A high-level approach to language description

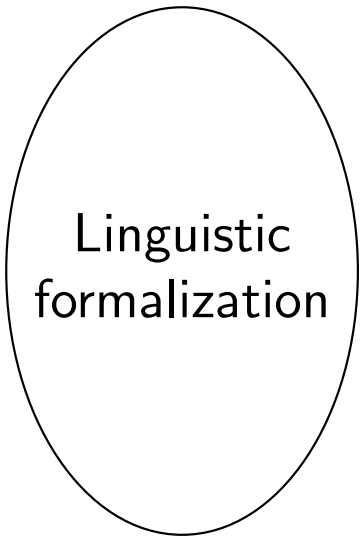
Lionel Clément and Jérôme Kirman and Sylvain Salvati

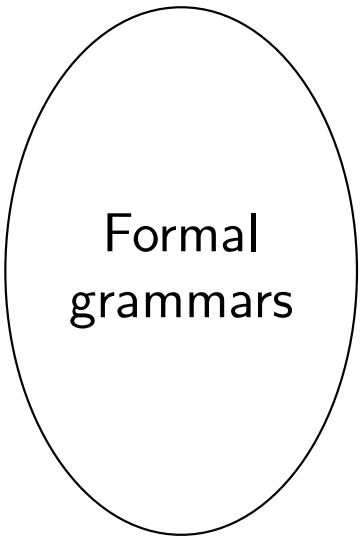
Université de Bordeaux - LaBRI
INRIA

Polymnie meeting

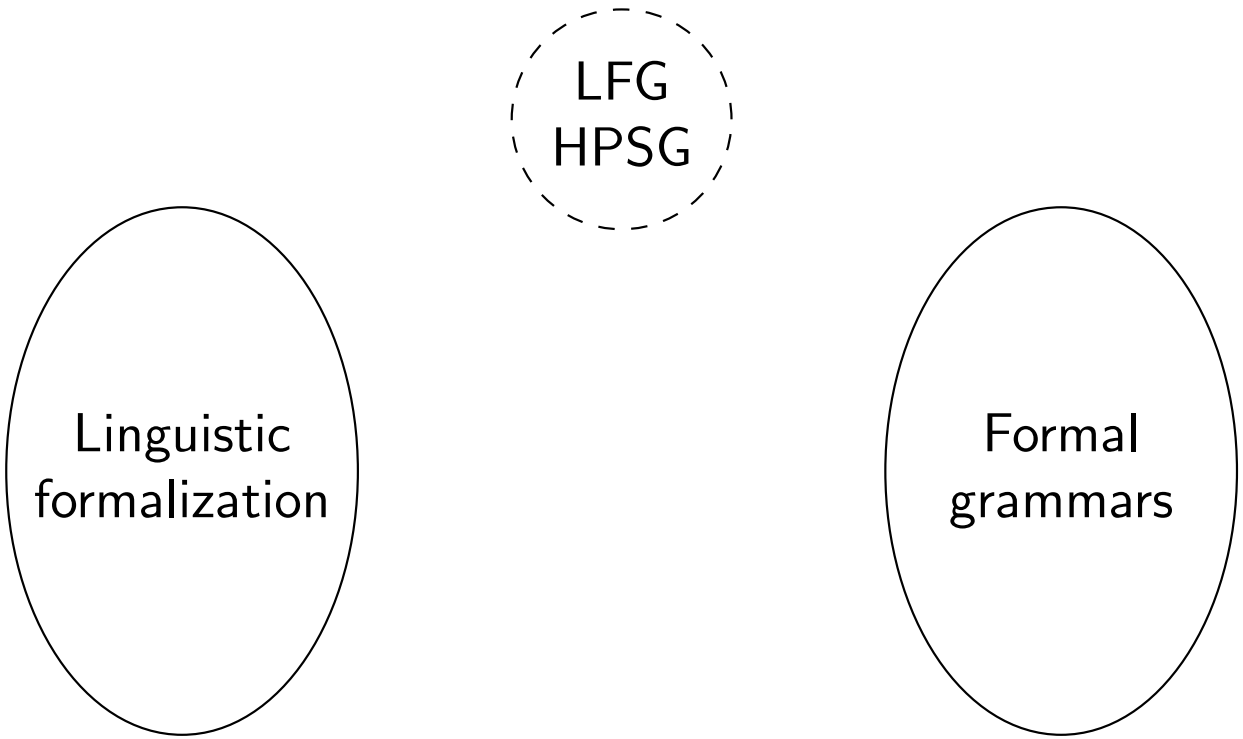
12 Mars 2014

