Constraint-based Computational Semantics: LTAG and LRS

Laura Kallmeyer, SFB 441, University of Tübingen

Cogeti Meeting, Tübingen, 5 February 2006

Overview

1. Lexicalized Tree Adjoining Grammars (LTAG)
   (a) Adjunction and substitution
   (b) The nature of elementary trees
   (c) Constituency and dependency
2. LTAG Semantics with Semantic Unification
   (a) Semantic Unification
   (b) Global features
3. LTAG and LRS
   (a) Quantifier Scope
   (b) Negative Concord
4. Conclusion: Comparing the frameworks

Lexicalized Tree Adjoining Grammars (LTAG)

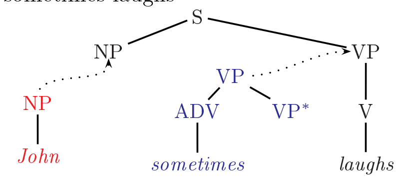
Tree Adjoining Grammars (1)

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

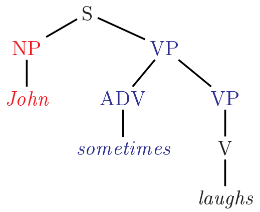
- *adjunction*: replacing an internal node with a new tree.
  The new tree is an *auxiliary* tree and has a special leaf, the *foot node*.
- *substitution*: replacing a leaf with a new tree.
  The new tree is an *initial* tree

Tree Adjoining Grammars (2)

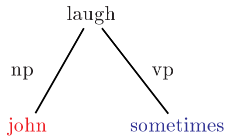
(1) John sometimes laughs



derived tree:



derivation tree:



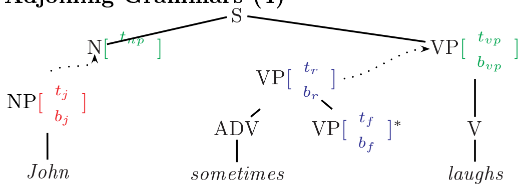
Tree Adjoining Grammars (3)

Feature-structure based TAG (FTAG): 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).

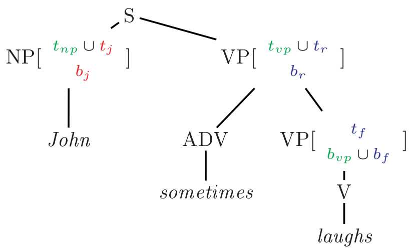
Unification during derivation:

- 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.

Tree Adjoining Grammars (4)



derived tree:



Tree Adjoining Grammars (5)

Formal properties of TAG:

TAGs are mildly context-sensitive:

- TAGs are slightly more powerful than CFG, they can describe cross-serial dependencies.
- TAGs are polynomially parsable (complexity $O(n^6)$).
- TALs are of constant growth.

Elementary trees (1)

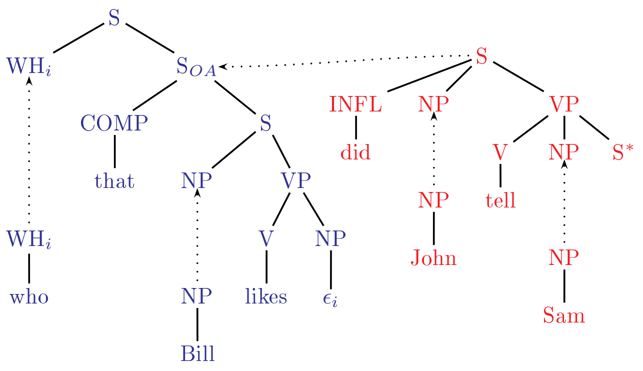
Important features of LTAG:

- 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 (3)

(2) 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.
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.

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.

Elementary trees (4)

In the lexicon, the trees are organized in tree families.

Each family contains a base tree and trees derived from the base tree using transformations.

Important: These transformations operate only on a finite set, i.e., on structures of bounded size.

Constituency and Dependency (1)

The derived tree gives the constituent structure.

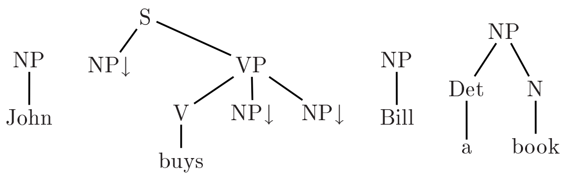
The derivation tree records the history of how the elementary trees are put together.
⇒ the edges in the derivation tree represent predicate-argument dependencies; the derivation tree is close to a semantic dependency graph.

