Language modeling with tree-adjoining grammars

Day 3 – part I

Kata Balogh & Simon Petitjean

(*Heinrich-Heine-Universität Düsseldorf*)

ESSLLI 2023

University of Ljubljana, 31 July – 4 August, 2023

erc TreeGraSP

Funded by DFG Deutsche Forschungsgemeinschaft German Research Foundation

hhu Heinrich Heine Universität Düsseldorf

Tree templates and tree families

A **tree family**

- is a set of *tree templates*
- represents a subcategorization frame, and
- contains all syntactic configurations the subcategorization frame can be realized in.

Tree templates and tree families

A **tree family**

- is a set of *tree templates*
- represents a subcategorization frame, and
- contains all syntactic configurations the subcategorization frame can be realized in.

Example tree families

- intransitive: `Tnx0V`

  tree templates: base tree, wh-moved subject, imperative, determiner gerund, ... etc.

Tree templates and tree families

A **tree family**

- is a set of *tree templates*
- represents a subcategorization frame, and
- contains all syntactic configurations the subcategorization frame can be realized in.

Example tree families

- intransitive: `Tnx0V`

  tree templates: base tree, wh-moved subject, imperative, determiner gerund, ... etc.

- transitive: `Tnx0Vnx1`

  tree templates: base tree, passive with *by*, wh-moved subject, wh-moved object, imperative, determiner gerund, ... etc.

Example syntactic phenomenon: extraction

- certain constructions permit an element in one position to fill the grammatical role associated with another position

Example syntactic phenomenon: extraction

- certain constructions permit an element in one position to fill the grammatical role associated with another position
- the positions can be arbitrarily far apart

Example syntactic phenomenon: extraction

- certain constructions permit an element in one position to fill the grammatical role associated with another position
- the positions can be arbitrarily far apart
- filler-gap constructions, e.g.
  - topicalization
  - wh-movement

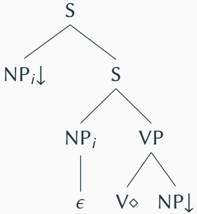
Example syntactic phenomenon: extraction

- certain constructions permit an element in one position to fill the grammatical role associated with another position
- the positions can be arbitrarily far apart
- filler-gap constructions, e.g.
  - topicalization
  - wh-movement
- long-distance dependencies ⇒ **extraction**
  - subject extraction (α`W0nx0V`)
  - object extraction (α`W1nx0Vnx1`)
  - preposition stranding (α`W1nx0VPnx1`)
  - AP complement extraction (α`W1nx0Vnx1`)

Extraction: tree templates

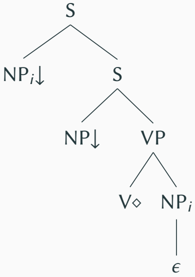
subject extraction

(α`W0nx0Vnx1`)



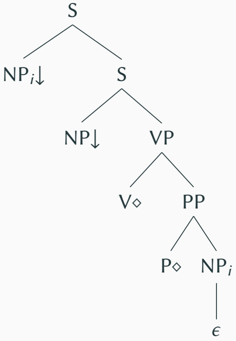
object extraction

(α`W1nx0Vnx1`)



preposition stranding

(α`W1nx0VPnx1`)



Topicalization

Topicalization

Placing a constituent (subject, object, ...) into a sentence-initial position.

Topicalization

Topicalization

Placing a constituent (subject, object, ...) into a sentence-initial position.

(1) a. Pim gave a book to Mia. (base configuration)

b. A book$_i$, Pim gave _$_i$ to Mia. (object NP)

c. Mia$_i$, Pim gave a book to _$_i$. (NP from PP)

d. To Mia$_i$, Pim gave a book _$_i$. (PP)

e. *Pim, _$_i$ gave a book to Mia. (no subject topicalization!)

Topicalization

Topicalization

Placing a constituent (subject, object, ...) into a sentence-initial position.

(1) a. Pim gave a book to Mia. (base configuration)
b. A book$_i$, Pim gave $\_{i}$ to Mia. (object NP)
c. Mia$_i$, Pim gave a book to $\_{i}$. (NP from PP)
d. To Mia$_i$, Pim gave a book $\_{i}$. (PP)
e. *Pim, $\_{i}$ gave a book to Mia. (no subject topicalization!)

