Semantic Modeling with Frames

Rainer Osswald & Wiebke Petersen

Department of Linguistics and Information Science
Heinrich-Heine-Universität Düsseldorf

ESSLLI 2018

Introductory Course

Sofia University

06. 08. – 10. 08. 2018

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

SFB 991

Part 3

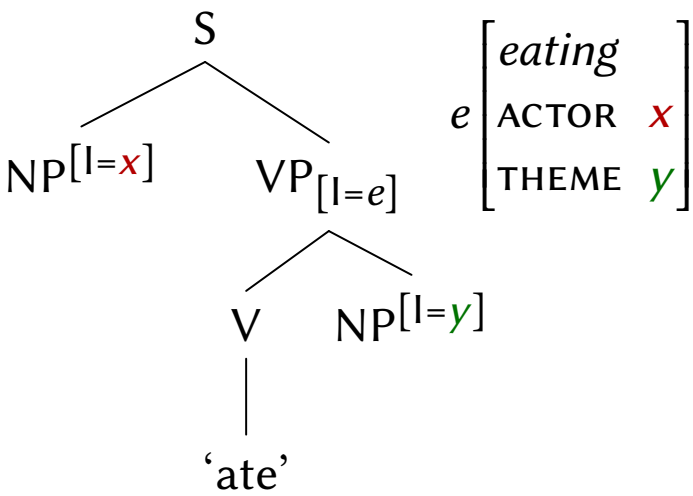
A model of the syntax-semantics interface

Topics

- Overview of the approach
- Lexicalized Tree Adjoining Grammars (LTAG)
- Feature structures based TAG (FTAG)
- Tree families and factorization in the metagrammar
- Elementary construction = elementary tree + semantic frame
- Applications (directed motion constructions, …)
- Outlook: Factorization of constructions

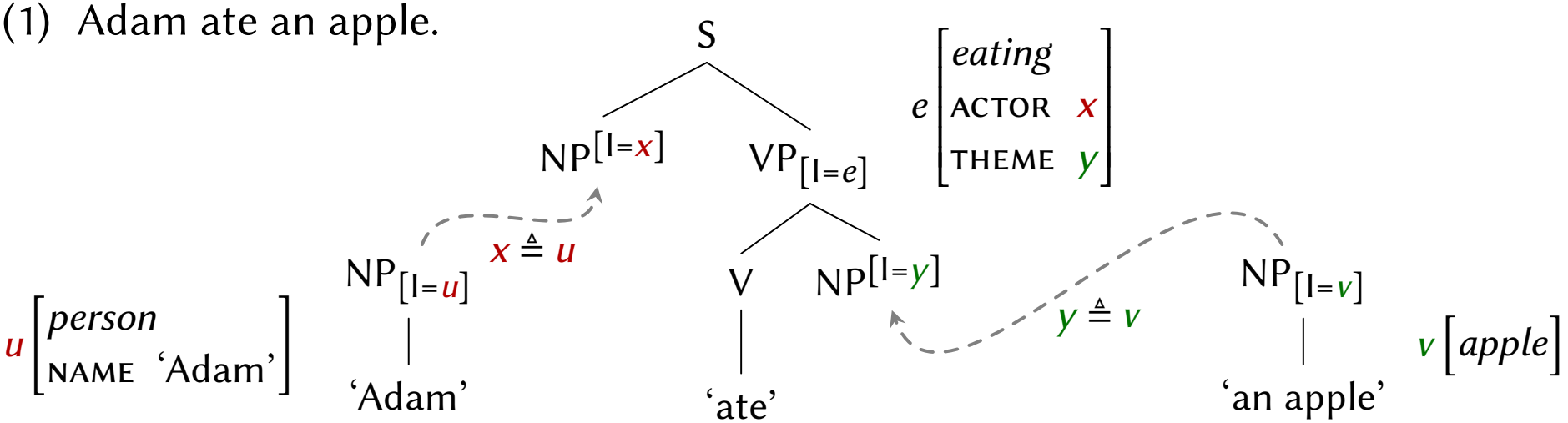
Example

(1) Adam ate an apple.



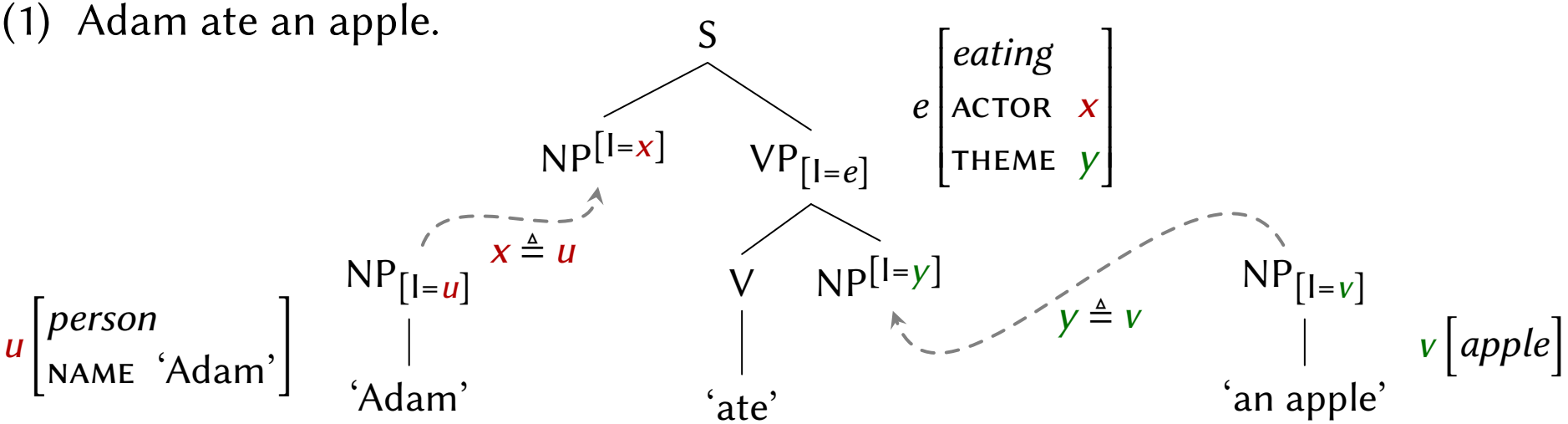
Example

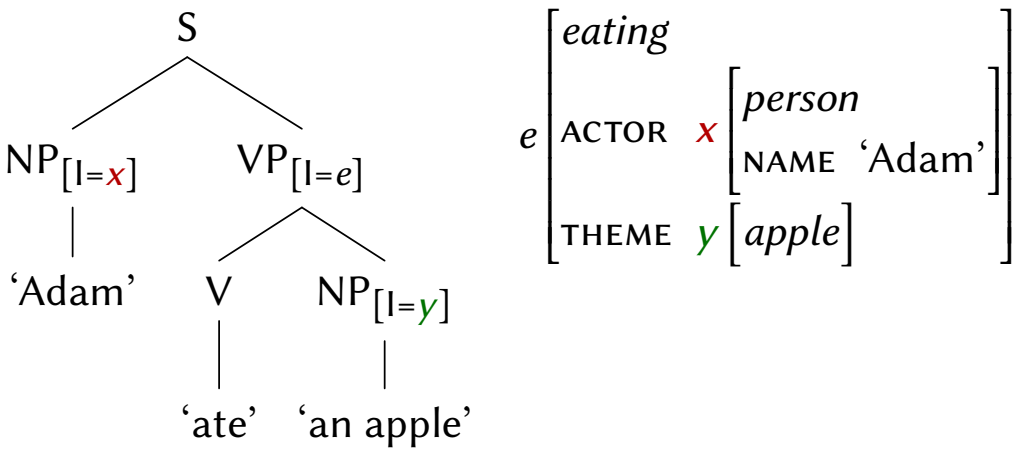
(1) Adam ate an apple.

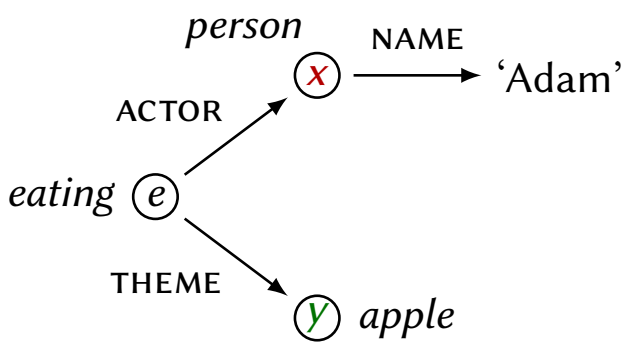


Example

(1) Adam ate an apple.







Overview of the approach

- Semantic composition (≈ unification) is triggered by syntactic composition (≈ substitution and adjunction).
- Semantic representations are linked to entire elementary trees.
  (A further decomposition is possible in the "metagrammar".)
- Interface features relate nodes in the syntactic tree to components in the semantic representation.

Introduction

Overview of the approach

- Semantic composition (≈ unification) is triggered by syntactic composition (≈ substitution and adjunction).
- Semantic representations are linked to entire elementary trees.
  (A further decomposition is possible in the "metagrammar".)
- Interface features relate nodes in the syntactic tree to components in the semantic representation.

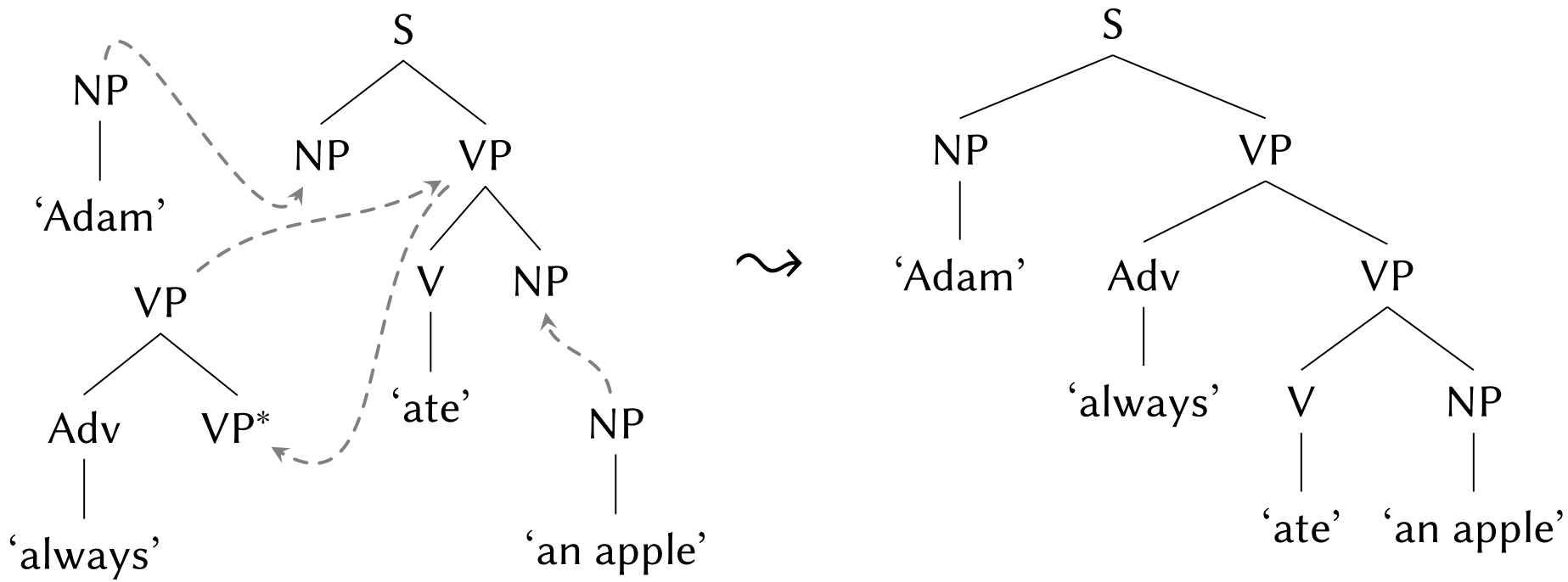
Main components of the framework [Kallmeyer/Osswald 2013]

- Frame Semantics
- Lexicalized Tree Adjoining Grammars (LTAG)
  [Joshi/Schabes 1997; Abeille/Rambow 2000]
- Metagrammatical specification and decomposition
  [Crabbé/Duchier 2005; Crabbé et al. 2013, Lichte/Petitjean 2015]

Lexicalized Tree Adjoining Grammars (LTAG)

Tree-rewriting system; mildly context sensitive grammar formalism

- Finite set of (**lexicalized**) **elementary trees**.
- Two operations: **substitution** (replacing a leaf with a new tree) and **adjunction** (replacing an internal node with a new tree).



