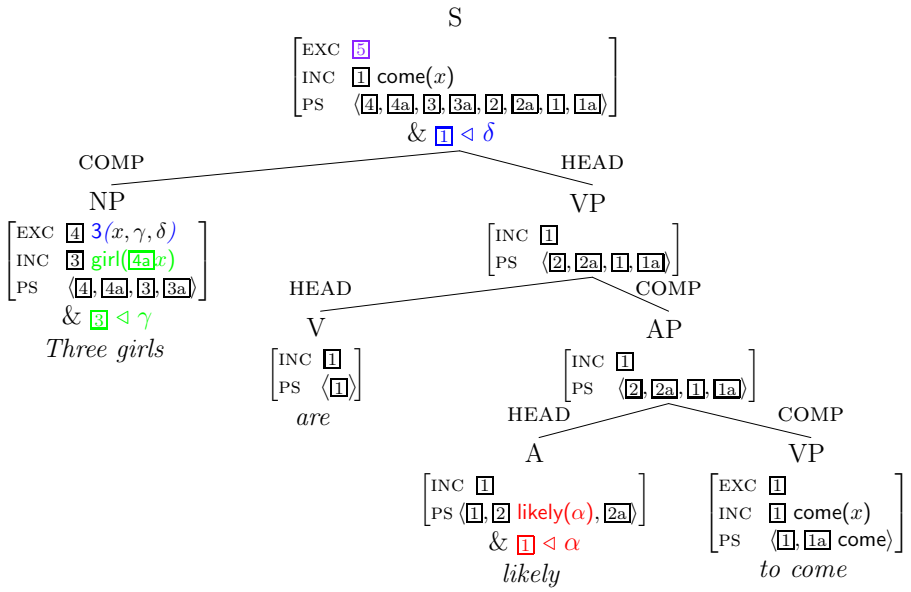
a. $\mathsf{3}(x, \mathsf{girl}'(x), \mathsf{likely}'(\mathsf{come}'(x)))$

b. $\mathsf{likely}'(\mathsf{3}(x, \mathsf{girl}'(x), \mathsf{come}'(x)))$

Quantifier Scope (9)



Quantifier Scope (10)



The grammar specifies two possible values for 5:

1. $\boxed{5} = 3(x,\ \text{girl}'(x),\ \text{likely}'(\text{come}'(x)))$
2. $\boxed{5} = \text{likely}'(3(x,\ \text{girl}'(x),\ \text{come}'(x)))$

Quantifier Scope (11)

Summary: both approaches model the freedom of quantifiers wrt scope

- using features for a quantifier scope window, and
- using underspecified representations involving 'component-of'-constraints

The use of feature structures and feature value identifications in combination with underspecification allows to avoid movements (in the syntax or in LF) as assumed in traditional generative semantics.

1. Constraint-based computational semantics
2. Specific analyses in LRS and LTAG
   (a) Quantifier Scope
   (b) Restrictions for Quantifier Scope
   (c) Negative Concord
   (d) Negative Polarity Items
3. Compositionality
4. Comparing the frameworks

Restrictions on Quantifier Scope (1)

Three things may happen to the upper boundary of the scoping possibilities of an NP when that NP is the argument of some predicate P embedded under some higher predicate Q:

1. Q blocks NP-scope,

   (25) Mary thinks John likes everybody
   thinks > everybody, *everybody > thinks

2. Q lets the NP-scope pass imposing no limitations,

   (26) Mary tries to be nice to everybody
   tries > everybody, *everybody > tries

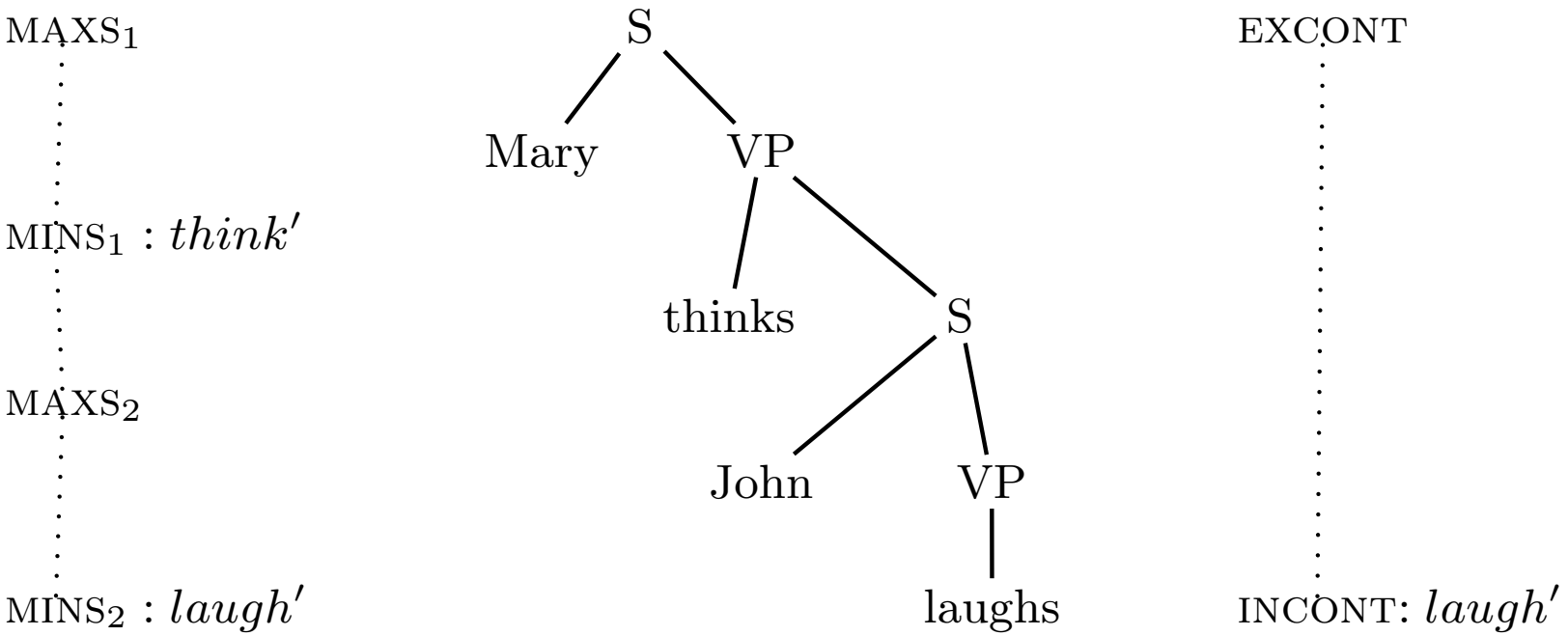
3. Q lets NP-scope pass imposing some limitations.