- unbounded dependency → the dependency between an extracted constituent and its trace may extend across more *clause boundaries*

Topicalization

Placing a constituent (subject, object, ...) into a sentence-initial position.

(1) a. Pim gave a book to Mia. (base configuration)
b. A book$_i$, Pim gave $\_{}_i$ to Mia. (object NP)
c. Mia$_i$, Pim gave a book to $\_{}_i$. (NP from PP)
d. To Mia$_i$, Pim gave a book $\_{}_i$. (PP)
e. *Pim, $\_{}_i$ gave a book to Mia. (no subject topicalization!)

- unbounded dependency → the dependency between an extracted constituent and its trace may extend across more *clause boundaries*

(2) a. The book$_i$, Bill knows (that) Joe loves $\_{}_i$.
b. The book$_i$, Tom believes (that) Bill knows (that) Joe loves $\_{}_i$.

Wh-constructions

wh-movement

A long-distance extraction of a constituent as a **wh-phrase**.

Wh-constructions

_wh_-movement

A long-distance extraction of a constituent as a **wh-phrase**.

- wh-questions (or constituent questions)

(3) a. *[Who]$_i$ _$_i$ read my book?*
  b. *[What]$_i$ did Joe read _$_i$?*
  c. *[Which book]$_i$ did Pim say Joe had read _$_i$?*

Wh-constructions

wh-movement

A long-distance extraction of a constituent as a **wh-phrase**.

- wh-questions (or constituent questions)

(3) a. *[Who]$_i$ _$_i$ read my book?*
  b. *[What]$_i$ did Joe read _$_i$?*
  c. *[Which book]$_i$ did Pim say Joe had read _$_i$?*

- bounded dependency → island constraints, for example:

(4) Sam knows the student that likes Pim.
  *Whom$_i$ does Sam know the student that likes _$_i$?

Wh-constructions

wh-movement

A long-distance extraction of a constituent as a **wh-phrase**.

- wh-questions (or constituent questions)

(3) a. *[Who]$_i$ _$_i$ read my book?*
b. *[What]$_i$ did Joe read _$_i$?*
c. *[Which book]$_i$ did Pim say Joe had read _$_i$?*

- bounded dependency → island constraints, for example:

(4) Sam knows the student that likes Pim.
*Whom$_i$ does Sam know the student that likes _$_i$?

- *wh*-questions involve **subject-auxiliary inversion**: the auxiliary verb (*do*, *have*, *be*, …) precedes the subject

Subject-auxiliary inversion

- **Obligatory subject-auxiliary inversion** in direct questions with object extraction:

(1) a. What$_i$ **does** John read _$_i$?
 b. *What$_i$ John **does** read _$_i$?
 c. *What$_i$ John reads _$_i$?

Subject-auxiliary inversion

- **Obligatory subject-auxiliary inversion** in direct questions with object extraction:

(1) a. What$_i$ **does** John read _$_i$?
    b. *What$_i$ John **does** read _$_i$?
    c. *What$_i$ John reads _$_i$?

- **No subject-auxiliary inversion** in embedded wh-questions:

(2) a. I wonder [what$_i$ John reads _$_i$].
    b. *I wonder [what$_i$ **does** John read _$_i$].

Subject-auxiliary inversion

- **Obligatory subject-auxiliary inversion** in direct questions with object extraction:

(1) a. What$_i$ **does** John read _$_i$?
 b. *What$_i$ John **does** read _$_i$?
 c. *What$_i$ John reads _$_i$?

- **No subject-auxiliary inversion** in embedded wh-questions:

(2) a. I wonder [what$_i$ John reads _$_i$].
 b. *I wonder [what$_i$ **does** John read _$_i$].

- **No subject-auxiliary inversion** in topicalization:

(3) a. *[The meeting]$_i$, **have** John missed _$_i$.
 b. [This meeting]$_i$ John **have** missed _$_i$.