Lexicalized Tree Adjoining Grammars (LTAG)

Feature-structure based TAG (FTAG) [Vijay-Shanker/Joshi 1988]

Each node has a top and a bottom feature structure:

- The top feature structure provide information about what the node presents within the surrounding structure.
- The bottom feature structure provide information about what the tree below the node represents.

In the final derived tree, top and bottom must be unified.

Operations on feature structures under substitution:

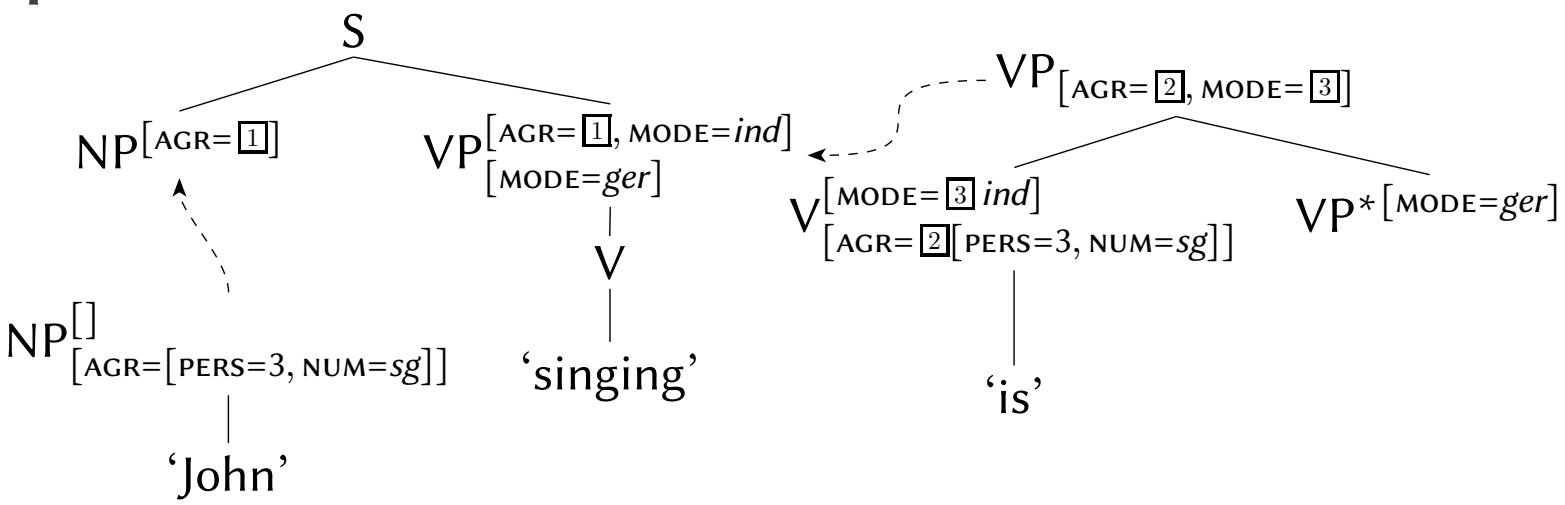
- The top of the root of the new initial tree unifies with the top of the substitution node.

Operations on feature structures under adjunction:

- The top of the root of the new auxiliary tree unifies with the top of the adjunction site; the bottom of the foot of the new tree unifies with the bottom of the adjunction site.

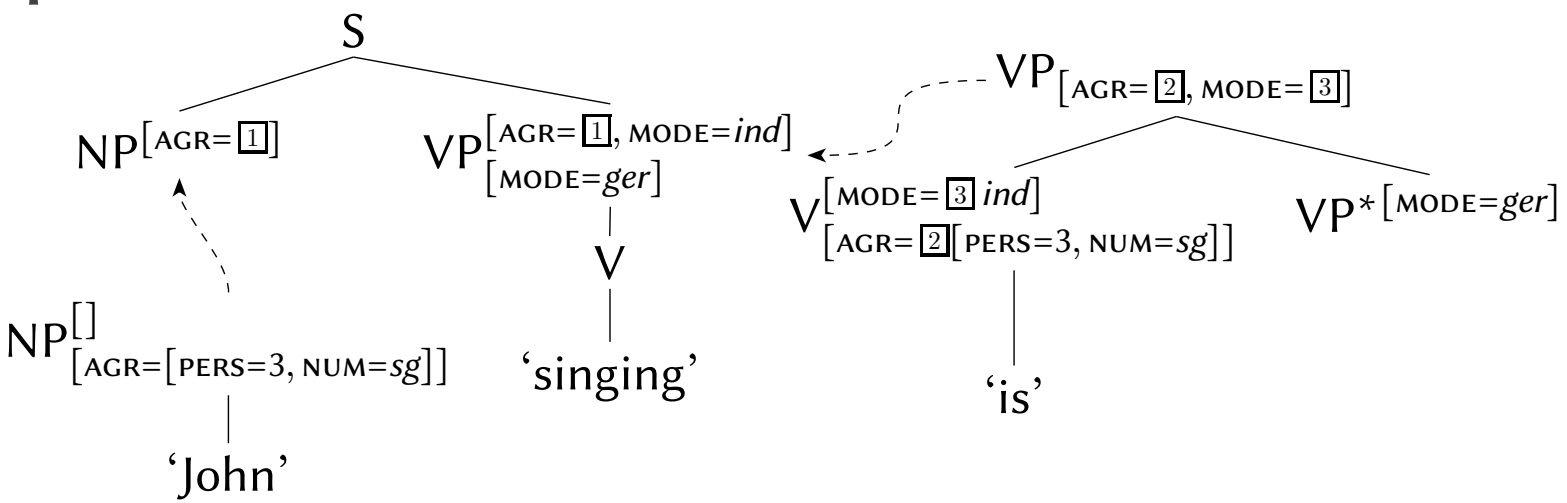
Lexicalized Tree Adjoining Grammars (LTAG)

Example

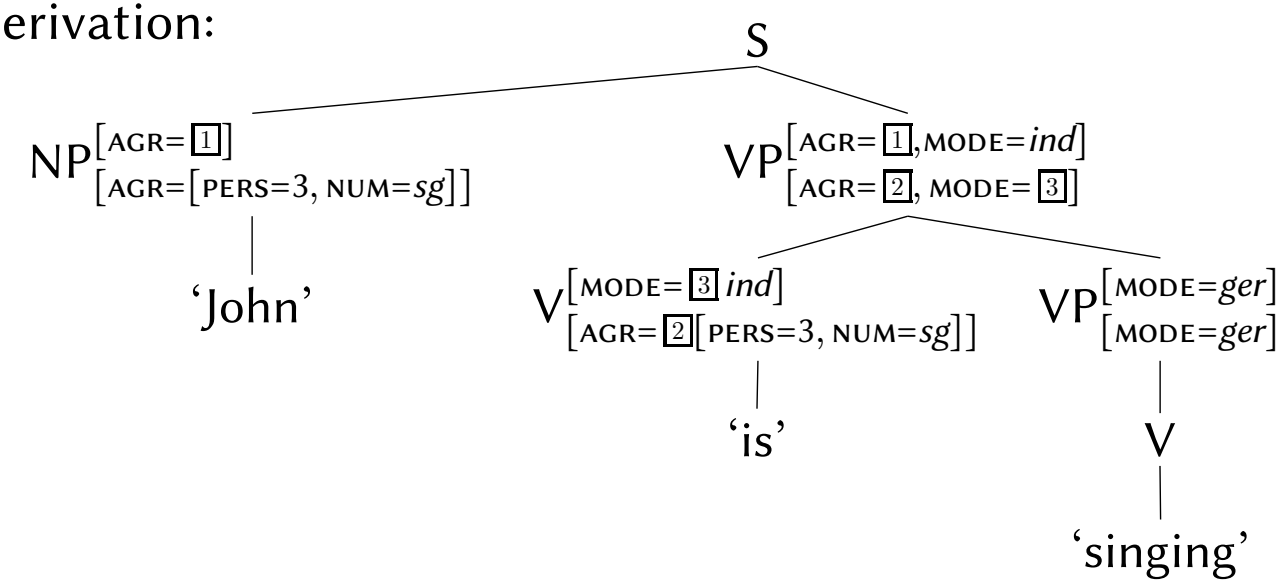


Lexicalized Tree Adjoining Grammars (LTAG)

Example

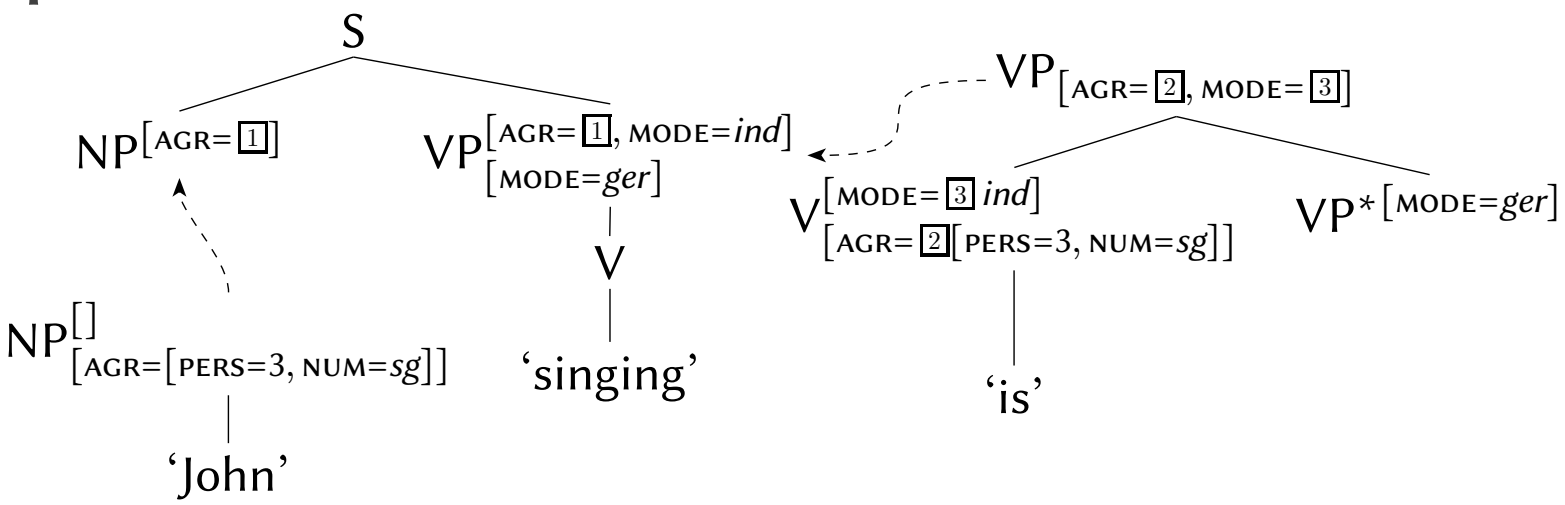


Result of derivation:

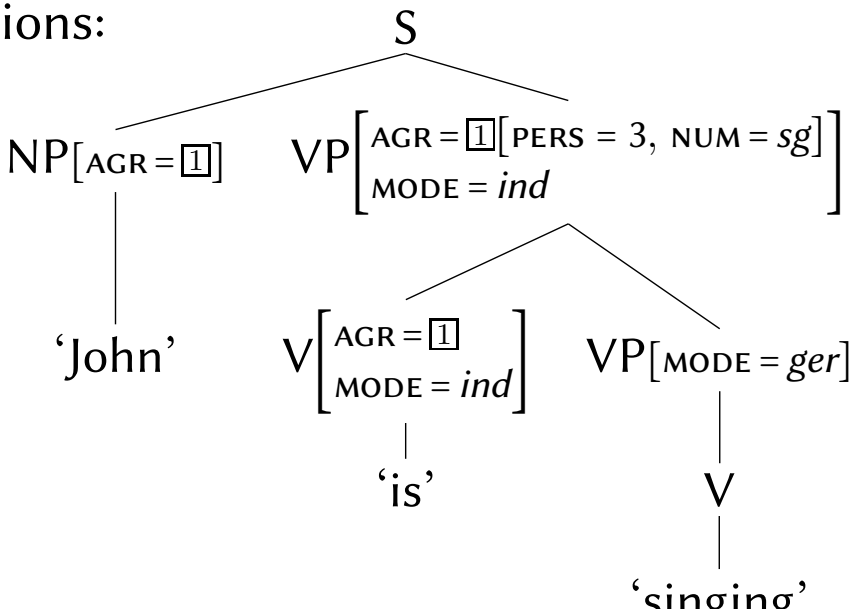


Lexicalized Tree Adjoining Grammars (LTAG)

Example



After top-bottom unifications:



Lexicalized Tree Adjoining Grammars (LTAG)

Two key properties of the LTAG formalism

- **Extended domain of locality**

  The full argument projection of a lexical item can be represented by a single elementary tree.