   (27) Two policemen spy on someone from every city
   $\forall > \exists > 2$ (among others), $*\forall > 2 > \exists$

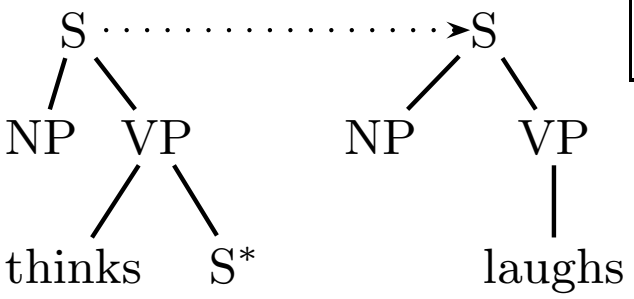
Restrictions on Quantifier Scope (2)

In LTAG, because of the extended domain of locality, it is straightforward to define some elements as blocking the embedded scope window and defining a new one for higher quantifiers:

(28) Mary thinks John laughs



Restrictions on Quantifier Scope (3)



$l_1 : laugh'(j)$,

$\boxed{1} \geq \boxed{2}$

s

$l_2 : think'(m, \boxed{3})$,

$\boxed{4} \geq \boxed{5}$, $\boxed{3} \geq \boxed{6}$

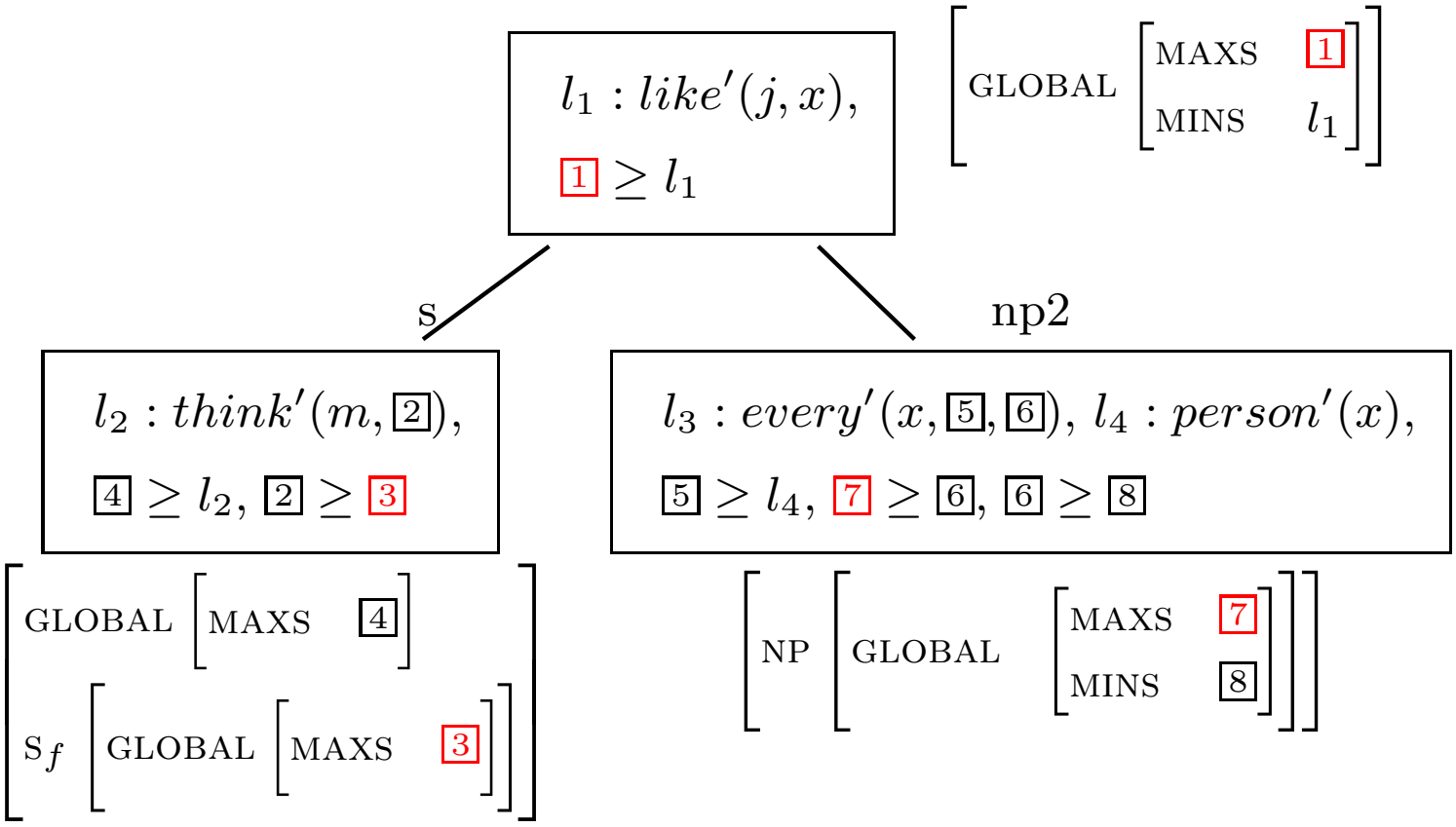
Argument of attitude verb embeds MAXS of embedded verb