Extraction: features

Features for extraction:

- <**extracted**> := + | − indicate extraction in the S-node
- <**wh**> := + | − indicate the presence of a wh-pronoun
- <**inv**> := + | − indicate inversion

Extraction: features

Features for extraction:

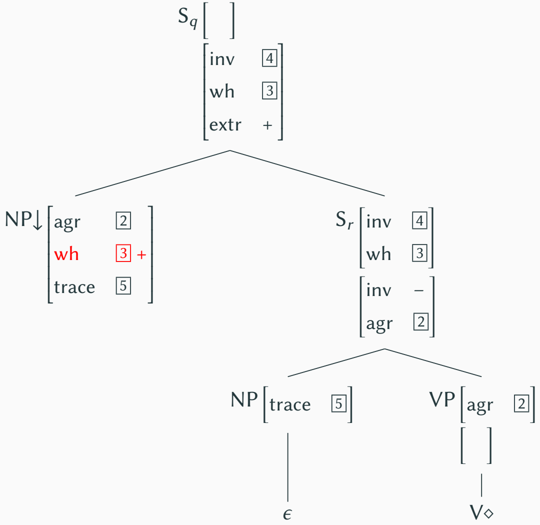
- <**extracted**> := + | − — indicate extraction in the S-node
- <**wh**> := + | − — indicate the presence of a wh-pronoun
- <**inv**> := + | − — indicate inversion

Capturing:

- no inversion with topicalization (*Books$_i$, people read _$_i$.*)
- no topicalized subject (**People$_i$, _$_i$ read books.*)
- no inversion with subject wh-extraction (*Who$_i$ _$_i$ read books?*)
- inversion with object wh-extraction (*What$_i$ do people read _$_i$?*)

Extraction: tree templates with features

Tree template for subject extraction (simplified); αW0nx0V



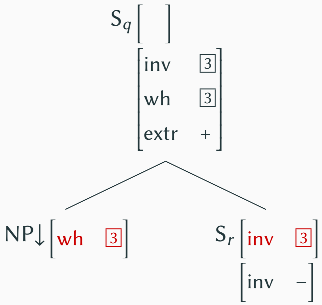
⇒ subject extraction only for *wh*-phrases; no topicalized subject

Inversion with object extraction

- in case of object extraction
  - topicalization → no inversion
  - wh-questions → inversion

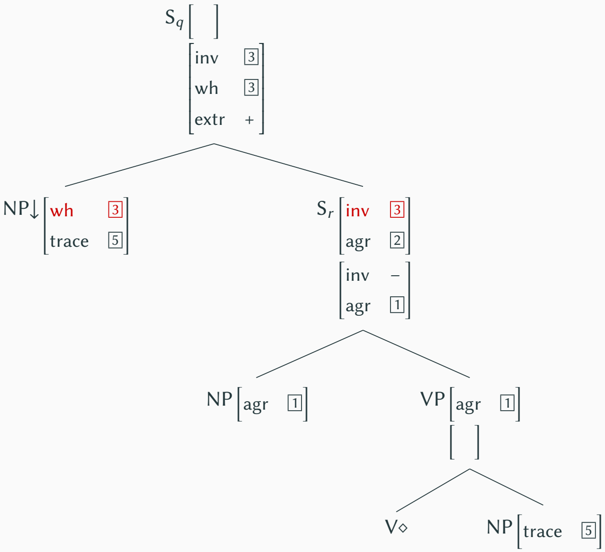
Inversion with object extraction

- in case of object extraction
  - topicalization → no inversion
  - wh-questions → inversion
- ⇒ equation of the values of
  S_r: top.<**inv**> and the extracted NP: top.<**wh**>



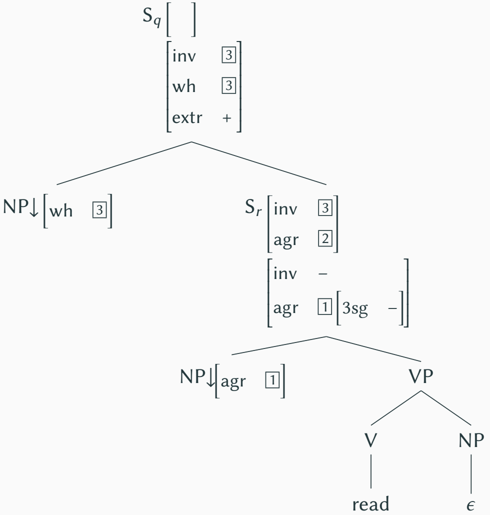
Extraction: tree templates with features