  Elementary trees can have a complex constituent structure.

- **Factoring recursion from the domain of dependencies**

  Constructions related to iteration and recursion are modeled by adjunction.

  Through adjunction, the local dependencies encoded by elementary trees can become long-distance dependencies in the derived trees.

Lexicalized Tree Adjoining Grammars (LTAG)

Two key properties of the LTAG formalism

- **Extended domain of locality**

  The full argument projection of a lexical item can be represented by a single elementary tree.

  Elementary trees can have a complex constituent structure.

- **Factoring recursion from the domain of dependencies**

  Constructions related to iteration and recursion are modeled by adjunction.

  Through adjunction, the local dependencies encoded by elementary trees can become long-distance dependencies in the derived trees.

Slogan: "**Complicate locally**, **simplify globally**" [Bangalore/Joshi 2010]

Lexicalized Tree Adjoining Grammars (LTAG)

"**Simplify globally**"

- The composition of elementary trees can be expressed by two general operations: substitution and adjunction.

  (Since basically all linguistic constraints are specified over the local domains represented by elementary trees.)

"**Complicate locally**"

- Elementary trees can have complex semantic representations which are not necessarily derived compositionally (in the syntax) from smaller parts of the trees.

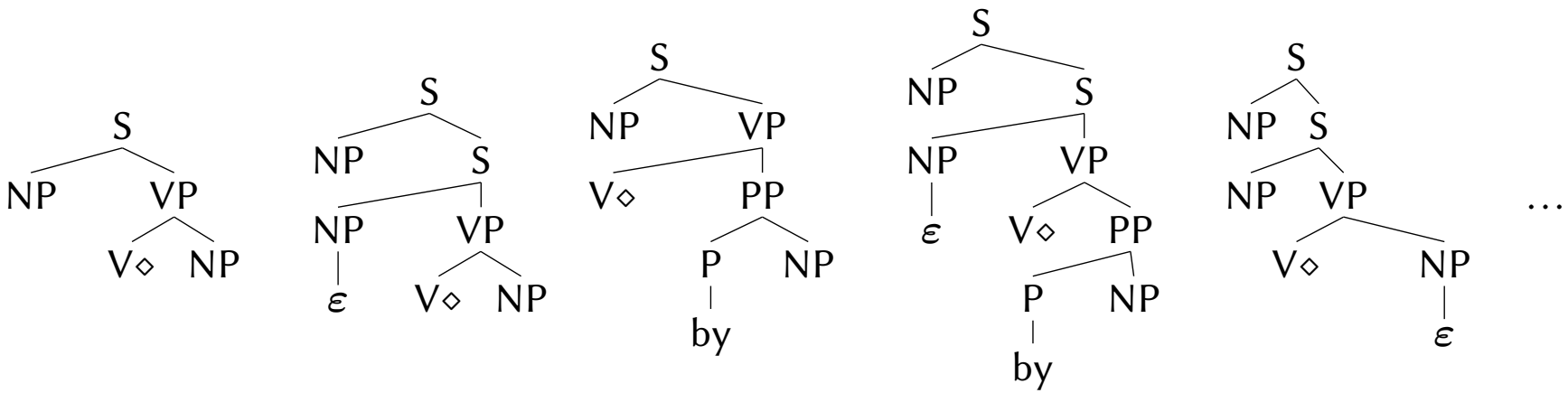
  In particular, there is no need to reproduce the internal structure of an elementary syntactic tree within its associated semantic representation. [Kallmeyer/Joshi 2003]

Lexicalized Tree Adjoining Grammars (LTAG)

Tree families

Unanchored elementary trees are organized in tree families, which capture variations in the (syntactic) subcategorization frames.

Example unanchored family for transitive verbs

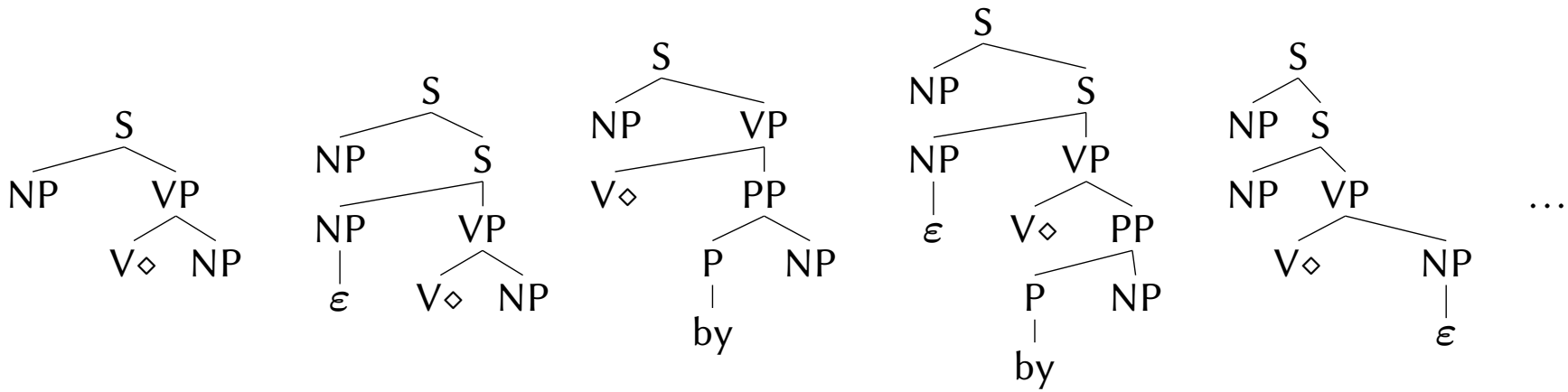


Lexicalized Tree Adjoining Grammars (LTAG)

Tree families

Unanchored elementary trees are organized in tree families, which capture variations in the (syntactic) subcategorization frames.

Example unanchored family for transitive verbs



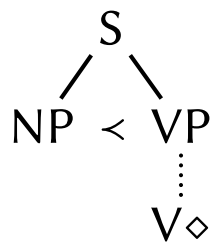
Metagrammar

Modular characterization of elementary trees by a system of tree descriptions.

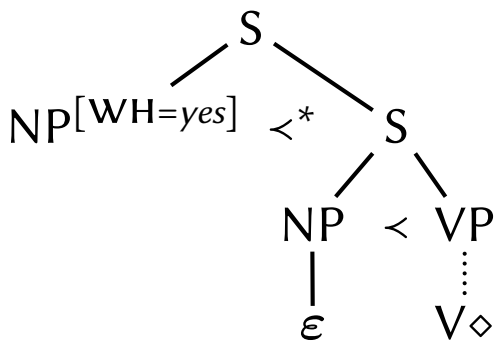
LTAG & metagrammar specification

Decomposition/factorization in the metagrammar

Class *CanSubj*



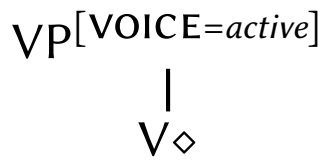
Class *ExtrSubj*



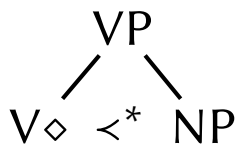
Class *Subj*

CanSubj ∨ *ExtSubj*

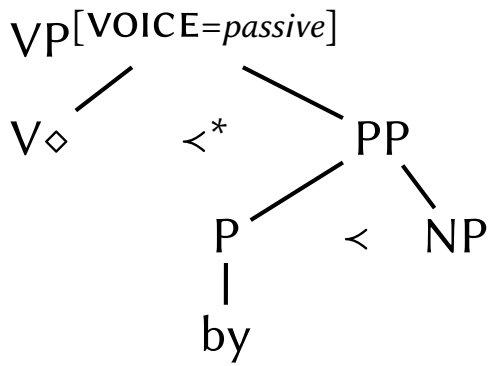
Class *ActV*



Class *DirObj*



Class *ByObj*



Class *PassV*

Class *Transitive*

$((Subj \wedge ActV) \vee ByObj \vee PassV) \wedge ((DirObj \wedge ActV) \vee (Subj \wedge PassV))$

LTAG & metagrammar specification

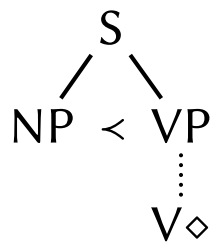
XMG (eXtensible MetaGrammar) [e.g., Crabbé et al. 2013]

- A framework for specifying (the elementary structures of) tree based grammars by means of a **declarative** language (e.g., by dominance and precedence constraints)
- The specifications are organized into **classes** that can be **reused** ("imported") by other classes.
- Classes may contain descriptions from different **dimensions**, and the XMG system can be extended in this respect, e.g., by a dimension of **frame** descriptions.
- An XMG **compiler** generates the elementary structures of a grammar from a metagrammar.