⇒ compute semantics on derivation tree

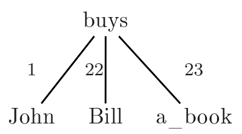
Constituency and Dependency (2)

(3) John buys Bill a book

Elementary trees:

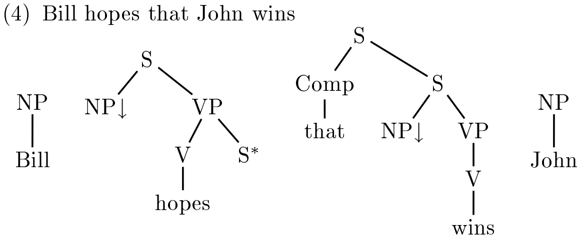


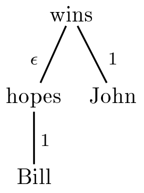
Derivation tree



Constituency and Dependency (3)

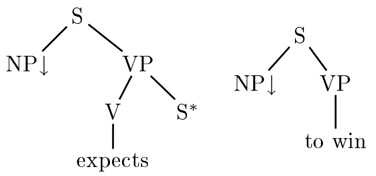
(4) Bill hopes that John wins

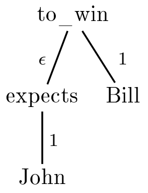




Constituency and Dependency (4)

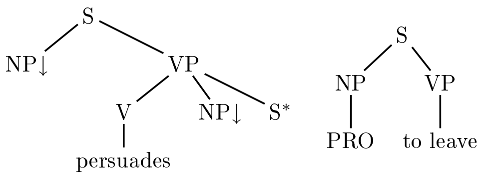
(5) John expects [Bill to win]

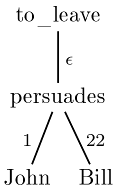




Constituency and Dependency (5)

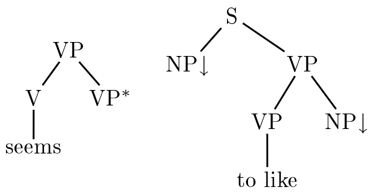
(6) John persuades Bill [PRO to leave]

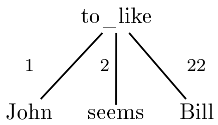




Constituency and Dependency (6)

(7) John seems to like Bill





LTAG Semantics with Semantic Unification

joint work with Maribel Romero,

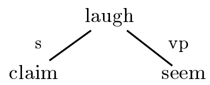
University of Pennsylvania

Semantic Unification (1)

Problems where derivation tree does not provide all necessary links:

(8) a. Mary claims John seems to laugh
b. Paul claims Mary apparently laughs

Derivation tree:

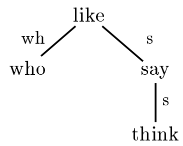


$\mathit{claim}'(m, (\mathit{seem}'(\mathit{laugh}'(j))))$

no link between *claim* and *seem*

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

Derivation tree:



$\mathit{who}'(x, \mathit{think}'(p, \mathit{say}'(j, \mathit{like}'(b, x))))$

no link between *like* and *think*

(neither between *who* and *think*)

Semantic Unification (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 Unification (3)

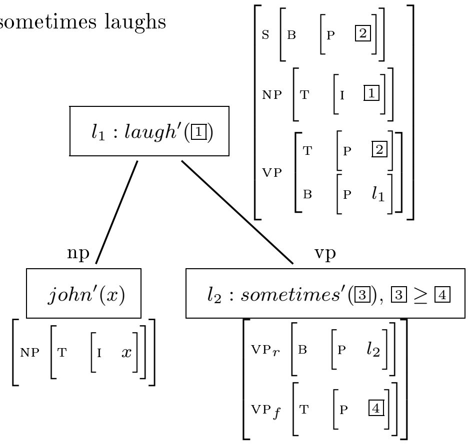
To separate more neatly between syntax and semantics, augment the derivation tree with semantic feature structure descriptions and do semantic unification on the derivation tree :

- semantic representations and semantic feature structure descriptions linked to elementary trees.
- semantic composition = unification (i.e., feature value identifications) between these feature structure descriptions.

The feature structures link top and bottom features to the nodes in the elementary tree. Unification proceeds parallel to syntactic feature structure unification.

Semantic Unification (4)

(1) John sometimes laughs



Semantic Unification (5)

Definition of the feature structures: The feature structures are typed.