Tree template for object extraction (simplified!); αW1nx0Vnx1

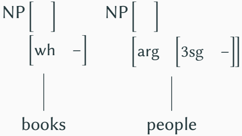


Analyses

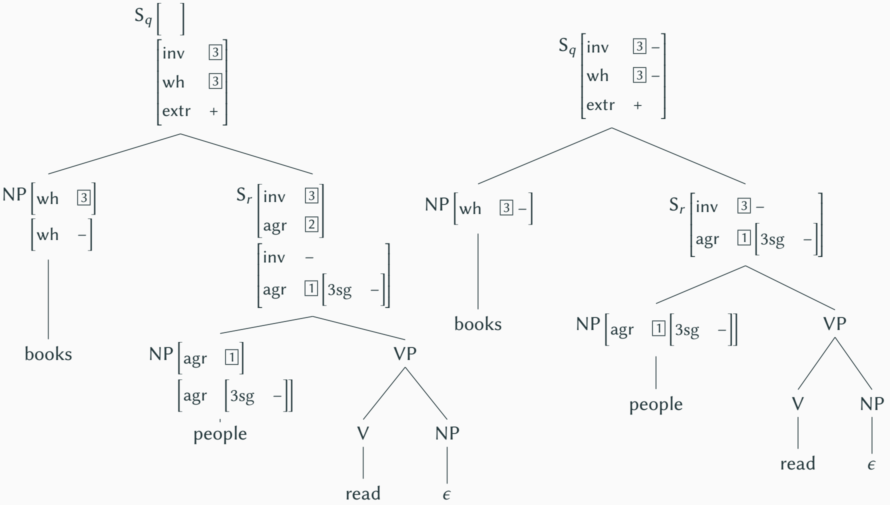
Books, people read.



NP-trees to substitute (subj, obj):

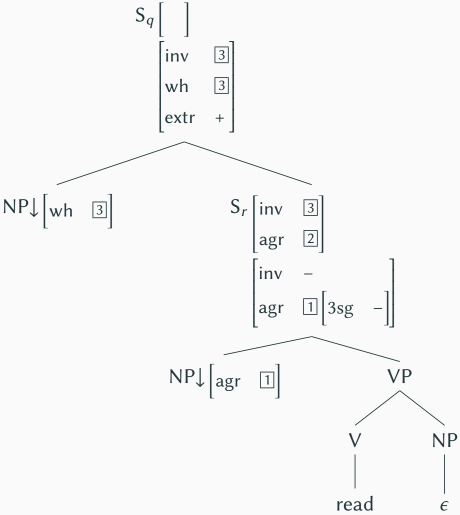


Books, people read.

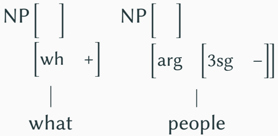


Analyses

What do people read?

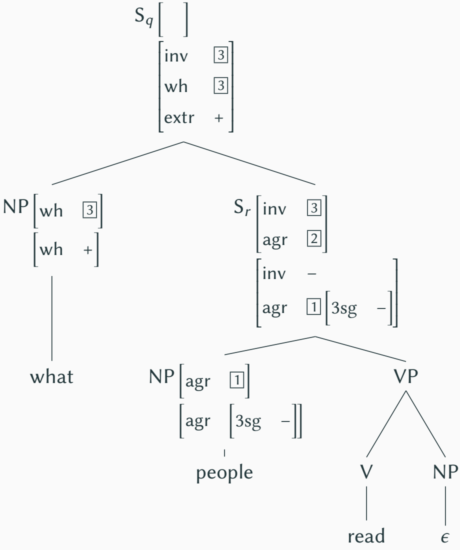


NP-trees to substitute (subj, obj):

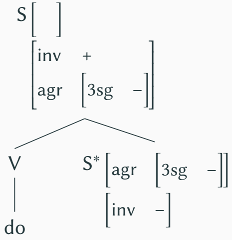


Analyses

What do people read?