$$\begin{bmatrix} \text{GLOBAL} & \begin{bmatrix} \text{MAXS} & \boxed{1} \end{bmatrix} \\ \text{S} & \begin{bmatrix} \text{B} & \begin{bmatrix} \text{P} & \boxed{2} \end{bmatrix} \end{bmatrix} \\ \text{VP} & \begin{bmatrix} \text{T} & \begin{bmatrix} \text{P} & \boxed{2} \end{bmatrix} \\ \text{B} & \begin{bmatrix} \text{P} & l_1 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{GLOBAL} & \begin{bmatrix} \text{MAXS} & \boxed{4} \end{bmatrix} \\ \text{S}_r & \begin{bmatrix} \text{B} & \begin{bmatrix} \text{P} & \boxed{5} \end{bmatrix} \end{bmatrix} \\ \text{VP} & \begin{bmatrix} \text{T} & \begin{bmatrix} \text{P} & \boxed{5} \end{bmatrix} \\ \text{B} & \begin{bmatrix} \text{P} & l_2 \end{bmatrix} \end{bmatrix} \\ \text{S}_f & \begin{bmatrix} \text{GLOBAL} & \begin{bmatrix} \text{MAXS} & \boxed{6} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Restrictions on Quantifier Scope (4)

(29) Mary thinks John likes everybody



only one scope order: $think'(m, every'(x, person'(x), like'(j, x)))$

Restrictions on Quantifier Scope (5)

LRS analysis:

LRS specifies scope islands in general grammar principles, like any other restrictions in HPSG.

Universal Boundary Principle

Universal quantifiers may not outscope finite clauses.

Or, slightly more technically:

For each finite clause, each universal quantifier contributed within the clause (= member of its PARTS list) is a component of the EXCONT of the clause.

Restrictions on Quantifier Scope (6)

(30) Two policemen spy on someone from every city.

Restriction: if *every* scopes over *some*, then no other quantifiers can scopally intervene, i.e., *every* has immediate scope over *some*.

Constraint cannot be captured with component-of constraints of the form $l \leq h$ with l being a label, h a hole.

Therefore, LTAG allows for slightly different scope constraints.

Restrictions on Quantifier Scope (7)

LTAG MAXS versus LRS EXCONT as upper quantifier scope limit:

- EXCONT: contains quantifier itself as component:

  NP node: EXC $\boxed{4}\forall x[\alpha \rightarrow \beta]$.

  $\boxed{4}$ is a component of the EXCONT of the utterance.

- MAXS (e.g., $\boxed{6}$) limits nuclear scope, not the quantifier:

  $l_3 : every'(x, \boxed{4}, \boxed{5})$. $\boxed{6} \geq \boxed{5}$ (not $\boxed{6} \geq l_3$)

  I.e., the quantifier can be higher than the MAXS limiting its nuclear scope.

Restrictions on Quantifier Scope (8)

(31) Two policemen spy on someone from every city.

$^{*}every > two > someone$

LTAG analysis: exclude this order by deriving a constraint saying that the maximal nuclear scope of $every'$ is the $some'$ proposition.
$\Rightarrow$ $every'$ can take scope over $some'$ but if it does so, then it has to take immediate scope over $some'$.

Restrictions on Quantifier Scope (9)

(32)

$l_1 : spy'(x, y)$,

$l_2 : 2(x, \boxed{3}, \boxed{4})$, $l_3 : policeman'(x)$

$l_4 : some'(y, \boxed{7}, \boxed{8})$, $l_5 : person'(y) \wedge \boxed{18}$,

$l_7 : from'(y, z)$

$l_8 : every'(z, \boxed{13}, \boxed{14})$, $l_9 : city'(z)$

$\boxed{1} \geq l_1$,

$\boxed{3} \geq l_3$, $\boxed{1} \geq \boxed{4}$, $\boxed{4} \geq l_1$,

$\boxed{7} \geq l_5$, $\boxed{1} \geq \boxed{8}$, $\boxed{8} \geq l_1$

$\boxed{18} \geq l_7$,

$\boxed{13} \geq l_9$, $l_4 \geq \boxed{14}$, $\boxed{14} \geq l_7$

Restrictions on Quantifier Scope (10)

LRS analysis:

As with the scope island constraints for finite clauses discussed above, LRS imposes a construction specific scope constraint to enforce the relevant restriction.