Signature:

$$_{sem}\begin{bmatrix} 0 & tb \\ 1 & tb \\ \dots & \end{bmatrix} \qquad _{tb}\begin{bmatrix} \text{T} & bindings \\ \text{B} & bindings \end{bmatrix} \qquad _{bindings}\begin{bmatrix} \text{I} & var_e \\ \text{P} & var_{\langle s,t\rangle} \\ \text{S} & var_s \end{bmatrix}$$

The features in *sem* are all node positions that can occur in elementary trees (finite for each TAG).

- Constants of type var_e: all individual variables (in our language for semantic representations),
- Constants of type $var_{\langle s,t\rangle}$: all propositional labels,
- Constants of type var_s: all situation variables.

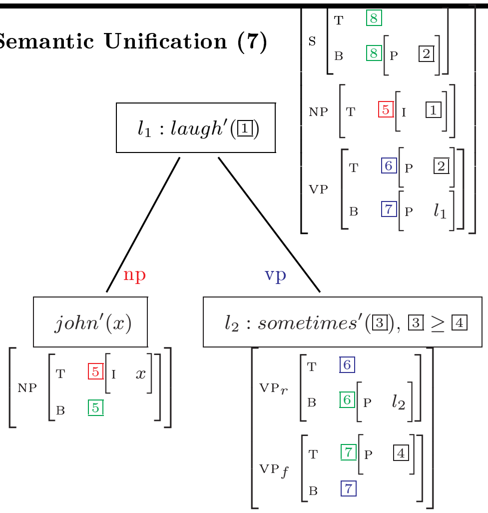
Semantic Unification (6)

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 (7)



Semantic Unification (8)

Identifications: $\boxed{1} = x$ (substitution), $\boxed{2} = l_2, \boxed{4} = l_1$ (adjunction and top-bottom unification)

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

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

Underspecified representation.

Semantic Unification (9)

Disambiguation: Function δ from the set of propositional metavariables to the set of propositional labels such that after applying it, the resulting subordination is a partial order.

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

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

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

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)

Revised signature:

$$sem\begin{bmatrix} \textsc{global} & global \\ 0 & node \\ 1 & node \\ \dots & \end{bmatrix} \qquad node\begin{bmatrix} \textsc{global} & global \\ \textsc{t} & bindings \\ \textsc{b} & bindings \end{bmatrix}$$

$$bindings\begin{bmatrix} \textsc{p} & var_{\langle s,t\rangle} \\ \textsc{s} & var_s \end{bmatrix} \qquad global\begin{bmatrix} \textsc{i} & var_e \\ \textsc{maxs} & var_{\langle s,t\rangle} \\ \textsc{mins} & var_{\langle s,t\rangle} \end{bmatrix}$$

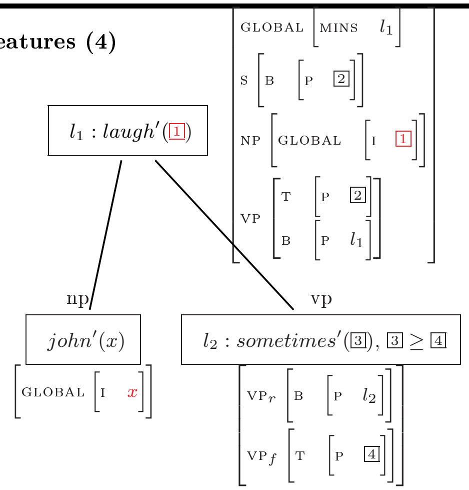
Global features (3)

Unification of global features:

Depending on the nature of the node n in an elementary tree that has a feature GLOBAL, this feature is identified with the GLOBAL of the mother or of a daughter in the derivation tree:

- If n is an argument slot (a substitution node/a foot node), its feature GLOBAL is identified with the feature GLOBAL of the argument tree (the substituted initial tree/the tree the auxiliary tree adjoins to).
- If n is the root node of an initial tree, its feature GLOBAL is identified with the feature GLOBAL of the tree this initial tree attaches to. In all other cases n's feature GLOBAL is identified with the GLOBAL of any auxiliary tree adjoining at n.

Global features (4)



LTAG and LRS

Quantifier Scope (1)

Quantificational NPs: can in principle scope freely; their scope is not directly linked to their surface position.
Quantificational elements attached to the verbal spine (adverbs, raising verbs, attitudes verbs, control verbs): scope over everything that is lower on the verbal spine.