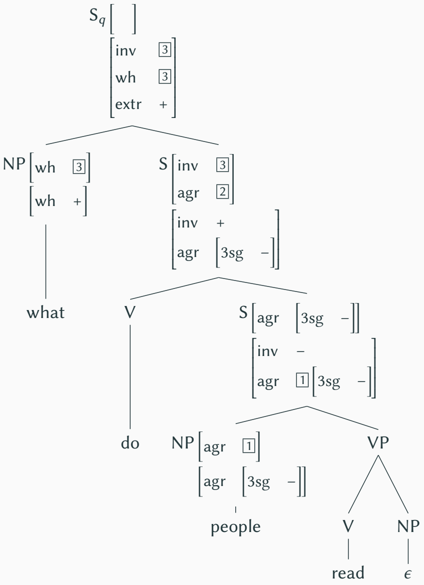


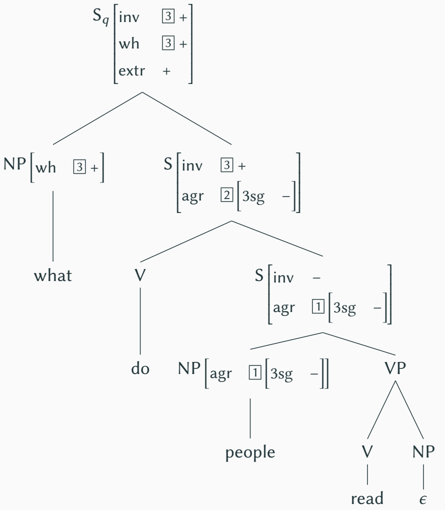
- cannot end the derivation here
- forcing adjunction at S_r
- adjoin the tree of 'do'



Analyses

What do people read?





LTAG semantics: overview

Goal: an LTAG architecture of the syntax-semantics interface that

LTAG semantics: overview

Goal: an LTAG architecture of the syntax-semantics interface that

- is compositional $\rightarrow$ the meaning of a complex expression can be computed from the meaning of its subparts and its composition operation.

LTAG semantics: overview

Goal: an LTAG architecture of the syntax-semantics interface that

- is compositional → the meaning of a complex expression can be computed from the meaning of its subparts and its composition operation.
- pairs entire elementary trees with meaning components

LTAG semantics: overview

Goal: an LTAG architecture of the syntax-semantics interface that

- is compositional → the meaning of a complex expression can be computed from the meaning of its subparts and its composition operation.
- pairs entire elementary trees with meaning components

Three principal approaches:

1. LTAG semantics with synchronous TAG (STAG)

   [Shieber 1994, Nesson & Shieber 2006, 2008]

2. unification based LTAG semantics with predicate logic

   [Kallmeyer & Joshi 2003, Gardent & Kallmeyer 2003, Kallmeyer & Romero 2008]

3. unification based LTAG semantics with frames

   [Kallmeyer & Osswald 2013, Kallmeyer & Osswald & Pogodalla 2016]

Synchronous TAG (STAG)

Idea:

- pair two TAGs, one for syntax and one for L(ogical) F(orm) (= typed predicate logic),
- and do derivations in parallel.

Synchronous TAG (STAG)

Idea:

- pair two TAGs, one for syntax and one for L(ogical) F(orm) (= typed predicate logic),
- and do derivations in parallel.

STAG = two TAGs G_1, G_2 whose trees are related to each other.

Synchronous TAG (STAG)

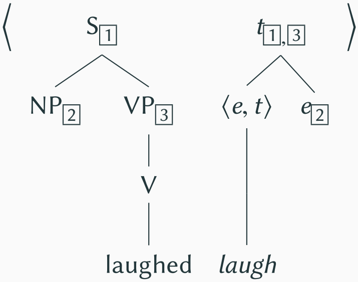
Idea:

- pair two TAGs, one for syntax and one for L(ogical) F(orm) (= typed predicate logic),
- and do derivations in parallel.