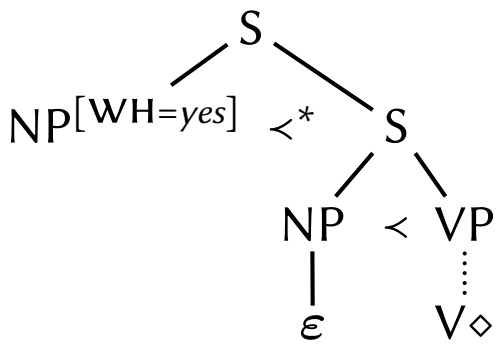
LTAG & metagrammar specification

Example

Class *CanSubj*



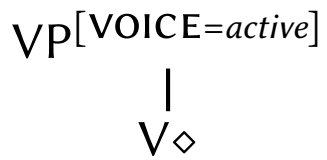
Class *ExtrSubj*



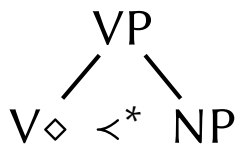
Class *Subj*

$CanSubj \vee ExtSubj$

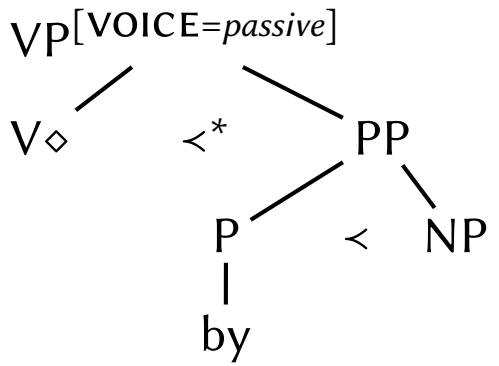
Class *ActV*



Class *DirObj*



Class *ByObj*

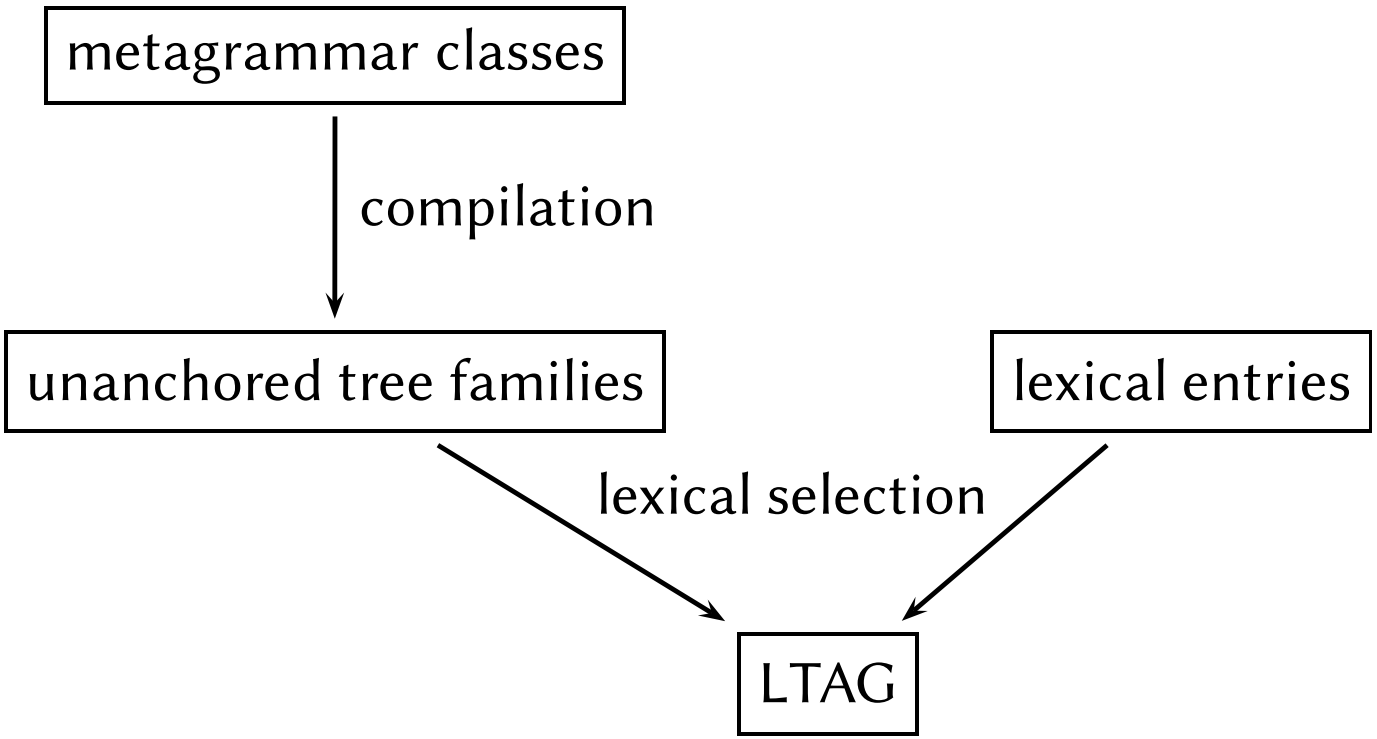


Class *PassV*

Class *Transitive*

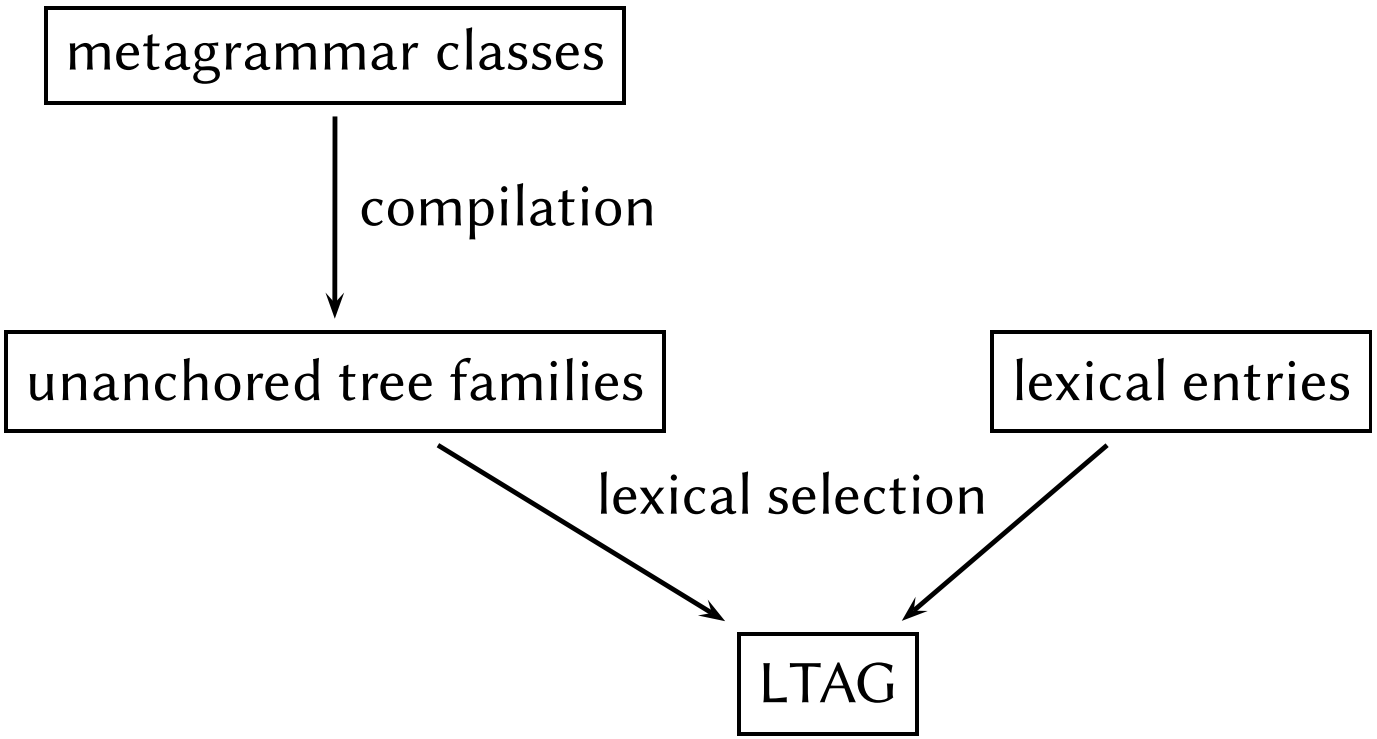
$((Subj \wedge ActV) \vee ByObj \vee PassV) \wedge ((DirObj \wedge ActV) \vee (Subj \wedge PassV))$

Summary of the LTAG architecture



LTAG & metagrammar specification

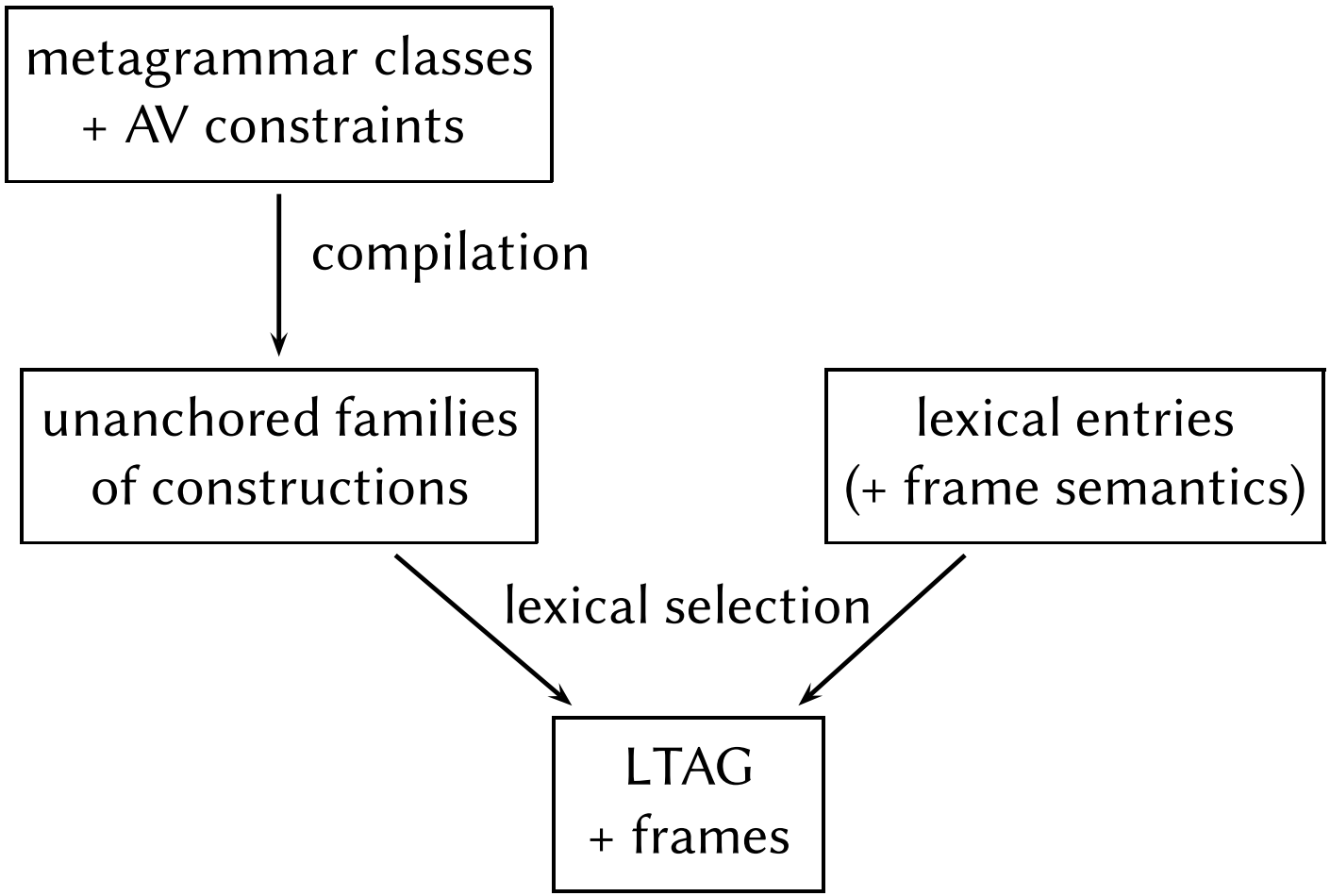
Summary of the LTAG architecture



Next step: Add (frame) semantics to all components and link syntax to semantics.

LTAG & frame semantics

Overall architecture (syntax + semantics)



LTAG & frame semantics

Elements of the syntax-semantics interface

- **Elementary construction:**

  elementary tree
  + semantic frame
  + linking of frame node variables to interface features in the tree

- Specification in the metagrammar:

  classes of tree constraints
  + sets of attribute-value constraints
  + linking of variables to interface features

  Note: Regularities about **argument linking** are expressed in the metagrammar. [Kallmeyer/Lichte/Osswald/Petitjean 2016]

- Semantic **composition** $\approx$ frame unification via identification of interface variables during substitution and adjunction.

Applications: directed motion construction

Intransitive:

(2) a. Mary walked to the house.
b. The ball rolled into the goal.

Applications: directed motion construction

Intransitive:

(2) a. Mary walked to the house.
b. The ball rolled into the goal.

Transitive:

(3) a. John threw/kicked the ball into the goal.
b. John pushed/pulled the cart to the station.
c. John rolled the ball into the hole.

Applications: directed motion construction

Intransitive:

(2) a. Mary walked to the house.
b. The ball rolled into the goal.

Transitive:

(3) a. John threw/kicked the ball into the goal.
b. John pushed/pulled the cart to the station.
c. John rolled the ball into the hole.

Directional specifications are not restricted to **goal** expressions but can also describe the **source** or the **course of the path** in more detail.

Applications: directed motion construction

Intransitive:

(2) a. Mary walked to the house.
b. The ball rolled into the goal.

Transitive:

(3) a. John threw/kicked the ball into the goal.
b. John pushed/pulled the cart to the station.
c. John rolled the ball into the hole.

Directional specifications are not restricted to **goal** expressions but can also describe the **source** or the **course of the path** in more detail. Moreover, path descriptions can be **iterated** to some extent:

(4) a. John walked through the gate along the fence to the house.
b. John threw the ball over the fence into the yard.

Applications: directed motion construction

Question: Syntactic treatment of directional PPs ?

- Construction ($\rightsquigarrow$ elementary tree)
- Syntactic composition ($\rightsquigarrow$ adjunction)

Applications: directed motion construction

Question: Syntactic treatment of directional PPs ?

- Construction ($\rightsquigarrow$ elementary tree)
- Syntactic composition ($\rightsquigarrow$ adjunction)

Arguments for treating goal (or **bounded**) PPs constructionally, in contrast to path (or **unbounded**) PPs:

- Goal PPs cannot be iterated.

Applications: directed motion construction

Question: Syntactic treatment of directional PPs?

- Construction ($\rightsquigarrow$ elementary tree)
- Syntactic composition ($\rightsquigarrow$ adjunction)

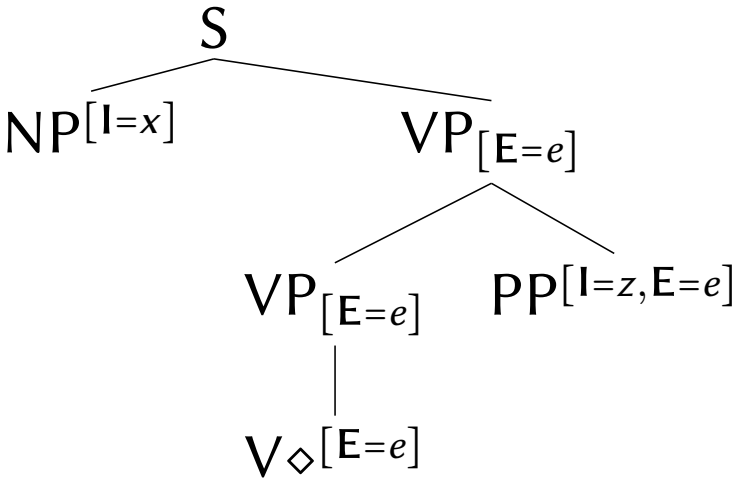
Arguments for treating goal (or **bounded**) PPs constructionally, in contrast to path (or **unbounded**) PPs:

- Goal PPs cannot be iterated.
- They affect the Aktionsart of the expression:

(5) a. She walked (*in half an hour/for half an hour).
b. She walked to the brook (in half an hour/*for half an hour).
c. She walked along the brook (*in half an hour/for half an hour).