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

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

(12) John seems to sometimes laugh:
$\mathit{seem} > \mathit{sometimes}, \mathit{sometimes} \not> \mathit{seem}$

(13) John seems to have visited everybody: $\mathit{seem} > \forall, \forall > \mathit{seem}$

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)

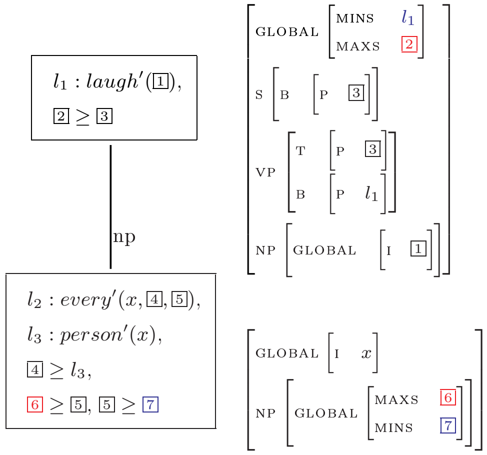
LTAG analysis: global features MAXS and MINS.

NP
|
everybody

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

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

Quantifier Scope (4)



Quantifier Scope (5)

Result:

$l_1 : laugh'(x)$,
$l_2 : every'(x, \boxed{4}, \boxed{5})$, $l_3 : person'(x)$
$\boxed{2} \geq l_1$,
$\boxed{4} \geq l_3, \boxed{2} \geq \boxed{5}, \boxed{5} \geq l_1$

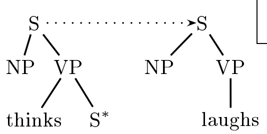
Disambiguation:

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

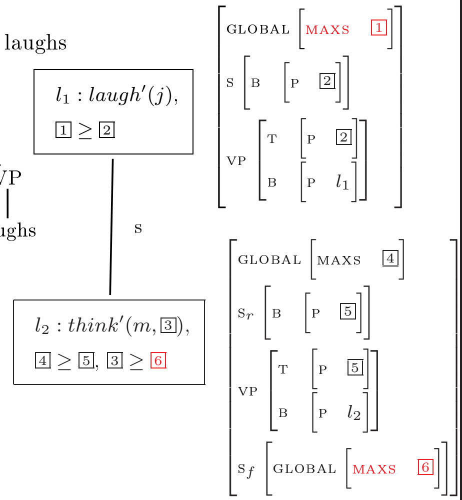
yields $every'(x, person'(x), laugh'(x))$

Quantifier Scope (6)

(14) Mary thinks John laughs

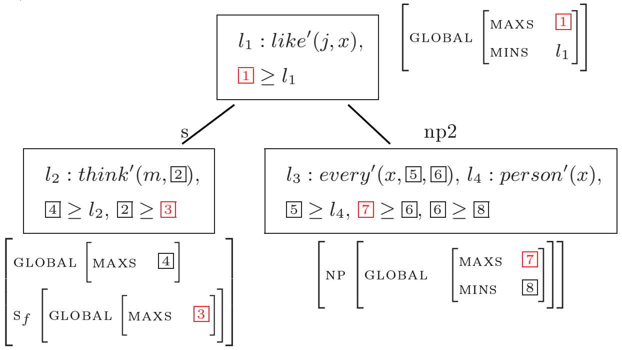


Argument of attitude verb embeds MAXS of embedded verb



Quantifier Scope (7)

(15) Mary thinks John likes everybody



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

Quantifier Scope (8)

LRS analysis:

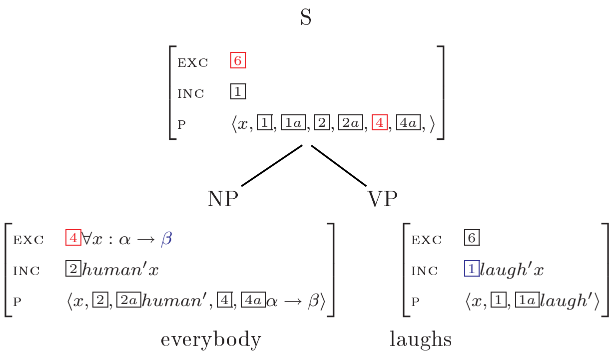
EXCONT PRINCIPLE:
a) The PARTS list of a phrase contains the EXCONT of the non-head daughter.
b) In an utterance, everything on the PARTS list is a component of the EXCONT.

$\Rightarrow$ maximal scope of a quantifier: EXCONT of the next embedding utterance.

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

$\Rightarrow$ minimal scope of a quantifier: INCONT of the verb it combines with.

Quantifier Scope (9)



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

Quantifier Scope (10)

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.