Context and motivation

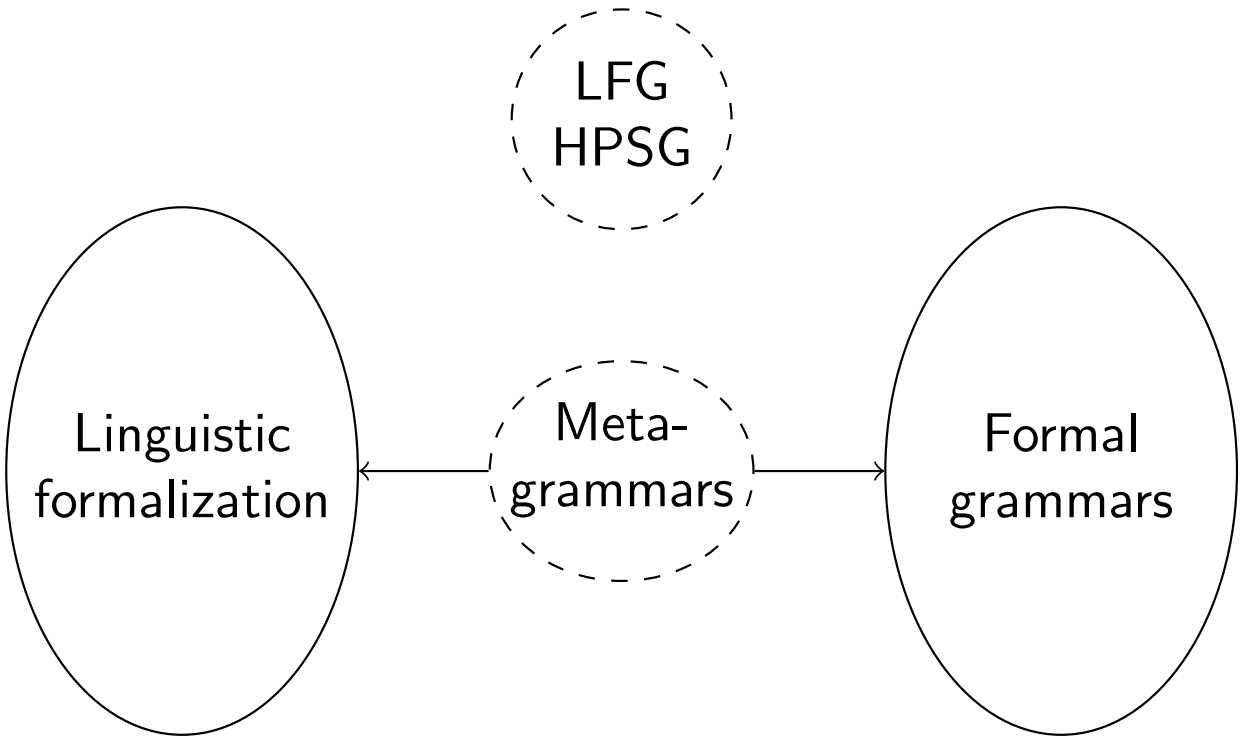




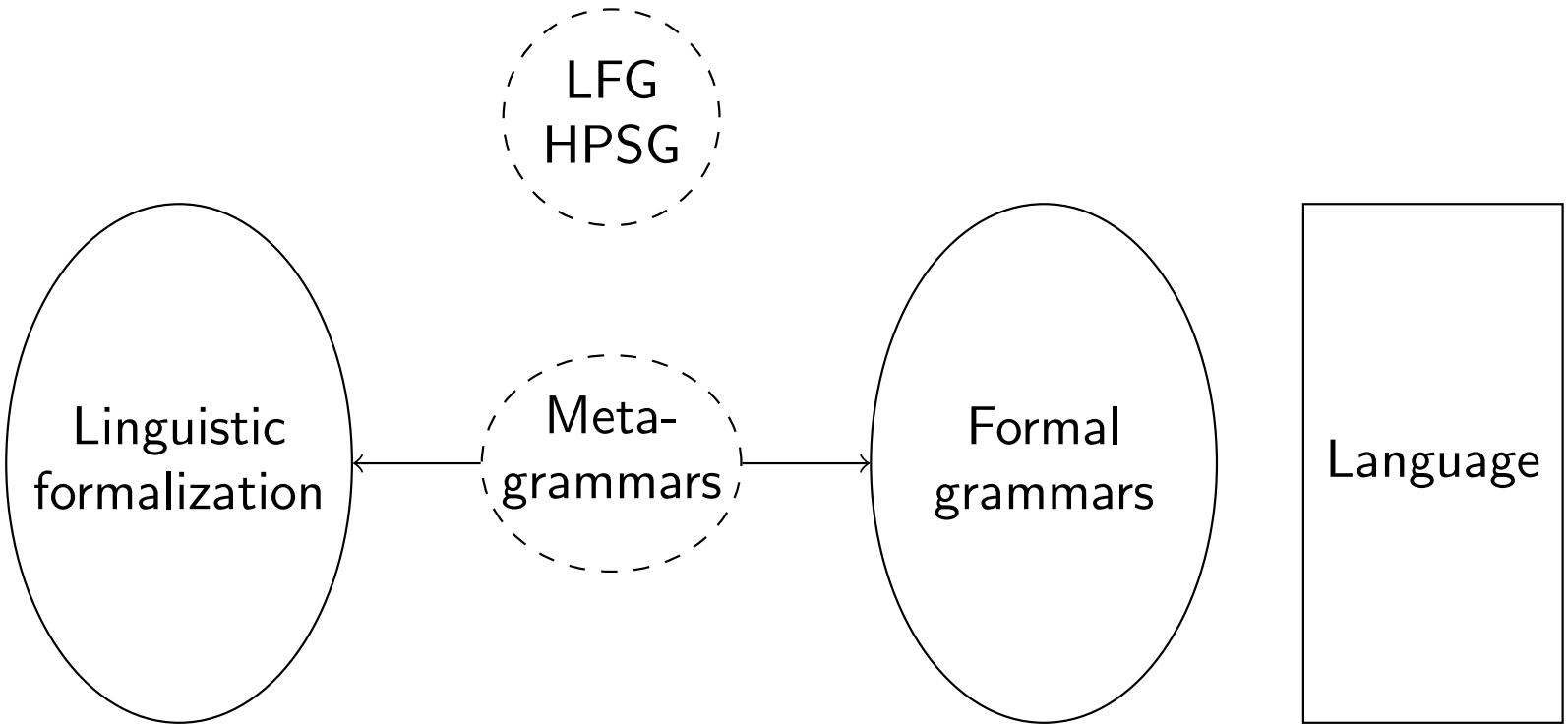
Context and motivation



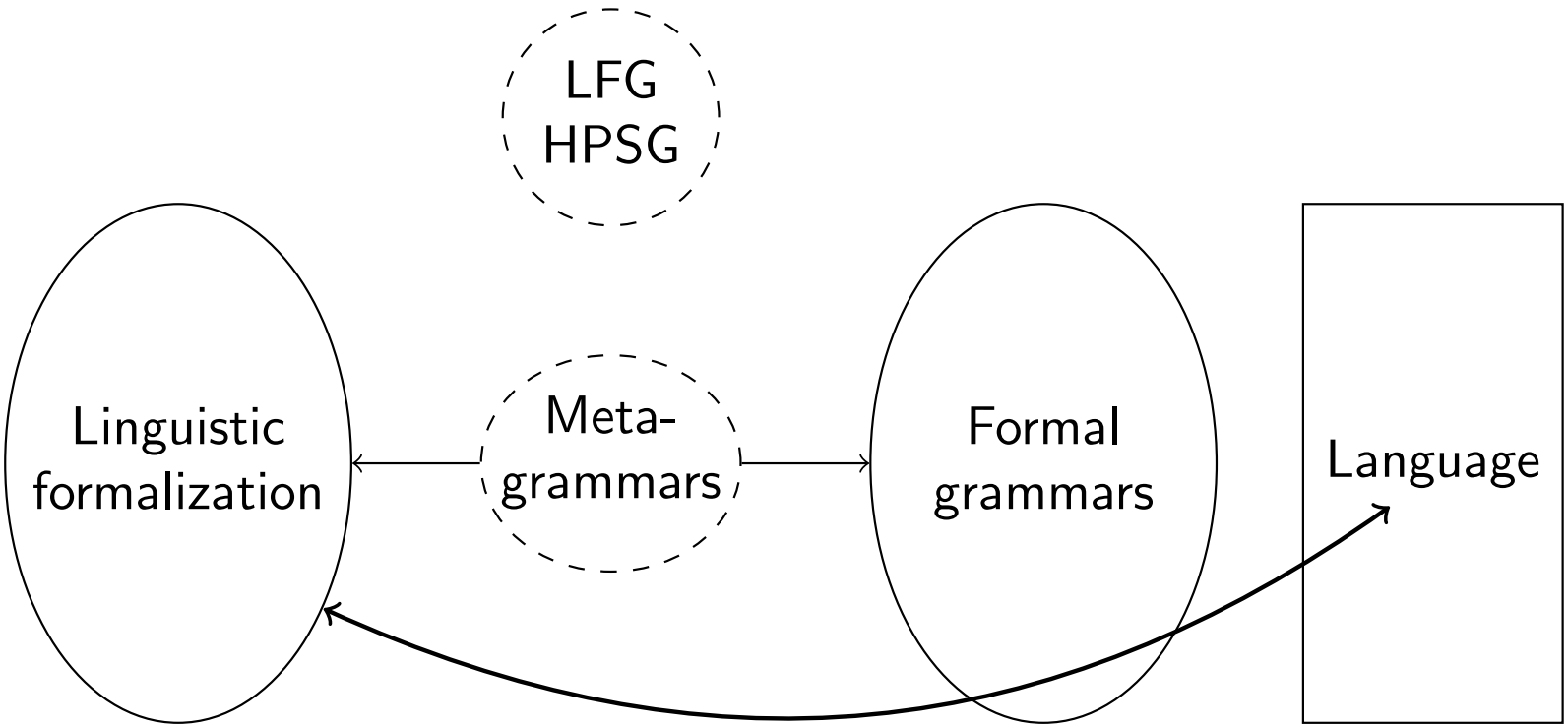
Context and motivation



Context and motivation



Context and motivation



Goals on both ends

Linguistic formalization

- High-level description of natural languages

Goals on both ends

Linguistic formalization

- High-level description of natural languages
- Highly modular

Goals on both ends

Linguistic formalization

- High-level description of natural languages
- Highly modular
- Close in spirit to meta-grammars