Quantified Complex NP Constraint

In a complex quantified NP, embedded quantifiers may only outscope the quantifier of the NP if they take immediate scope over it.

Restrictions on Quantifier Scope (11)

Summary: The two theories differ with respect to their treatment of imposing quantifier scope restrictions in the empirical domains we considered.

- LTAG semantics prevents the quantifiers from taking wide scope by restricting them lexically.
- LRS formulates conditions in principles of grammar which are formulated with respect to certain syntactic environments (finite clauses, complex quantified NPs,. . .)

1. Constraint-based computational semantics
2. Specific analyses in LRS and LTAG
   (a) Quantifier Scope
   (b) Restrictions for Quantifier Scope
   (c) Negative Concord
   (d) Negative Polarity Items
3. Compositionality
4. Comparing the frameworks

Negative Concord (1)

Polish:

(33) Janek nie pomaga ojcu.

Janek NM help father

'... Janek doesn't help his father'

(34) a. Janek nie pomaga nikomu.

Janek NM help nobody

'... Janek doesn't help anybody'

b. *Janek pomaga nikomu.

Only one negation in (34a.).

Negative Concord(2)

LRS analysis

- "nie" introduces a negation and every N-word introduces a negation
- crucial: in LRS, descriptions of formulas ⇒ identifications of different negations possible
- Negation Complexity Constraint: makes sure there is only one negation per clause taking scope over the main verb (language specific)
- Neg First Principle: if there is a negation and a verb, then the negation must be on the verb's PARTS list (i.e., verbal prefix "nie" obligatory)

Negative Concord in Polish (3)

$$\begin{bmatrix} \text{EXC} & \boxed{6} \\ \text{INC} & \boxed{1} \\ \text{P} & \langle e, x, \boxed{0}, \boxed{1}, \boxed{1a}, \boxed{1b}, \boxed{2}, \boxed{3}, \boxed{3a}, \boxed{4}, \boxed{5}, \boxed{5a}, \rangle \end{bmatrix}$$

nikt

$$\begin{bmatrix} \text{EXC} & \boxed{5}\exists x : \gamma \wedge \delta \\ \text{INC} & \boxed{3}human'x \\ \text{P} & \langle x, \boxed{3}, \boxed{3a}human', \\ & \boxed{4}\neg\beta, \boxed{5}, \boxed{5a}\gamma \wedge \delta, \rangle \end{bmatrix}$$

nie przyszedł

$$\begin{bmatrix} \text{EXC} & \boxed{6} \\ \text{INC} & \boxed{1}come'e, x \\ \text{P} & \langle e, x, \boxed{1}, \boxed{1a}come'e, \\ & \boxed{1b}come', \boxed{2}\neg\alpha, \boxed{0}\exists e\phi\rangle \end{bmatrix}$$

$\boxed{1} \lhd \alpha$, $\boxed{2} \lhd \boxed{6}$, $\boxed{5} \lhd \beta$, $\boxed{3} \lhd \gamma$, $\boxed{1} \lhd \delta$, $\boxed{1} \lhd \phi$, $\boxed{1} \lhd \alpha$

Negative Concord (4)

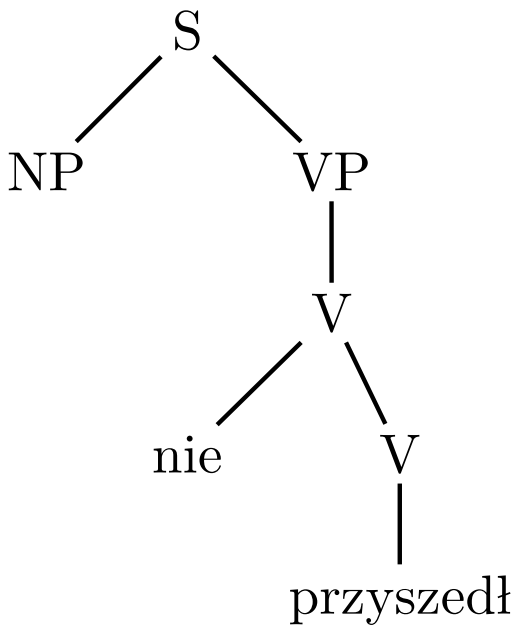
LTAG analysis:

- only one negation introduced, coming from the negated verb.
- N-word as existential quantifier in the scope of the negation
- N-word needs licensing negation

Consider

(35) nikt nie przyszedł.
nobody NM comes
'Nobody comes'

Negative Concord (5)



$$l_1 : \neg\boxed{1},\ l_2 : come'(\boxed{2},\boxed{3})$$

$$\boxed{1} \geq l_2,\ \boxed{4} \geq l_1$$

$$\begin{bmatrix} \text{GLOBAL} & \begin{bmatrix} \text{MAXS} & \boxed{4} \\ \text{N-SCOPE} & \boxed{1} \\ \text{MINS} & l_2 \\ \text{NEG} & \text{yes} \end{bmatrix} \\ \text{NP} & \begin{bmatrix} \text{GLOBAL} & \begin{bmatrix} \text{I} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

NEG marks verb as negated

N-SCOPE provides scope of negation as maximal scope of N-words.