Reason for LTAG choice: scope constraints for nested quantifiers

Quantifier Scope (10)

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

Only four possible scope orders: the two orders where *someone* and *every* do not have immediate scope over each other are not possible.

**someone* > *two* > *every*
**every* > *two* > *someone*

The first order is excluded anyway since it does not yield a tree-shaped term: In order to obtain this order, *every'* must be at the same time in the restriction and in the nuclear scope of *some'*.

Quantifier Scope (11)

The order one needs to exclude: inverse linking order

**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 $some'$.

Negative Concord in Polish (1)

(17)Janek nie pomaga oicu.

Janek NM helped father

'... Janek didn't help his father'

(18) a.Janek nie pomaga nikomu.

Janek NM helped nobody

'... Janek didn't help anybody'

b. *Janek pomaga nikomu.

Only one negation in (18a.).

Negative Concord in Polish (2)

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 licencing negation

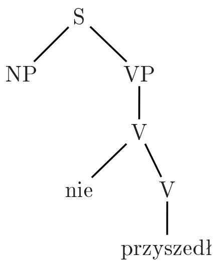
Consider

(19)nikt nie przyszedł.

nobody NM comes

'Nobody comes'

Negative Concord in Polish (3)



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

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

$$\left[\begin{array}{ll} \text{GLOBAL} & \left[\begin{array}{ll} \text{MAXS} & \boxed{4} \\ \text{N-MAXS} & \boxed{1} \\ \text{MINS} & l_2 \\ \text{NEG} & \text{yes} \end{array}\right] \\ \text{NP} & \left[\text{GLOBAL} \left[\text{I} \quad \boxed{2}\right]\right] \end{array}\right]$$

NEG marks verb as negated

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

Negative Concord in Polish (4)

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

$$\begin{array}{c} \text{NP} \\ | \\ \text{nikt} \end{array} \quad \left[\begin{array}{ll} \textsc{global} & \left[\begin{array}{ll} \textsc{i} & \textsc{x} \end{array}\right] \\ \textsc{np} & \left[\begin{array}{ll} \textsc{global} & \left[\begin{array}{ll} \textsc{n-maxs} & \boxed{7} \\ \textsc{mins} & \boxed{8} \\ \textsc{neg} & \textsc{yes} \end{array}\right] \end{array}\right] \end{array}\right]$$

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

Negative Concord in Polish (5)

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 (language specific)
- NEG FIRST PRINCIPLE: if there is a negation and a verb, then the negation must be on the verb's part list (i.e., "nie" obligatory)

Negative Concord in Polish (6)

$$\left[\begin{array}{ll} \textsc{exc} & \boxed{6} \\ \textsc{inc} & \boxed{1} \\ \textsc{p} & \langle e, x, \boxed{0}, \boxed{1}, \boxed{1a}, \boxed{1b}, \boxed{2}, \boxed{3}, \boxed{3a}, \boxed{4}, \boxed{5}, \boxed{5a}, \rangle \end{array}\right]$$

nikt

$$\left[\begin{array}{ll} \textsc{exc} & \boxed{5}\exists x : \gamma \wedge \delta \\ \textsc{inc} & \boxed{3} human' x \\ \textsc{p} & \langle x, \boxed{3}, \boxed{3a} human', \\ & \boxed{4} \neg\beta, \boxed{5}, \boxed{5a} \gamma \wedge \delta, \rangle \end{array}\right]$$

nie przyszedł

$$\left[\begin{array}{ll} \textsc{exc} & \boxed{6} \\ \textsc{inc} & \boxed{1} come' e, x \\ \textsc{p} & \langle e, x, \boxed{1}, \boxed{1a} come' e, \\ & \boxed{1b} come', \boxed{2} \neg\alpha, \boxed{0} \exists e \phi \rangle \end{array}\right]$$

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

Conclusion

Comparing the frameworks (1)

Things LRS and LTAG have in common:

- Ty(2) language for semantics
- scope constraints $\geq$ in LTAG and component-of constraints $\triangleleft$ in LRS for scope ambiguities
- scope window to delimit quantifier scope: MAXS and MINS in LTAG, EXCONT and INCONT in LRS

Comparing the frameworks (2)

Differences between LRS and LTAG:

- Extended domain of locality in LTAG
- General principles in LRS vs. lexicalization in LTAG
- LRS: (partial) descriptions of fully specified models vs LTAG: underspecified representations in the style of Bos with later disambiguation (pluggings)
- LRS: does not contain formulas but descriptions of formulas $\Rightarrow$ different descriptions can denote the same formula, in particular: formulas can be identified

  LTAG: different formulas in semantic representations are different objects