Goals on both ends

Linguistic formalization

- High-level description of natural languages
- Highly modular
- Close in spirit to meta-grammars

Target class of languages

- Mildly Context Sensitive Languages

Goals on both ends

Linguistic formalization

- High-level description of natural languages
- Highly modular
- Close in spirit to meta-grammars

Target class of languages

- Mildly Context Sensitive Languages
- $\Rightarrow$ Polynomial parsability

Working hypotheses

Abstract Categorial Grammar:

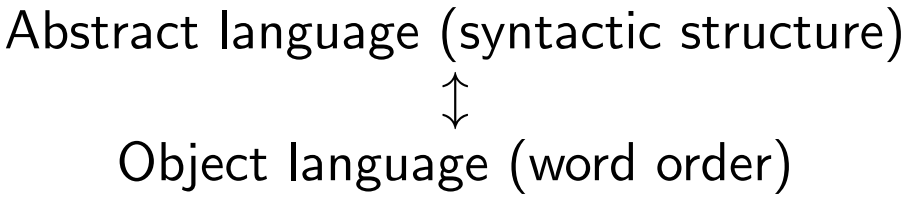
Abstract language (syntactic structure)

$\updownarrow$

Object language (word order)

Working hypotheses

Abstract Categorial Grammar:



Abstract structure = tree (hierarchy of syntactic components)

Working hypotheses

Abstract Categorial Grammar:

Abstract language (syntactic structure)

$\updownarrow$

Object language (word order)

Abstract structure = tree (hierarchy of syntactic components)

Description language = logic on trees (MTS)

Working hypotheses

Abstract Categorial Grammar:

Abstract language (syntactic structure)
↕
Object language (word order)

Abstract structure = tree (hierarchy of syntactic components)

Description language = logic on trees (MTS)

- Validity of an abstract structure → conjunction of logical constraints
- Non-local phenomena → logical long-distance relations
- Also helps defining the linearization

Working hypotheses

Abstract Categorial Grammar:

Abstract language (syntactic structure)
$\updownarrow$
Object language (word order)

Abstract structure = tree (hierarchy of syntactic components)

Description language = logic on trees (MTS)

- Validity of an abstract structure $\rightarrow$ conjunction of logical constraints
- Non-local phenomena $\rightarrow$ logical long-distance relations
- Also helps defining the linearization

Linearization: mapping from abstract to surface structures
= linear composition of lexical entries

Existing tools and results

Base tools

- Logic (MSO or weaker on trees)
- λ-calculus (simply-typed, linear)

Existing tools and results

Base tools

- Logic (MSO or weaker on trees)
- λ-calculus (simply-typed, linear)

J. E. Doner. (*MSOkS = FSTA*)
Decidability of the weak second-order theory of two successors.
Notices Amer. Math. Soc., 12:365–468, 1965.

Existing tools and results

Base tools

- Logic (MSO or weaker on trees)
- λ-calculus (simply-typed, linear)

J. E. Doner. ($MSOkS = FSTA$)
Decidability of the weak second-order theory of two successors.
Notices Amer. Math. Soc., 12:365–468, 1965.

B. Courcelle and J. Engelfriet. (HR are closed by MSO transductions)
A logical characterization of the sets of hypergraphs defined by hyperedge replacement grammars.
Mathematical Systems Theory, 28(6):515–552, 1995.

S. Salvati. ($HR \cap T \rightarrow 2\,ACG_{lin} \cap T$)
Encoding second order string acg with deterministic tree walking transducers.
In S. Wintner, editor, *Proceedings FG 2006: the 11th conference on Formal Grammars*, FG Online Proceedings, pages 143–156. CSLI Publications, 2007.

Example grammar

Construct a simple grammar for French, that covers:

- Simple SVO structure
- Valency and subject-verb agreement

Example grammar

Construct a simple grammar for French, that covers:

- Simple SVO structure
- Valency and subject-verb agreement

Describe the semantics of the formalization as we go

Example grammar

Construct a simple grammar for French, that covers:

- Simple SVO structure
- Valency and subject-verb agreement

Describe the semantics of the formalization as we go

Enrich the grammar to include

- Relative clauses as modifiers
- Complement clauses (for long distance wh-movement)
- Island constraints

Abstract structures

Abstract language = set of *abstract structure* trees

Trees have:

- Unlabelled internal nodes
- Labelled edges (bounded arity)
- Leaves are lexical entries or $\perp$