STAG = two TAGs G_1, G_2 whose trees are related to each other.

More precisely, it contains pairs $\langle \gamma_1, \gamma_2, \mathit{link} \rangle$ where γ_1 is an elementary tree from G_1, γ_2 an elementary tree from G_2, and *link* is a set of pairs of node addresses from γ_1 and γ_2 respectively.

LTAG semantics: STAG



(The links are shown with boxed numbers.)

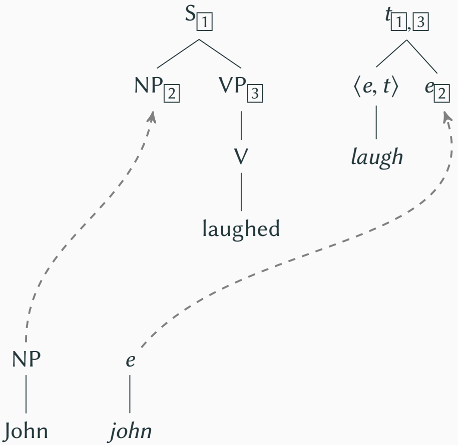
- The non-terminals of the semantic TAG are types t, e, $\langle e, t\rangle$, ...
- The semantic TAG describes the syntactic structure of typed predicate logical formulas.
- The links in this example tell us, for instance, that the subject NP corresponds to the e argument of *laugh*.

LTAG semantics: STAG

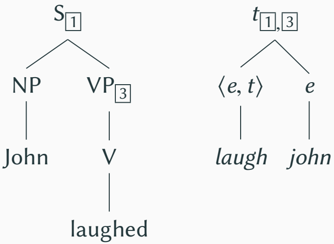
STAG derivation proceeds as in TAG, except that all operations must be paired. In every derivation step:

- A new elementary tree pair $\langle \gamma_1, \gamma_2 \rangle$ is picked.
- γ_1 is attached (substituted or adjoined) to the syntactic tree while γ_2 is attached to the semantic tree.
- The nodes that the two trees attach to must be linked.
- The link that is used in this derivation step disappears while all other links involving the attachment sites are inherited by the root of the attaching tree.