Negative Concord (6)

NP
|
nikt

$$\boxed{\begin{array}{l} l_3 : some'(x, \boxed{5}, \boxed{6}), \\ l_4 : person'(x) \\ \boxed{5} \geq l_4, \\ \boxed{7} \geq \boxed{6}, \boxed{6} \geq \boxed{8} \end{array}}$$

$$\left[\begin{array}{ll} \text{GLOBAL} & \left[\begin{array}{ll} \text{I} & \text{x} \end{array}\right] \\ \text{NP} & \left[\begin{array}{ll} \text{GLOBAL} & \left[\begin{array}{l} \text{N-SCOPE } \boxed{7} \\ \text{MINS } \boxed{8} \\ \text{NEG YES} \end{array}\right] \end{array}\right] \end{array}\right]$$

N-word requires a negated verb and takes the N-SCOPE of the verb as maximal scope.

Negative Concord (7)

Summary: LRS' flexible and powerful feature logic allows to introduce different descriptions of the same formula in different places of the analysis.

LTAG tries to stay close to normal dominance constraints. Therefore, each subformula of the final semantic representation is introduced exactly once.

1. Constraint-based computational semantics
2. Specific analyses in LRS and LTAG
   (a) Quantifier Scope
   (b) Restrictions for Quantifier Scope
   (c) Negative Concord
   (d) Negative Polarity Items
3. Compositionality
4. Comparing the frameworks

Negative Polarity Items (1)

(36) a. He has**n't** seen <u>any</u> students.

b. *He has seen <u>any</u> students.

(37) a. Es <u>schert</u> ihn **nicht**

it bothers him not

'He does not give a damn about it'

b. *Es <u>schert</u> ihn.

Negative Polarity Items (2)

Licensing of NPIs obeys the following conditions (cf. Linebarger's Immediate Scope Constraint):

- **1. licensing condition (L1):**
  NPIs are in the scope of a negation that is semantically interpreted in the same finite clause.
- **2. licensing condition (L2):**
  No regular quantifier is scopally intervening between the negation and the NPI.

Negative Polarity Items (3)

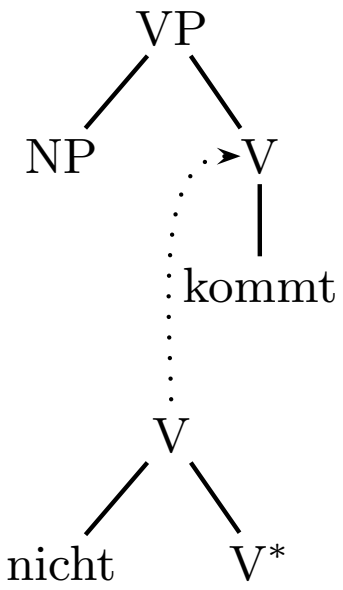
LTAG analysis (Lichte & Kallmeyer 2006):

1. global feature N-SCOPE: indicates the scope of a negation
2. global feature NEG: indicates whether a tree semantically contains a negation
3. local feature NEG: indicates the presence of a negation at the verbal spine
4. global feature MINP for the minimal proposition coming with a verb

Negative Polarity Items (4)

(38) dass Hans nicht kommt

that John not comes



$l_1 : come'(\boxed{1})$

v

$l_2 : \neg\boxed{7}$

$\boxed{7} \geq \boxed{8}$

$$\begin{bmatrix} \text{GLOBAL} & \begin{bmatrix} \text{N-SCOPE} & \boxed{2} \\ \text{MINP} & l_1 \\ \text{NEG} & \boxed{3} \end{bmatrix} \\ \text{VP} & \begin{bmatrix} \text{T} & \begin{bmatrix} \text{NEG} & \boxed{3} \end{bmatrix} \\ \text{B} & \begin{bmatrix} \text{NEG} & \boxed{4} \end{bmatrix} \end{bmatrix} \\ \text{V} & \begin{bmatrix} \text{T} & \begin{bmatrix} \text{NEG} & \boxed{4} \end{bmatrix} \\ \text{B} & \begin{bmatrix} \text{NEG} & \text{no} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \text{V}_r & \begin{bmatrix} \text{B} & \begin{bmatrix} \text{NEG} & \text{yes} \end{bmatrix} \end{bmatrix} \\ \text{V}_f & \begin{bmatrix} \text{GLOBAL} & \begin{bmatrix} \text{N-SCOPE} & \boxed{7} \\ \text{MINP} & \boxed{8} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Negative Polarity Items (5)