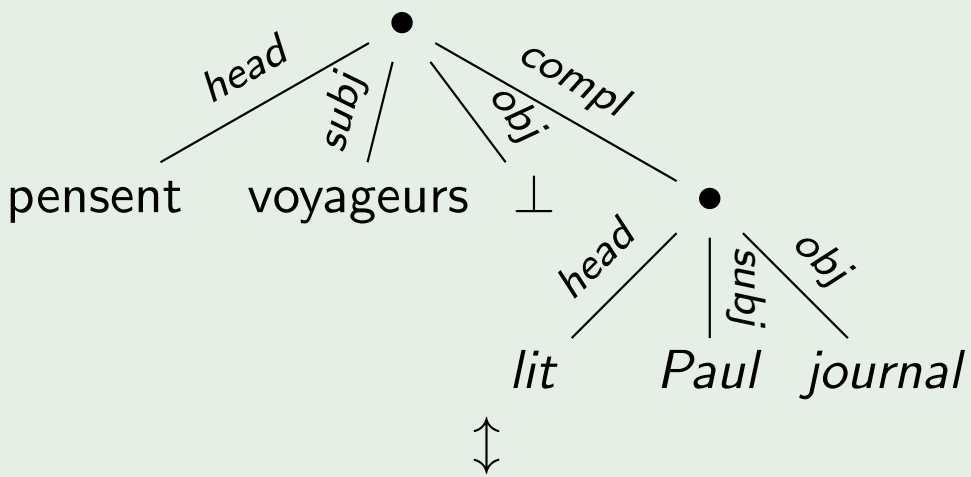
Abstract structures

Abstract language = set of *abstract structure* trees

Trees have:

- Unlabelled internal nodes
- Labelled edges (bounded arity)
- Leaves are lexical entries or $\perp$

An abstract structure



Des voyageurs pensent que Paul lit le journal.

Lexical entries

Lexical entries decorate leaves in the abstract structure tree

Each entry has a set of associated properties (from a finite set)

Lexical entries

Lexical entries decorate leaves in the abstract structure tree

Each entry has a set of associated properties (from a finite set)

A lexicon

Entry	Properties
le journal	*noun*, *def* , *masculine*, *singular*
des voyageurs	*noun*, *indef* , *masculine*, *plural*
Paul	*proper noun*, *masculine*, *singular*
lit	*verb*, *transitive*, *intransitive*, *singular* , *3rd pers*
lisent	*verb*, *transitive*, *intransitive*, *plural* , *3rd pers*
marche	*verb*, *intransitive*, *singular* , *3rd pers*

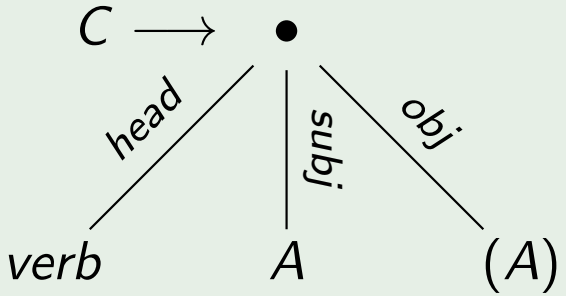
Regular backbone

Definition of abstract structures by a regular tree grammar (over-approximation)

Regular backbone

Definition of abstract structures by a regular tree grammar (over-approximation)

A simple over-approximation RTG



$$A \longrightarrow noun \mid proper\ noun$$

Terminals are POS properties (any lexical entry with the right POS tag)

Some arguments may be optional (between parentheses): $(T) \rightarrow T \mid \bot$

Logical vocabulary (1/2)

Variables ($x, y, z, \dots$): positions in the tree

Logical vocabulary (1/2)

Variables ($x, y, z, \dots$): positions in the tree

Predicates and relations:

- $prop(x) :=$ x is a lexical entry that has the property *prop*

Logical vocabulary (1/2)

Variables ($x, y, z, \dots$): positions in the tree

Predicates and relations:

- $prop(x) :=$ x is a lexical entry that has the property $prop$
- $none(x) := x = \bot$; $some(x) := \neg none(x)$

Logical vocabulary (1/2)

Variables ($x, y, z, \ldots$): positions in the tree

Predicates and relations:

- $prop(x)$:= x is a lexical entry that has the property *prop*
- $none(x) := x = \bot$; $some(x) := \neg none(x)$
- $lbl(x, y)$:= x immediately dominates y by an arc labelled with *lbl*

Logical vocabulary (1/2)

Variables ($x, y, z, \ldots$): positions in the tree

Predicates and relations:

- $prop(x) :=$ x is a lexical entry that has the property $prop$
- $none(x) := x = \bot$; $some(x) := \neg none(x)$
- $lbl(x, y) :=$ x immediately dominates y by an arc labelled with lbl
- $regex(x, y) :=$ x dominates y by a series of arcs $l_1 \ldots l_n \in L(regex)$ (with $regex := \varepsilon | lbl_1.lbl_2 | lbl_1 + lbl_2 | lbl^*$)

Logical vocabulary (1/2)