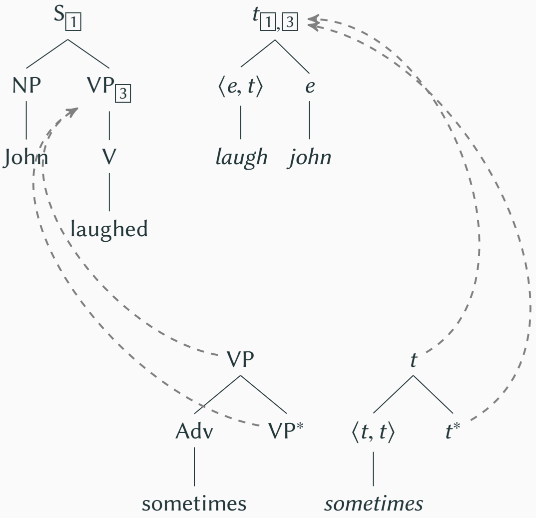
LTAG semantics: STAG



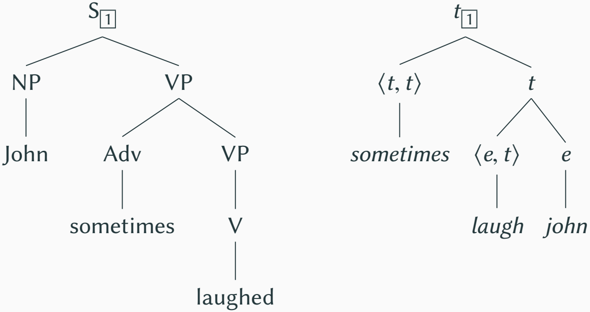
LTAG semantics: STAG



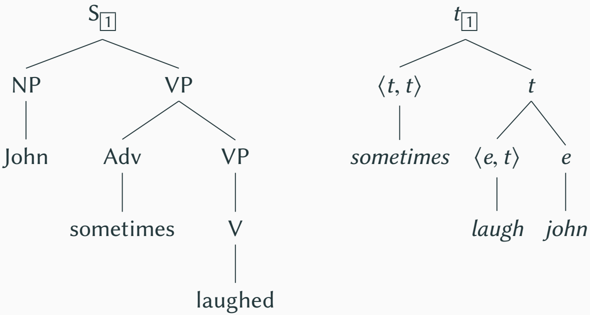
LTAG semantics: STAG



LTAG semantics: STAG



LTAG semantics: STAG



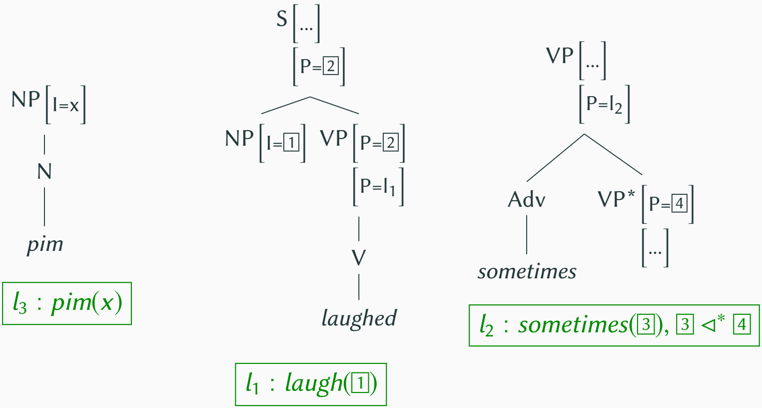
Logical form: *sometimes*(*laugh*(*john*))

Unification-based LTAG semantics with predicate logic

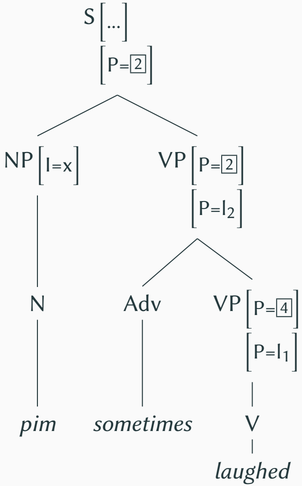
- syntax-semantics interface for LTAG
- Idea: each elementary tree is paired with
  - a set of **typed predicate logic expressions** and
  - a set of **scope constraints** (i.e., constraints on sub-term relations)
  - **interface features** that characterizes
    a) which arguments need to be filled,
    b) which elements are available as arguments for other elementary trees and
    c) the scope behaviour.

  The features are linked to positions in the elementary tree.

Unification-based LTAG semantics with predicate logic

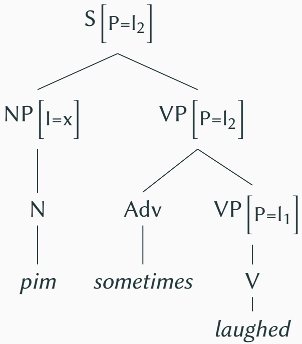


Unification-based LTAG semantics with predicate logic



l_1 : $laugh(x)$,
l_3 : $pim(x)$,
l_2 : $sometimes(\boxed{3})$,
$\boxed{3} \triangleleft^* \boxed{4}$

Unification-based LTAG semantics with predicate logic



l_1 : *laugh*(x),
l_3 : *pim*(x),
l_2 : *sometimes*($\boxed{3}$),
$\boxed{3} \triangleleft^* l_1$

- $\boxed{3} \triangleleft^* l_1$ signifies that the formula labeled l_1 is a subformula of the formula that has to be placed in the hole $\boxed{3}$
- disambiguation leads to *pim*(x) $\wedge$ *sometimes*(*laugh*(x))

Unification-based LTAG semantics with frames

- Semantic representations are linked to entire elementary trees (as in the previous approaches).
- Semantic representations: frames, expressed as typed feature structures.
- Interface features relate nodes in the syntactic tree to nodes in the frame graph.
- Frame composition by unification, triggered by the unifications on the interface features that are in turn triggered by substitution, adjunction and final top-bottom unification on the derived tree.

Unification-based LTAG semantics with frames

(4) Pim ate an apple.