Applications: directed motion construction

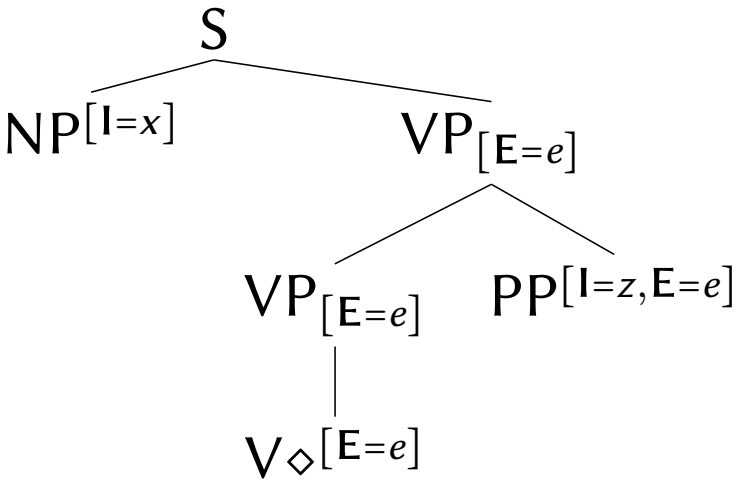
Unanchored construction for intransitive directed motion (*n0Vpp(dir)*):



$$e\begin{bmatrix} \textit{bounded-translocation} \\ \textsc{mover} & x \\ \textsc{goal} & z \\ \textsc{path} & [\textit{path}] \end{bmatrix}$$

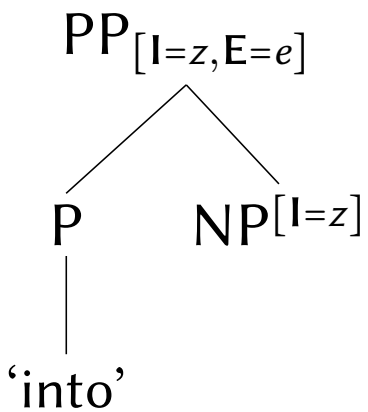
Applications: directed motion construction

Unanchored construction for intransitive directed motion (*n0Vpp(dir)*):



$$e\begin{bmatrix} \textit{bounded-translocation} \\ \textsc{mover} & x \\ \textsc{goal} & z \\ \textsc{path} & [\textit{path}] \end{bmatrix}$$

Elementary tree for ‘into’:



$$e\begin{bmatrix} \textit{bounded-translocation} \\ \textsc{path} \begin{bmatrix} \textit{path} \\ \textsc{endp} \ \boxed{1} \end{bmatrix} \end{bmatrix}$$

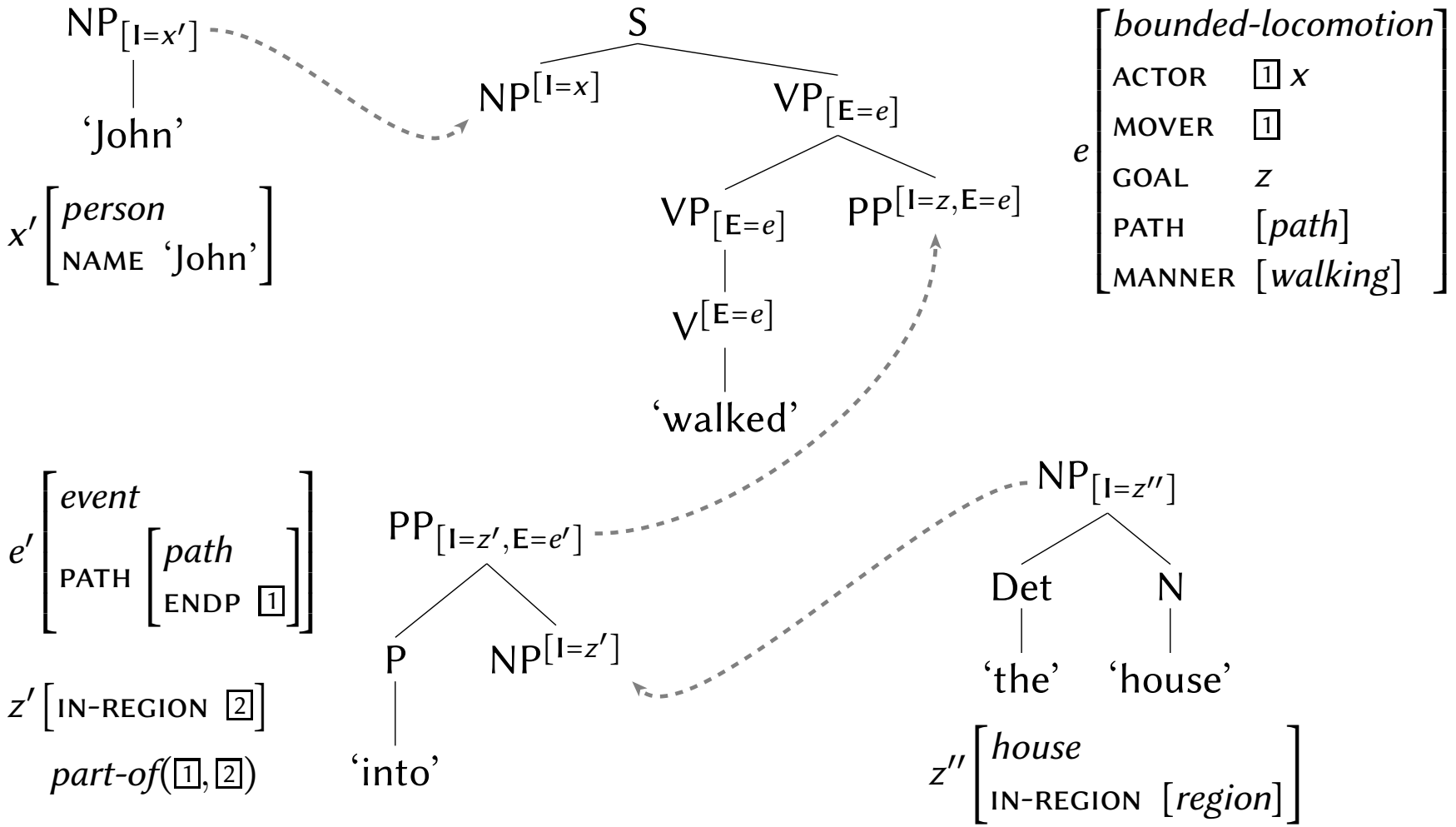
$$z\left[\textsc{in-region} \ \boxed{2}\right]$$

$$\textit{part-of}(\boxed{1}, \boxed{2})$$

Applications: directed motion construction

Example (intransitive directed motion)

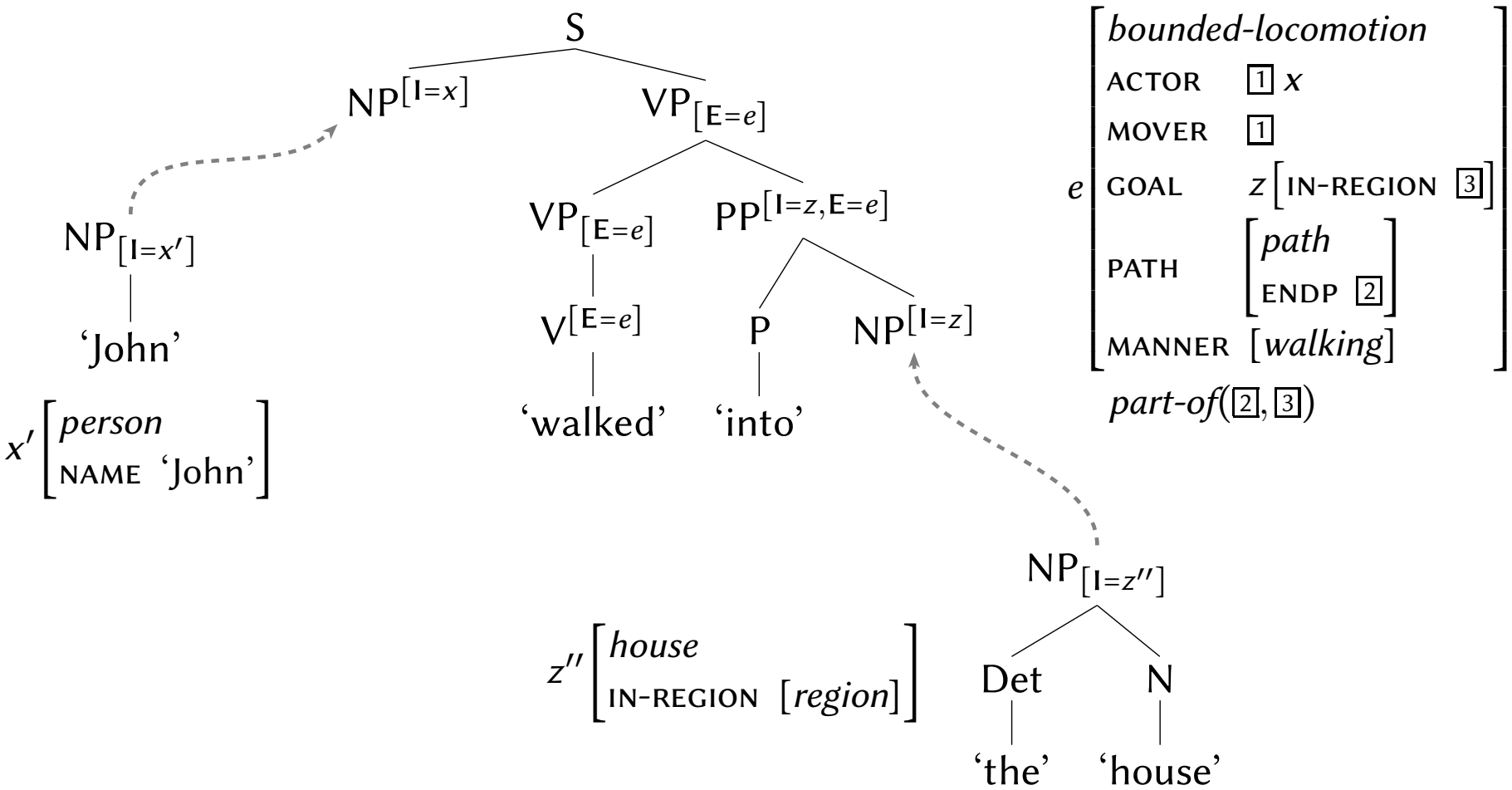
(6) John walked into the house.



Applications: directed motion construction

Example (intransitive directed motion)

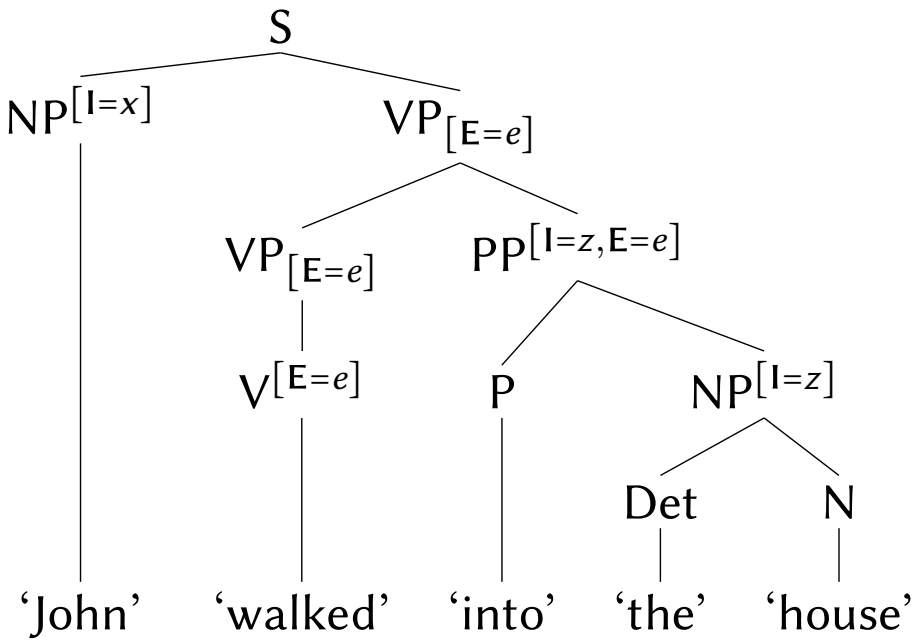
(6) John walked into the house.



Applications: directed motion construction

Example (intransitive directed motion)

(6) John walked into the house.