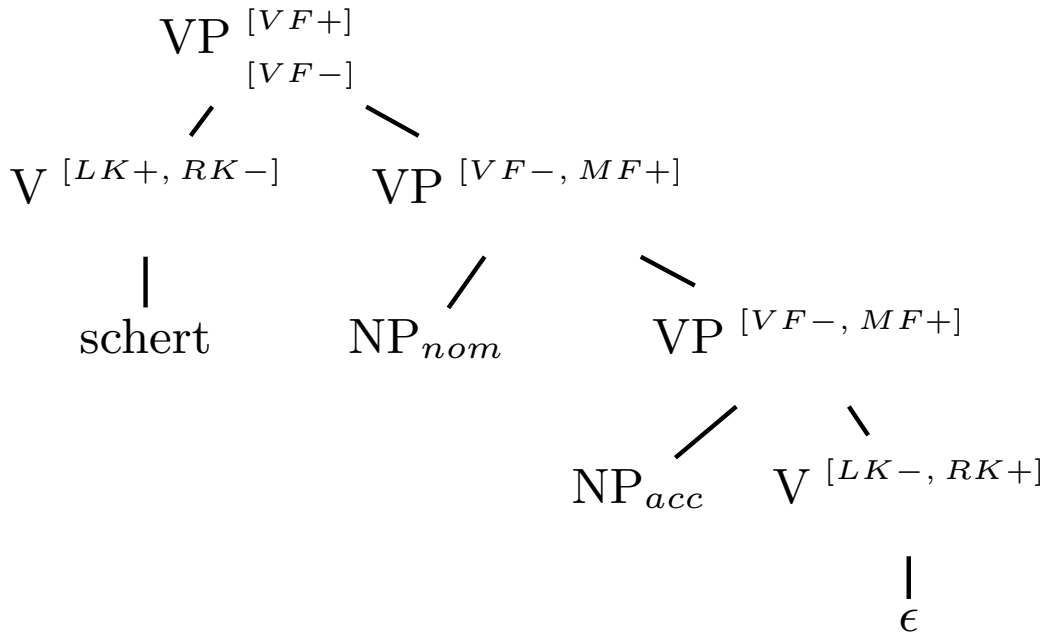
(39) Es <u>schert</u> ihn **nicht**

it bothers him not

'He does not give a damn about it'

Lexical entry of the NPI *schert* ('to give a damn about'):

Elementary tree:



Negative Polarity Items (6)

$l_1 : scheren'(\boxed{1}, \boxed{2})$

$\boxed{7} \geq \boxed{8}, \boxed{8} \geq l_1,$

$\boxed{8} \geq \boxed{9}, \boxed{9} \geq l_1$

1. negation required (NEG)
2. NPI in scope of negation:

   N-SCOPE $\geq$ MINP
3. quantifiers scope over negation:

   MINS $\geq$ N-SCOPE, i.e., no quantifier scopes between neg and NPI

$$\begin{bmatrix}
\text{GLOBAL} & \begin{bmatrix} \text{MINP} & l_1 \\ \text{MINS} & \boxed{8} \\ \text{N-SCOPE} & \boxed{9} \\ \text{NEG} & \text{yes} \end{bmatrix} \\
\text{VP}_\epsilon & \begin{bmatrix} \text{T} & \begin{bmatrix} \text{NEG} & \text{yes} \end{bmatrix} \\ \text{B} & \begin{bmatrix} \text{NEG} & \boxed{4} \end{bmatrix} \end{bmatrix} \\
\text{VP}_2 & \begin{bmatrix} \text{T} & \begin{bmatrix} \text{NEG} & \boxed{4} \end{bmatrix} \\ \text{B} & \begin{bmatrix} \text{NEG} & \boxed{5} \end{bmatrix} \end{bmatrix} \\
\text{VP}_{22} & \begin{bmatrix} \text{T} & \begin{bmatrix} \text{NEG} & \boxed{5} \end{bmatrix} \\ \text{B} & \begin{bmatrix} \text{NEG} & \boxed{6} \end{bmatrix} \end{bmatrix} \\
\text{V} & \begin{bmatrix} \text{T} & \begin{bmatrix} \text{NEG} & \boxed{6} \end{bmatrix} \\ \text{B} & \begin{bmatrix} \text{NEG} & \text{no} \end{bmatrix} \end{bmatrix}
\end{bmatrix}$$

Negative Polarity Items (7)

The LRS analysis of NPI licensing of Richter & Soehn (2006) encompasses several modules of grammar. The following components are most prominent:

- A collocation module of grammar restricting the distribution of NPIs to their idiosyncratic licensing domains. The licensing domain can be syntactic, semantic and pragmatic (and any combination thereof)
- A (potentially discontinuous) lexical LRS analysis of scope taking licensers (e.g. quantifiers)
- A semantic licenser hierarchy of NPIs, encoded in relations that identify anti-morphic, anti-additive, downward entailing and other semantically relevant environments

Negative Polarity Items (8)

An example of a lexical entry of an NPI in LRS:

$$\begin{bmatrix} \textit{word} \\ \textsc{phon} \ \langle \text{scheren} \rangle \\ \textsc{synsem} \ \begin{bmatrix} \textsc{local} \ \begin{bmatrix} \textsc{cat head} \quad \textit{verb} \\ \textsc{cont main} \ \boxed{1} \ \mathsf{scheren}' \end{bmatrix} \end{bmatrix} \\ \textsc{coll} \quad \left\langle \begin{bmatrix} \textit{complete-clause} \\ \textsc{lf-lic} \ \left[\textsc{exc} \ \texttt{deint-strength-op}(\boxed{1}) \right] \end{bmatrix} \right\rangle \end{bmatrix}$$

Negative Polarity Items (9)

Locality of NPI licensing and of negative concord: Comparison LTAG-LRS.

In NPI licensing, the licenser must occur within a certain local domain. (The same holds for the negative marker in negative concord.)

In LTAG, because of the extended domain of locality, this is straightforward to model. (Though sometimes, the information of the presence of a licenser must percolate because the licensee does not always attach to the verb carrying the licenser.)

In NPI licensing in LRS, the licensing domain must be explicitely specified (in lexical entries). This is achieved by a collcoation theory which enforces appropriate contextual conditions.