Variables ($x, y, z, \ldots$): positions in the tree

Predicates and relations:

- $prop(x) :=$ x is a lexical entry that has the property $prop$
- $none(x) := x = \bot$; $some(x) := \neg none(x)$
- $lbl(x, y) :=$ x immediately dominates y by an arc labelled with lbl
- $regex(x, y) :=$ x dominates y by a series of arcs $l_1 \ldots l_n \in L(regex)$ (with $regex := \varepsilon | lbl_1.lbl_2 | lbl_1 + lbl_2 | lbl^*$)

Used to restrict the set of valid abstract structures:

- Sub-categorization, selection restrictions, etc.
- Regexps allow long-distance constraints

Logical vocabulary (2/2)

Linguistic formalization by building additional predicates and relations

Logical vocabulary (2/2)

Linguistic formalization by building additional predicates and relations

Agreement relations

$$numb\_agr(x,y) := singular(x) \wedge singular(y) \vee plural(x) \wedge plural(y)$$

$$gend\_agr(x,y) := masculine(x) \wedge masculine(y) \vee feminine(x) \wedge feminine(y)$$

$$third\_pers(x) := 3rd\ pers(x) \vee \neg(1st\ pers(x) \vee 2nd\ pers(x))$$

$$pers\_agr(x,y) := 1st\ pers(x) \wedge 1st\ pers(y) \vee 2nd\ pers(x) \wedge 2nd\ pers(y) \vee third\_pers(x) \wedge third\_pers(y)$$

Logical vocabulary (2/2)

Linguistic formalization by building additional predicates and relations

Agreement relations

$$\begin{aligned} numb\_agr(x,y) := & \; singular(x) \wedge singular(y) \\ \vee & \; plural(x) \wedge plural(y) \end{aligned}$$

$$\begin{aligned} gend\_agr(x,y) := & \; masculine(x) \wedge masculine(y) \\ \vee & \; feminine(x) \wedge feminine(y) \end{aligned}$$

$$third\_pers(x) := 3rd\ pers(x) \vee \neg(1st\ pers(x) \vee 2nd\ pers(x))$$

$$\begin{aligned} pers\_agr(x,y) := & \; 1st\ pers(x) \wedge 1st\ pers(y) \\ \vee & \; 2nd\ pers(x) \wedge 2nd\ pers(y) \\ \vee & \; third\_pers(x) \wedge third\_pers(y) \end{aligned}$$

Subject-verb agreement relation

$$sv\_agr(x,y) := numb\_agr(x,y) \wedge pers\_agr(x,y)$$

Constraints on validity

Valid abstract structures must satisfy additional constraints on the RTG

Constraints on validity

Valid abstract structures must satisfy additional constraints on the RTG

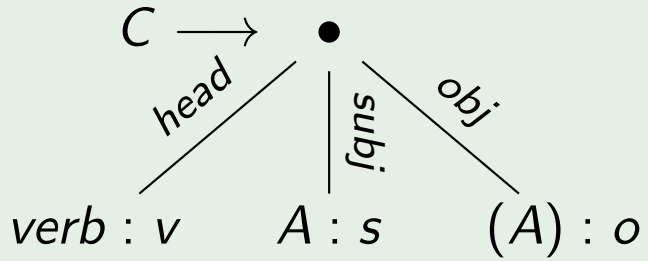
Constraints are logical formulas tied to the RHS of productions

Constraints on validity

Valid abstract structures must satisfy additional constraints on the RTG

Constraints are logical formulas tied to the RHS of productions

Enforcing subject-verb agreement and valency



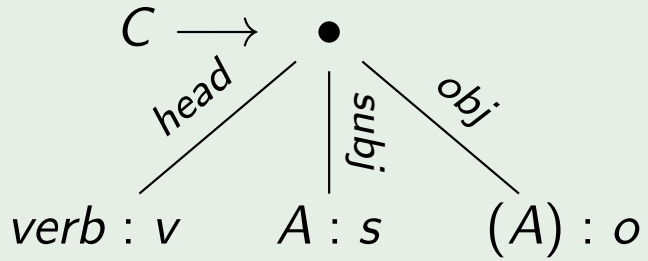
$$some(o) \Rightarrow transitive(v)$$
$$none(o) \Rightarrow intransitive(v)$$
$$sv\_agr(s, v)$$

Constraints on validity

Valid abstract structures must satisfy additional constraints on the RTG

Constraints are logical formulas tied to the RHS of productions

Enforcing subject-verb agreement and valency



$$some(o) \Rightarrow transitive(v)$$
$$none(o) \Rightarrow intransitive(v)$$
$$sv\_agr(s, v)$$