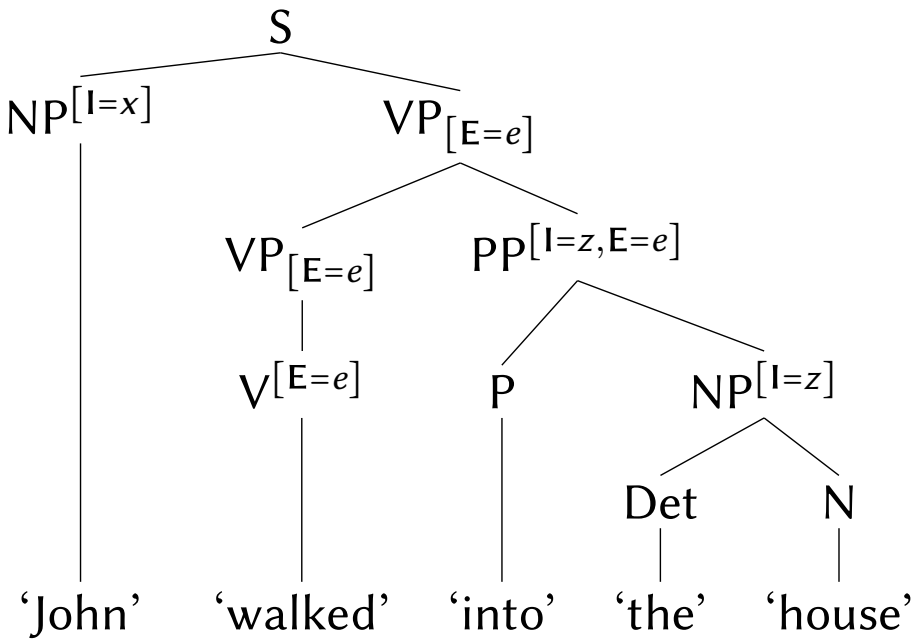
$$e\begin{bmatrix} \textit{bounded-locomotion} \\ \textsc{actor} \quad \boxed{1}\, x\begin{bmatrix} \textit{person} \\ \textsc{name} \ \text{`John'} \end{bmatrix} \\ \textsc{mover} \quad \boxed{1} \\ \textsc{goal} \quad z\begin{bmatrix} \textit{house} \\ \textsc{in-region} \ \boxed{3}\ [\textit{region}] \end{bmatrix} \\ \textsc{path} \quad \begin{bmatrix} \textit{path} \\ \textsc{endp} \ \boxed{2} \end{bmatrix} \\ \textsc{manner} \ [\textit{walking}] \end{bmatrix}$$

$$\textit{part-of}(\boxed{2}, \boxed{3})$$

Applications: directed motion construction

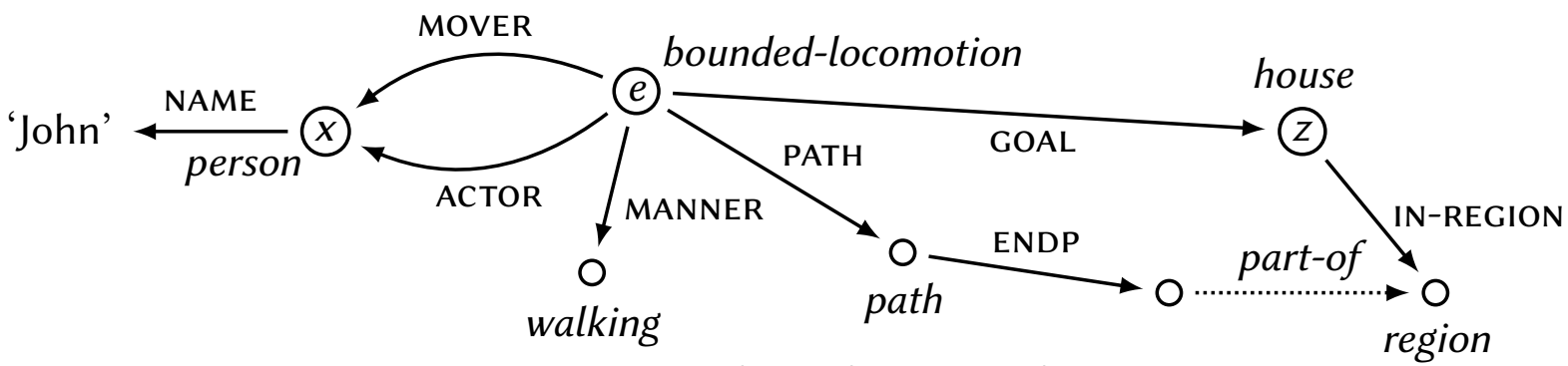
Example (intransitive directed motion)

(6) John walked into the house.



$$e\begin{bmatrix} \textit{bounded-locomotion} \\ \textsc{actor} \quad \boxed{1}\, x\begin{bmatrix} \textit{person} \\ \textsc{name} \ \text{'John'} \end{bmatrix} \\ \textsc{mover} \quad \boxed{1} \\ \textsc{goal} \quad z\begin{bmatrix} \textit{house} \\ \textsc{in-region} \ \boxed{3}\ [\textit{region}] \end{bmatrix} \\ \textsc{path} \quad \begin{bmatrix} \textit{path} \\ \textsc{endp} \ \boxed{2} \end{bmatrix} \\ \textsc{manner} \ [\textit{walking}] \end{bmatrix}$$

$$\textit{part-of}(\boxed{2}, \boxed{3})$$



Applications: directed motion construction

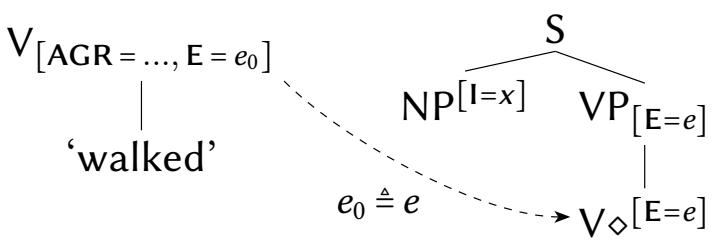
Lexical anchoring (non-directed case)

morph entry +
'walked'
pos: V
Syn_1:
$\left[\text{AGR} = \left[\begin{array}{l}\text{PERS} = 3\\ \text{NUM} = sg\end{array}\right]\right]$
lemma: walk

lemma entry +
walk:
FAM: n0V, …
Syn_2:
$\left[\text{E} = e_0\right]$
Sem:
$e_0\left[\begin{array}{l}locomotion\\ \text{MANNER}\ [walking]\end{array}\right]$

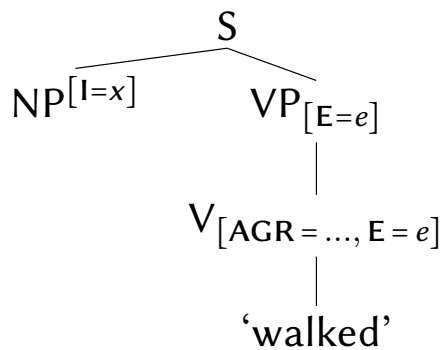
Constraints:
locomotion $\Rrightarrow$ *activity* $\wedge$ *translocation*
translocation $\Rrightarrow$ *motion* $\wedge$ PATH : *path*
activity $\Rrightarrow$ ACTOR : $\top$
motion $\Rrightarrow$ MOVER : $\top$
activity $\wedge$ *motion* $\Rrightarrow$ ACTOR $\doteq$ MOVER

$\rightsquigarrow$ $e_0\left[\begin{array}{ll}locomotion & \\ \text{ACTOR} & \boxed{1}\\ \text{MOVER} & \boxed{1}\\ \text{PATH} & [path]\\ \text{MANNER} & [walking]\end{array}\right]$



$e\left[\begin{array}{ll}activity & \\ \text{ACTOR} & x\end{array}\right]$

$\rightsquigarrow$ $e\left[\begin{array}{ll}locomotion & \\ \text{ACTOR} & \boxed{1}\ x\\ \text{MOVER} & \boxed{1}\\ \text{PATH} & [path]\\ \text{MANNER} & [walking]\end{array}\right]$



Applications: directed motion construction

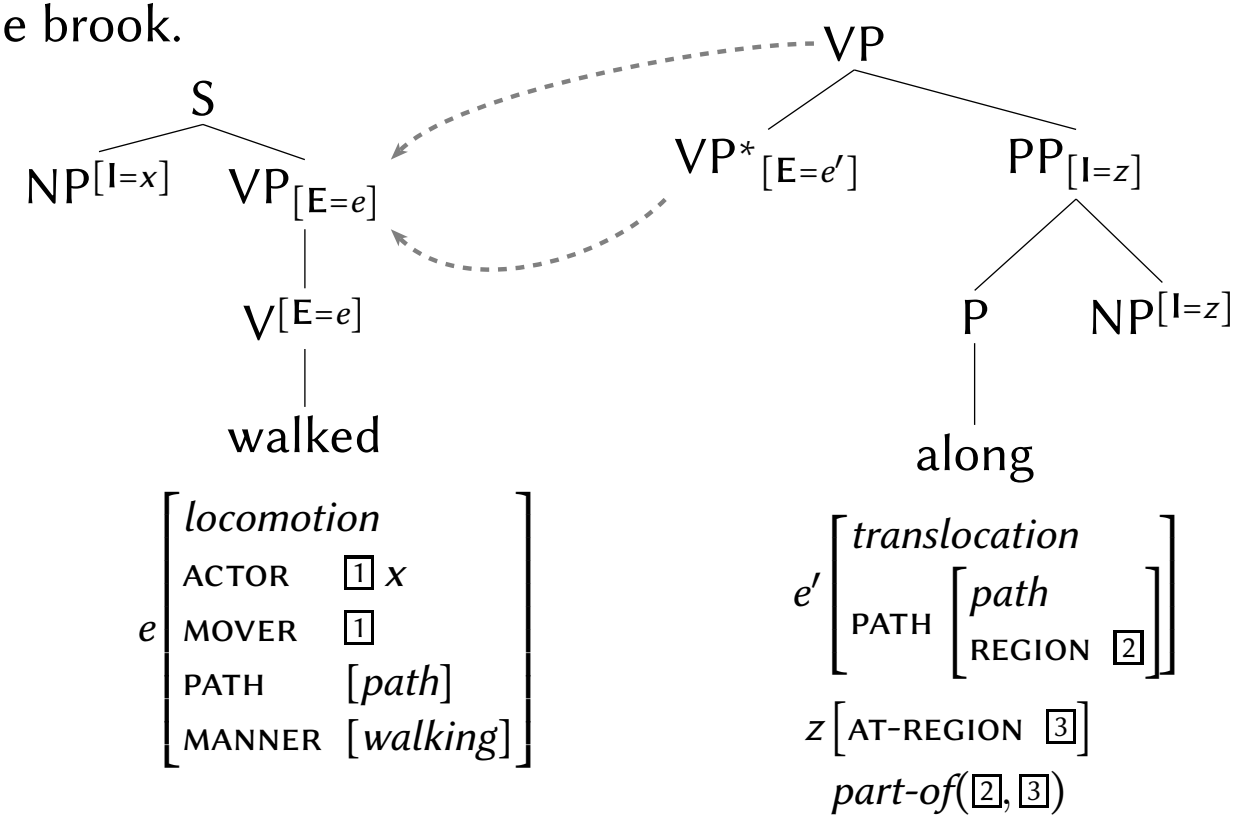
Example

(7) John walked along the brook.

Applications: directed motion construction

Example

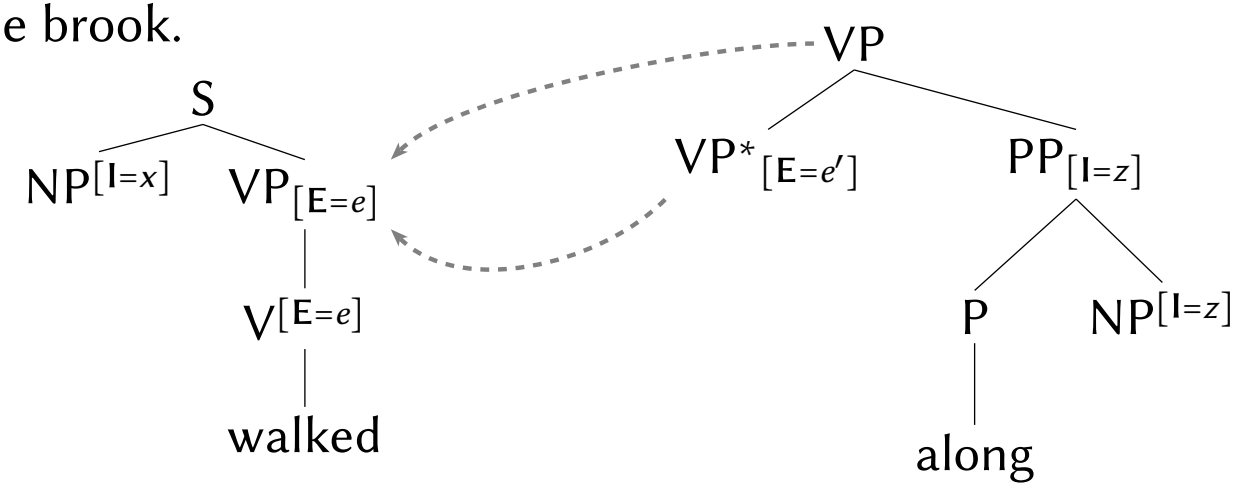
(7) John walked along the brook.