1. Constraint-based computational semantics
2. Specific analyses in LRS and LTAG
   (a) Quantifier Scope
   (b) Restrictions for Quantifier Scope
   (c) Negative Concord
   (d) Negative Polarity Items
3. Compositionality
4. Comparing the frameworks

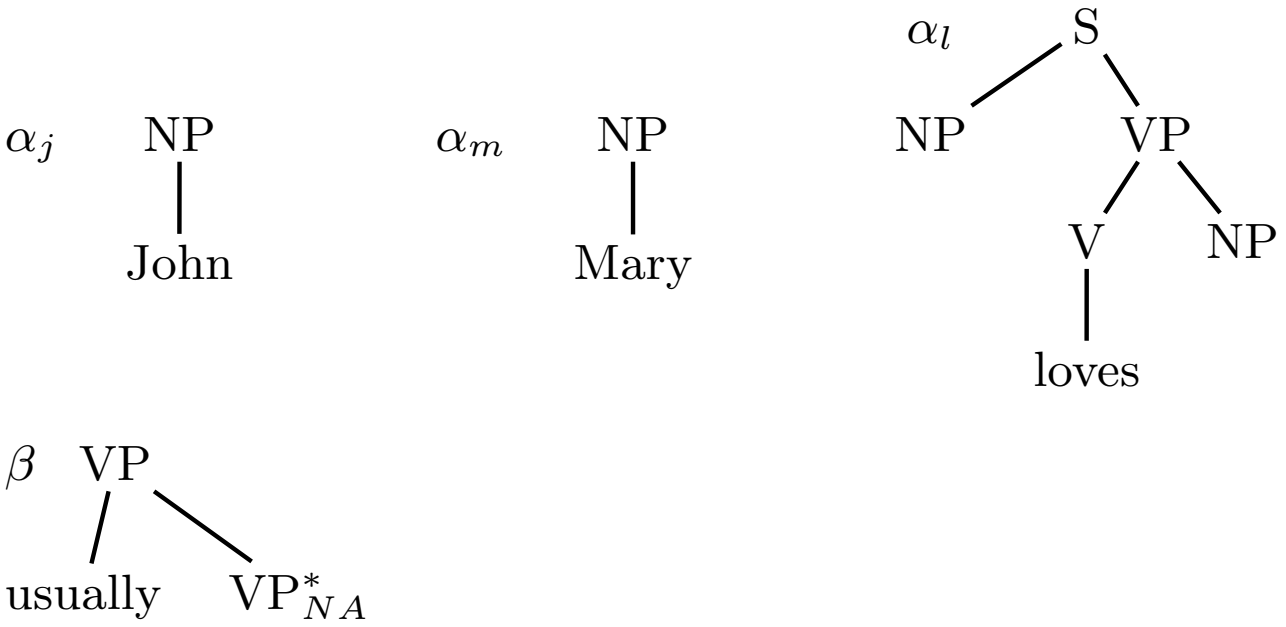
Compositionality (1)

LTAG can be written as a linear context-free rewriting system (LCFRS) consisting of

- a generalized context free grammar (GCFG) generating terms in a term algebra that correspond to the derivation trees,
- the denotations of these terms (the derived trees), and
- functions specifying how to compute the strings they yield.

Compositionality (2)

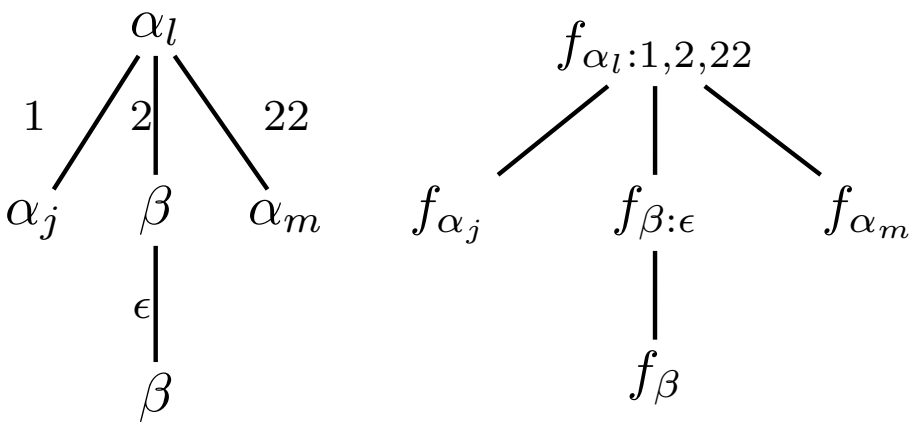
Sample TAG:



Compositionality (3)

Sample derivation tree and corresponding term tree:

(40) John usually usually loves Mary



Compositionality (4)

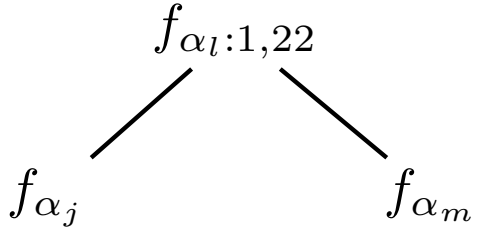
The functions in the term trees specify syntactic compositions.

It is also possible to give them a semantic composition denotion. I.e., the GCFG productions also specify semantic composition.