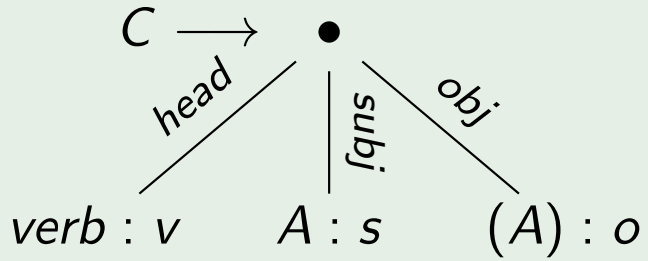
Filtered structures

Constraints on validity

Valid abstract structures must satisfy additional constraints on the RTG

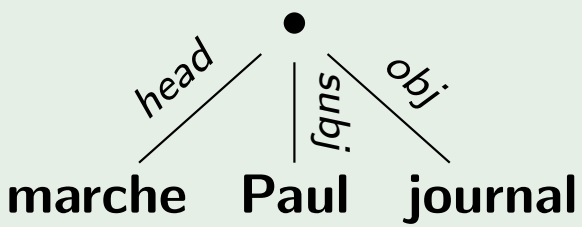
Constraints are logical formulas tied to the RHS of productions

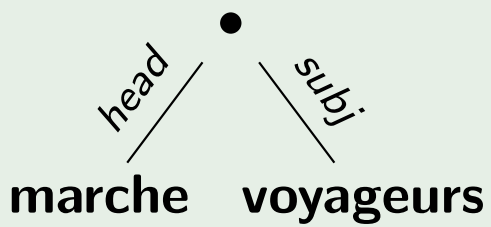
Enforcing subject-verb agreement and valency

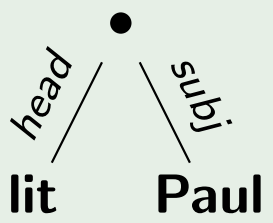


$$some(o) \Rightarrow transitive(v)$$
$$none(o) \Rightarrow intransitive(v)$$
$$sv\_agr(s, v)$$

Filtered structures







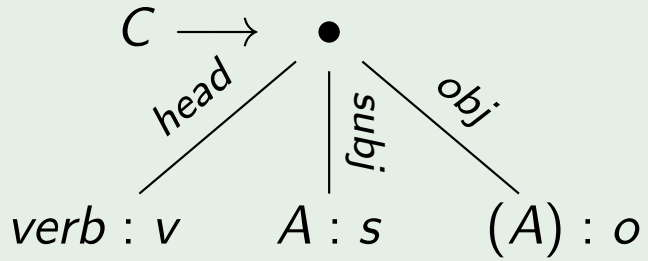
$some(o) \wedge \neg transitive(v)$

Constraints on validity

Valid abstract structures must satisfy additional constraints on the RTG

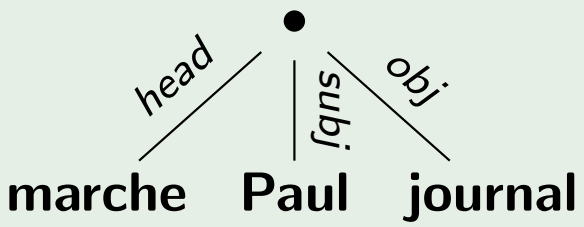
Constraints are logical formulas tied to the RHS of productions

Enforcing subject-verb agreement and valency

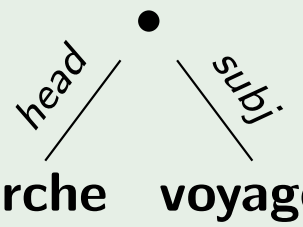


$$some(o) \Rightarrow transitive(v)$$
$$none(o) \Rightarrow intransitive(v)$$
$$sv\_agr(s, v)$$

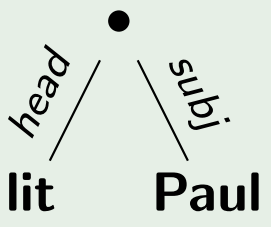
Filtered structures



$some(o) \wedge \neg transitive(v)$



$\neg plural(v) \wedge \neg singular(s)$

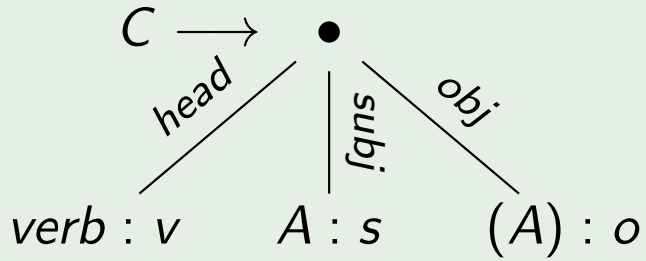


Constraints on validity

Valid abstract structures must satisfy additional constraints on the RTG

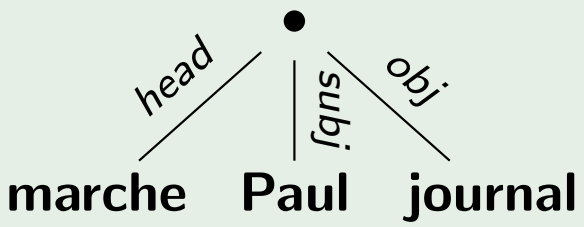
Constraints are logical formulas tied to the RHS of productions