Applications: directed motion construction

Example

(7) John walked along the brook.

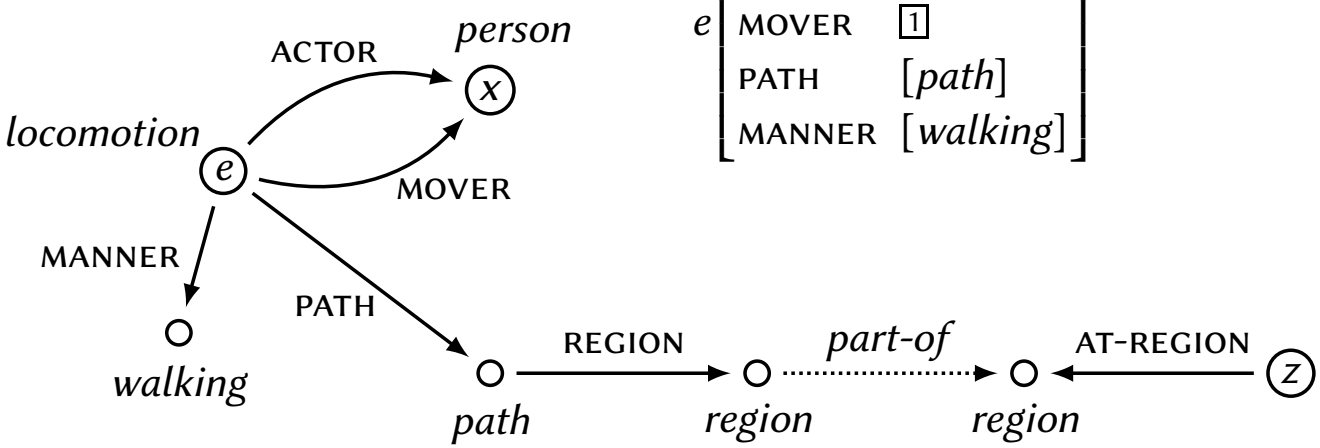


$$e\begin{bmatrix} \textit{locomotion} \\ \textsc{actor} & \boxed{1}\ x \\ \textsc{mover} & \boxed{1} \\ \textsc{path} & [\textit{path}] \\ \textsc{manner} & [\textit{walking}] \end{bmatrix}$$

$$e'\begin{bmatrix} \textit{translocation} \\ \textsc{path} \begin{bmatrix} \textit{path} \\ \textsc{region} & \boxed{2} \end{bmatrix} \end{bmatrix}$$

$$z\left[\textsc{at-region}\ \boxed{3}\right]$$

$$\textit{part-of}(\boxed{2}, \boxed{3})$$



Applications: directed motion construction

Example (causative directed motion)

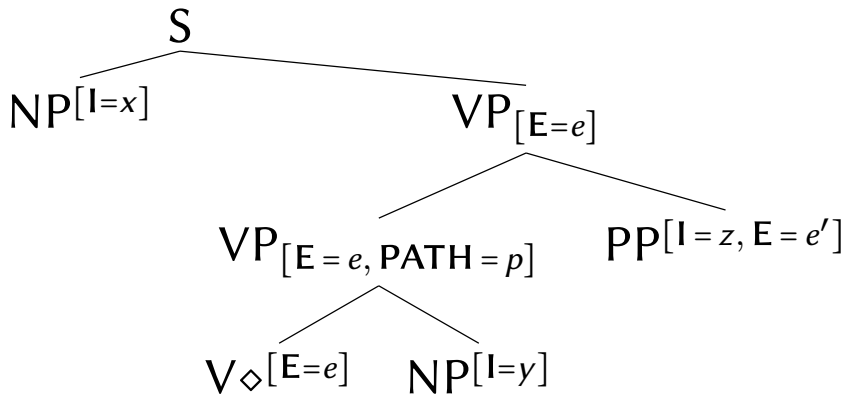
(8) Mary threw/kicked/rolled the ball into the room.

Applications: directed motion construction

Example (causative directed motion)

(8) Mary threw/kicked/rolled the ball into the room.

Unanchored construction (*n0Vn1pp(dir)*):



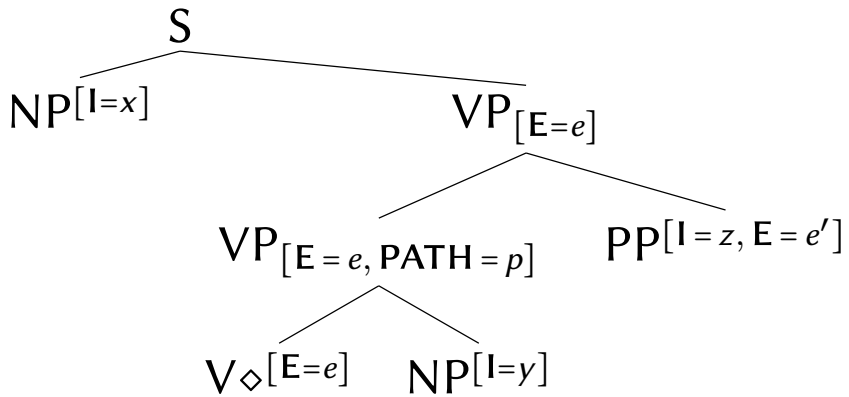
$$e\begin{bmatrix} \textit{causation} & \\ \textsc{cause} & \begin{bmatrix} \textit{activity} & \\ \textsc{actor} & x \\ \textsc{theme} & y \end{bmatrix} \\ \textsc{effect} & e'\begin{bmatrix} \textit{bounded-translocation} & \\ \textsc{mover} & y \\ \textsc{goal} & z \\ \textsc{path} & p \end{bmatrix} \end{bmatrix}$$

Applications: directed motion construction

Example (causative directed motion)

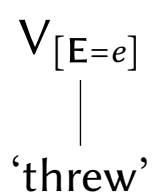
(8) Mary threw/kicked/rolled the ball into the room.

Unanchored construction (*n0Vn1pp(dir)*):



$$e\begin{bmatrix} \textit{causation} & & \\ \textsc{cause} & \begin{bmatrix} \textit{activity} \\ \textsc{actor} & x \\ \textsc{theme} & y \end{bmatrix} & \\ \textsc{effect} & e' \begin{bmatrix} \textit{bounded-translocation} \\ \textsc{mover} & y \\ \textsc{goal} & z \\ \textsc{path} & p \end{bmatrix} \end{bmatrix}$$

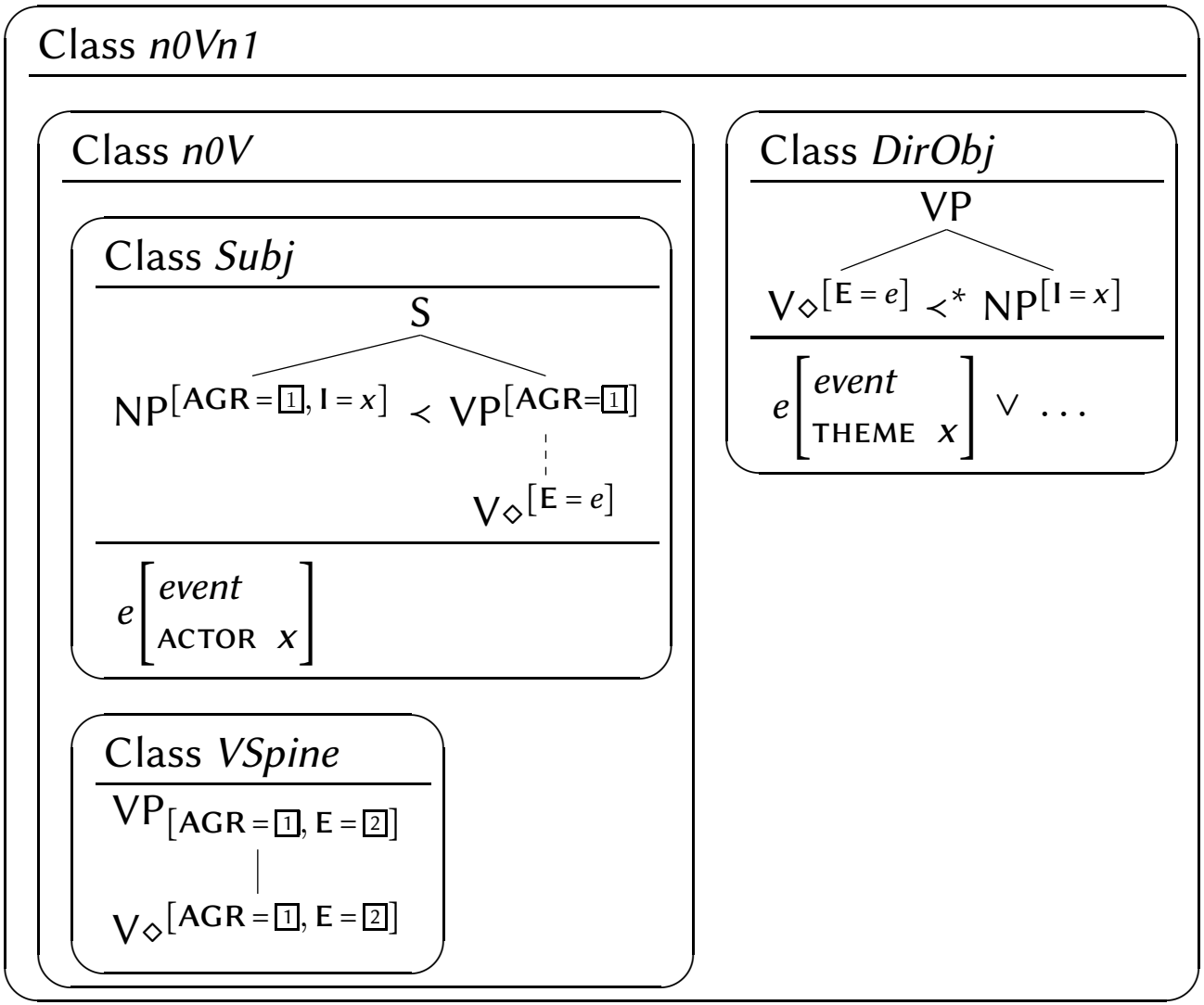
(Partial) lexical entry for ‘threw’:



$$e\begin{bmatrix} \textit{onset-causation} & & \\ \textsc{cause} & \begin{bmatrix} \textit{activity} \\ \textsc{actor} & \top \\ \textsc{theme} & \boxed{1} \\ \textsc{manner} & [\textit{throwing}] \end{bmatrix} & \\ \textsc{effect} & \begin{bmatrix} \textit{translocation} \wedge \textit{undergoing} \\ \textsc{theme} & \boxed{1} \\ \textsc{mover} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