Example:

(41) John loves Mary

GCFG term tree:



Compositionality (5)

$[\![f_{\alpha_j}()]\!]_{sem} = \langle \sigma_{\alpha_j}, \delta'_{\alpha_j}, \boxed{0} \rangle$ with

$\sigma_{\alpha_j} = \boxed{john'(x)}$

$\delta'_{\alpha_j} = \mathrm{I}(\mathrm{GLOBAL}(\boxed{0})) = x$

$\wedge \mathrm{T}(\epsilon(\boxed{0})) = \mathrm{B}(\epsilon(\boxed{0}))$

$[\![f_{\alpha_m}()]\!]_{sem} = \langle \sigma_{\alpha_m}, \delta'_{\alpha_m}, \boxed{1} \rangle$ with

$\sigma_{\alpha_m} = \boxed{mary'(y)}$

$\delta'_{\alpha_m} = \mathrm{I}(\mathrm{GLOBAL}(\boxed{1})) = y$

$\wedge \mathrm{T}(\epsilon(\boxed{1})) = \mathrm{B}(\epsilon(\boxed{1}))$

Compositionality (6)

$[\![f_{\alpha_l:1,22}(f_{\alpha_j}(), f_{\alpha_m})]\!]_{sem} = \langle \sigma_{\alpha_l}, {\delta_{\alpha_l}}', \boxed{4} \rangle$ with

$$\sigma_{\alpha_l} = \boxed{john'(x),\ mary'(y),\ l_1 : love'(x,y)}$$

$$\begin{aligned}
\delta'_{\alpha_l} = {} & \mathrm{I}(\mathrm{GLOBAL}(\mathrm{NP1}(\boxed{4}))) = \boxed{2} \wedge \mathrm{I}(\mathrm{GLOBAL}(\mathrm{NP2}(\boxed{4}))) = \boxed{3} \\
& \wedge \mathrm{P}(\mathrm{T}(\mathrm{VP}(\boxed{4}))) = \boxed{5} \wedge \mathrm{P}(\mathrm{B}(\mathrm{VP}(\boxed{4}))) = l_1 \\
& \wedge \delta'_{\alpha_j} \wedge \delta'_{\alpha_m} \\
& \wedge \mathrm{GLOBAL}(\boxed{0}) = \mathrm{GLOBAL}(\mathrm{NP1}(\boxed{4})) \\
& \wedge \mathrm{GLOBAL}(\boxed{1}) = \mathrm{GLOBAL}(\mathrm{NP2}(\boxed{4})) \\
& \wedge \mathrm{T}(\mathrm{S}(\boxed{4})) = \mathrm{B}(\mathrm{S}(\boxed{4})) \wedge \mathrm{T}(\mathrm{VP}(\boxed{4})) = \mathrm{B}(\mathrm{VP}(\boxed{4})) \\
& \wedge \mathrm{T}(\mathrm{V}(\boxed{4})) = \mathrm{B}(\mathrm{V}(\boxed{4}))
\end{aligned}$$

Compositionality (7)

Summary: the context-free LTAG derivation tree uniquely specifies not only syntactic composition but also semantic composition.

$\Rightarrow$ even including semantic computation, the grammar remains mildly context-sensitive.

The semantic denotation of a node in the derivation tree depends only the denotations of the daughters, the semantic representation from the lexicon chosen for this node and the way the daughters combine with the mother. In this sense, LTAG semantics is compositional.

Comparison and Results

Summing up: Similarities

What LRS and LTAG have in common:

- Ty2 language for semantics
- scope constraints $\geq$ in LTAG and component-of constraints $\lhd$ in LRS for scope ambiguities
- scope window to delimit quantifier scope: MAXS and MINS in LTAG, EXCONT and INCONT in LRS
- feature logic for specifying and computing semantic representations

Summing up: Differences

Differences between LRS and LTAG:

- Extended domain of locality in LTAG
- 'Global' principles in LRS vs. lexicalization in LTAG
- Interleaved specification of syntax and semantics in LRS
  vs. more sequential architecture in LTAG
- LRS: (partial) descriptions of fully specified models vs.
  LTAG: Underspecified representations in the style of Bos with subsequent disambiguation (pluggings)
- LRS: Does not contain formulas but descriptions of formulas $\Rightarrow$ different descriptions can denote the same formula

  LTAG: Two formulas in semantic representations are always distinct syntactic objects

Summing up: Architectures at a Glance

1. Architecture:
   - LTAG: Trees + tree building operations + (underspec.) semantic representations + FL decorations
   - HPSG: Uniform FL specification with model-theoretic interpretation
2. Implementation:
   - LTAG: Formalism mildly context sensitive
   - LRS: Implementation in separate, denotationally equivalent constraint language
3. Feature Logic:
   - LTAG: FL as computational glue
   - LRS: Expressive FL for interleaving syntax and semantics

Consequences for Model-theoretic Semantics (1)

- Semantic argument identification by feature value identification instead of higher order type shifting
- Combination of scope windows and underspecification:
  - No complex LF movement for the sake of describing scoping possibilities of quantifiers and operators
    → questions about restrictions on pseudo-syntactic movement don't even arise
  - Local specification of the scope potential of scope taking elements
  - Clear separation of the scope taking part of the semantics of words and their 'nuclear' semantic contribution (e.g. 'nobody')

Consequences for Model-theoretic Semantics (2)

- Compositionality
  - Compositionality is not given by a homomorphism from ('derived') syntactic tree structure to a semantic algebra
  - The systems can still be compositional (as sketched already for LTAG semantics)
- Computational properties
  - Polynomial parsability in LTAG
  - LRS implemented in the CLLRS module of TRALE
  - Claim: Logical representations qualify as semantic only to the extent that they support inference
  - Idea: Use underspecification and the type system to guide constraint-solving and to support inferencing

Thank You