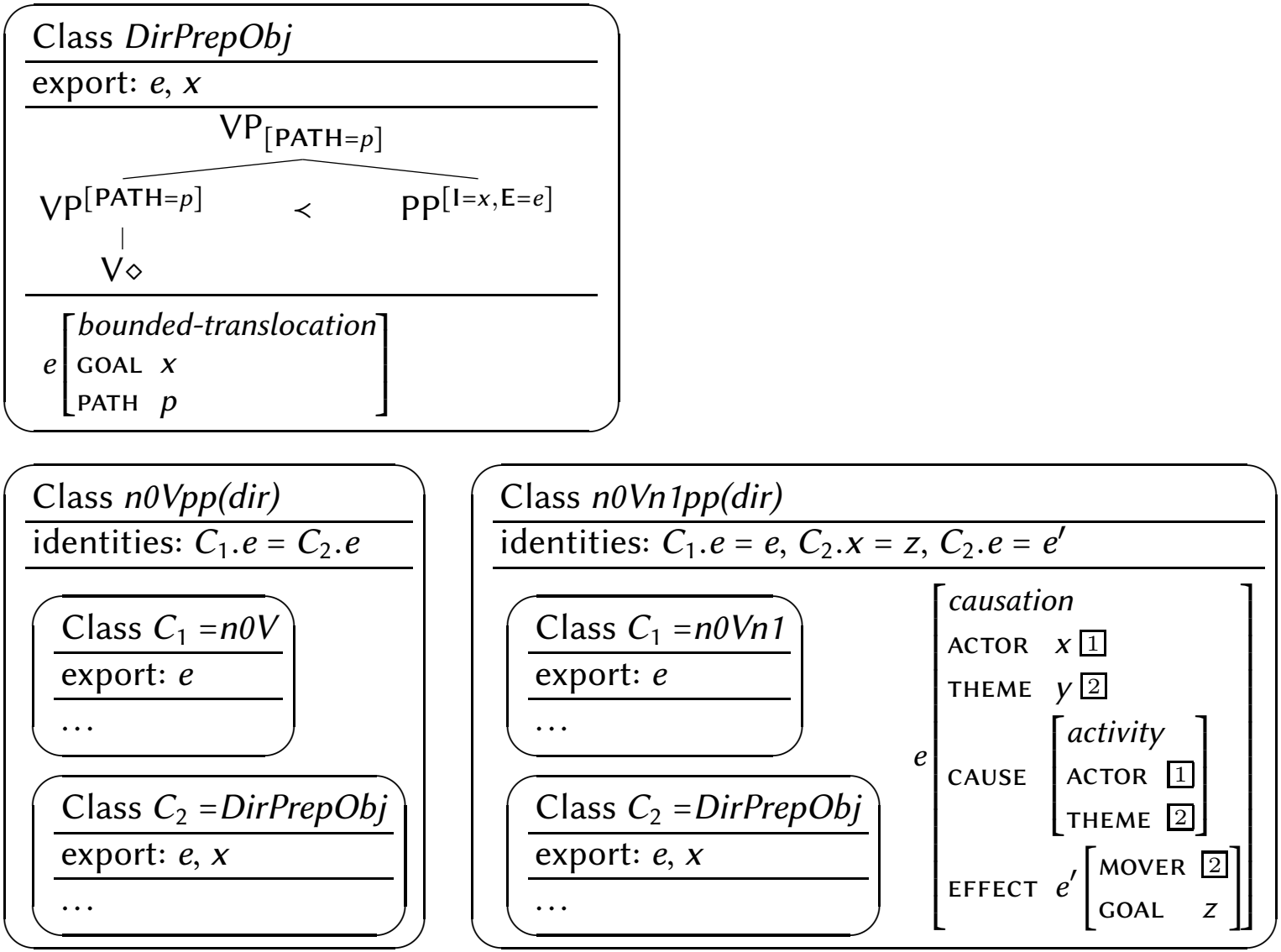
Applications: directed motion construction

Outlook: **Metagrammar classes** (syntax + semantics)



Applications: directed motion construction

Outlook: **Metagrammar classes** (syntax + semantics)



Applications: dative alternation

Sketch

[→ Kallmeyer/Osswald 2013]

(9) a. John sent Mary the book. (double object construction)
b. John sent the book to Mary. (prepositional object construction)

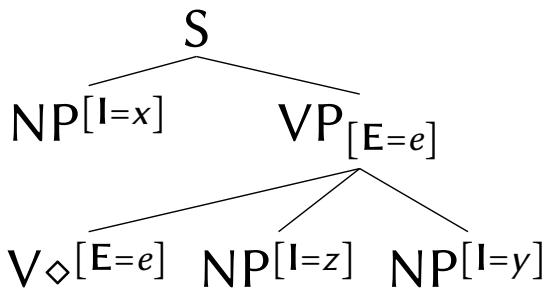
Applications: dative alternation

Sketch

[→ Kallmeyer/Osswald 2013]

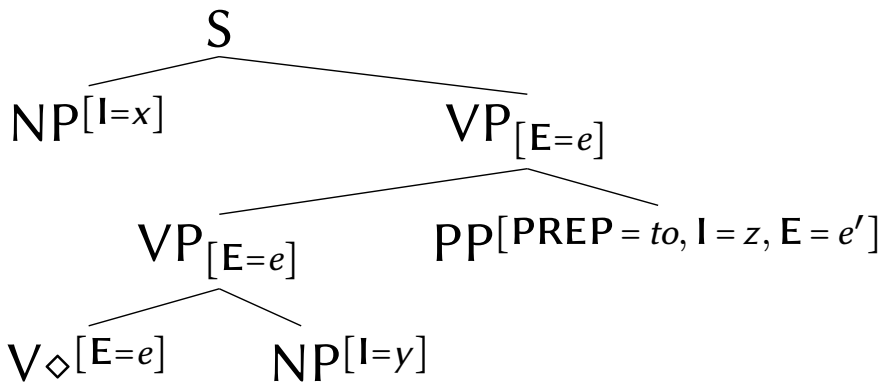
(9) a. John sent Mary the book. (double object construction)
b. John sent the book to Mary. (prepositional object construction)

a)



$$e\begin{bmatrix} \textit{causation} & \\ \textsc{cause} & \begin{bmatrix} \textit{activity} \\ \textsc{actor} \quad x \end{bmatrix} \\ \textsc{effect} & \begin{bmatrix} \textit{change-of-possession} \\ \textsc{theme} \quad y \\ \textsc{recipient} \quad z \end{bmatrix} \end{bmatrix}$$

b)



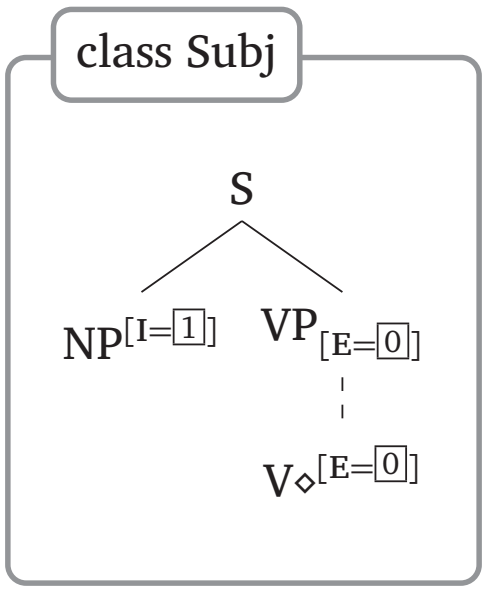
$$e\begin{bmatrix} \textit{causation} & \\ \textsc{cause} & \begin{bmatrix} \textit{activity} \\ \textsc{actor} \quad x \end{bmatrix} \\ \textsc{effect} & e' \begin{bmatrix} \textit{bounded-translocation} \\ \textsc{mover} \quad y \\ \textsc{goal} \quad z \end{bmatrix} \end{bmatrix}$$

XMG implementation of frame semantics

Examples

[from Lichte/Petitjean 2015]

<syn>-dimension of class *Subj*



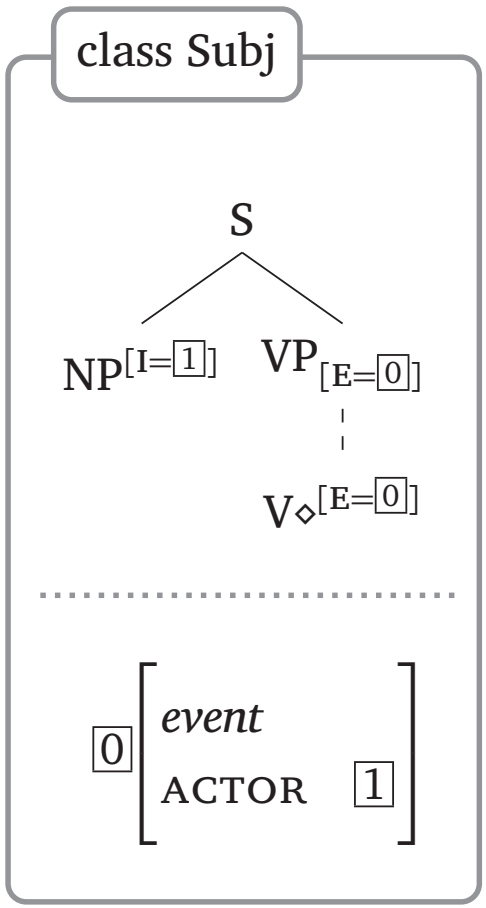

```
class Subj
...
<syn>{
  node ?S [cat=s];
  node ?SUBJ [cat=np,
              top=[i=?1]];
  node ?VP [cat=vp,bot=[e=?0]];
  node ?V (mark=anchor)
          [cat=v,top=[e=?0]];
  ?S->?SUBJ; ?S->?VP; ?VP->*?V;
  ?SUBJ>>?VP
}
```

XMG implementation of frame semantics

Examples

[from Lichte/Petitjean 2015]

<syn>-dimension + <frame>-dimension of class *Subj*



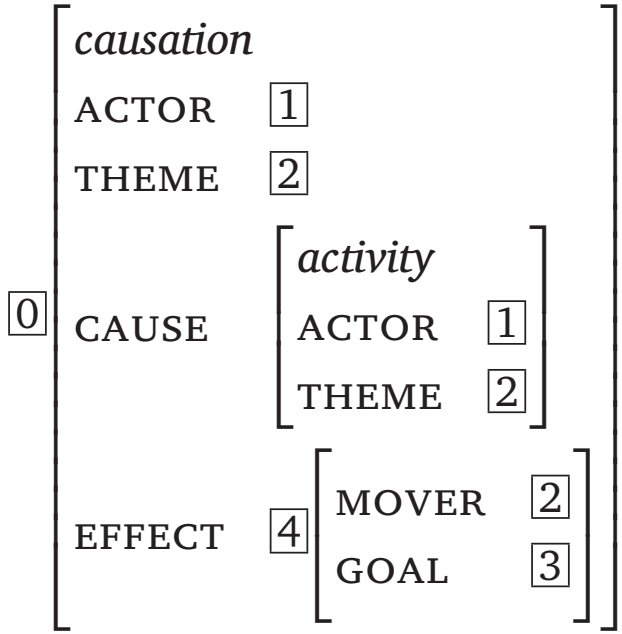

```
class Subj
...
<syn>{
  node ?S [cat=s];
  node ?SUBJ [cat=np,
              top=[i=?1]];
  node ?VP [cat=vp,bot=[e=?0]];
  node ?V (mark=anchor)
          [cat=v,top=[e=?0]];
  ?S->?SUBJ; ?S->?VP; ?VP->*?V;
  ?SUBJ>>?VP
}
<frame>{
?0[event,
    actor:?1]
}
```

XMG implementation of frame semantics

Examples

[from Lichte/Petitjean 2015]

Specification of frames:



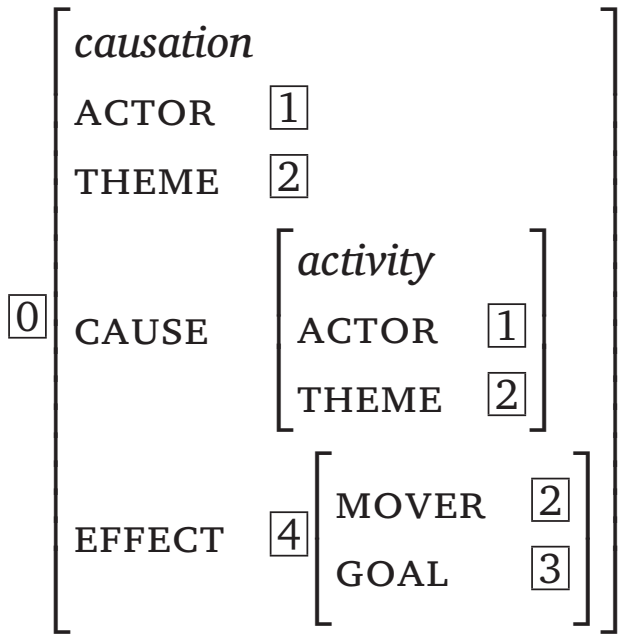

```
<frame>{
?0[causation,
    actor:?1,
    theme:?2,
    cause:[activity,
           actor:?1,
           theme:?2],
    effect:?4[mover:?2,
              goal:?3]
]}
```

XMG implementation of frame semantics

Examples

[from Lichte/Petitjean 2015]

Specification of frames:




```
<frame>{
?0[causation,
    actor:?1,
    theme:?2,
    cause:[activity,
           actor:?1,
           theme:?2],
    effect:?4[mover:?2,
              goal:?3]
]}
```

Specification of attribute-value constraints:

```
frame_constraints = {
    activity -> event, activity -> [actor:+],
    motion -> event, motion -> [mover:+],
    causation -> event, causation -> [cause:+,effect:+],
    locomotion -> activity motion}
```