Enforcing subject-verb agreement and valency

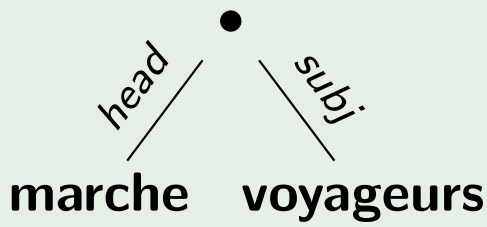


$$some(o) \Rightarrow transitive(v)$$
$$none(o) \Rightarrow intransitive(v)$$
$$sv\_agr(s, v)$$

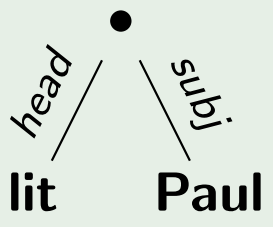
Filtered structures



$some(o) \wedge \neg transitive(v)$



$\neg plural(v) \wedge \neg singular(s)$



OK

Linearization rules

Like constraints, linearizations will be based on the RTG productions

Linearization rules

Like constraints, linearizations will be based on the RTG productions

The realization attached to the LHS will depend on :

- Logical conditions
- Realizations of other nodes (mostly RHS leaves)

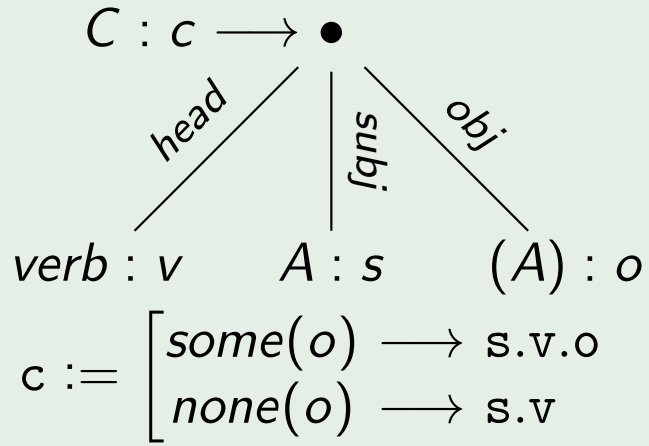
Linearization rules

Like constraints, linearizations will be based on the RTG productions

The realization attached to the LHS will depend on :

- Logical conditions
- Realizations of other nodes (mostly RHS leaves)

A linearization rule



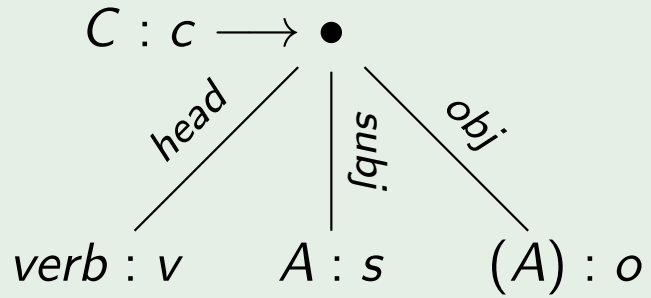
Linearization rules

Like constraints, linearizations will be based on the RTG productions

The realization attached to the LHS will depend on :

- Logical conditions
- Realizations of other nodes (mostly RHS leaves)

A linearization rule



$\mathtt{c} := \mathtt{s.v.o}$ $\quad$ $(lin(\bot) := \varepsilon)$

Adding relative clauses (1/3)

We enrich the previous grammar to include relative clauses

Adding relative clauses (1/3)

We enrich the previous grammar to include relative clauses

Lexical entries for french relative pronouns

Entry	Properties
qui	*pronoun*, *rel_pro*, *masculine*, *feminine*, *singular*, *plural*, *1st*, *2nd*, *3rd pers.*, *nom*
que	*pronoun*, *rel_pro*, *masculine*, *feminine*, *singular*, *plural*, *1st*, *2nd*, *3rd pers.*, *acc*

Adding relative clauses (1/3)

We enrich the previous grammar to include relative clauses

Lexical entries for french relative pronouns

Entry	Properties
qui	*pronoun*, *rel_pro*, *masculine*, *feminine*, *singular*, *plural*, *1st*, *2nd*, *3rd pers.*, *nom*
que	*pronoun*, *rel_pro*, *masculine*, *feminine*, *singular*, *plural*, *1st*, *2nd*, *3rd pers.*, *acc*

Logical vocabulary for wh-movement and agreement in relatives

$$wh\_path(x, y) := subj(x, y) \vee obj(x, y)$$

$$ext(x) := rel\_pro(x)$$

$$rel(r) := \exists np.mod(np, r) \wedge \exists v.verb(v) \wedge head(r, v)$$

$$ap\_agr(a, p) := pers\_agr(a, p) \wedge numb\_agr(a, p) \wedge gend\_agr(a, p)$$

Adding relative clauses (1/3)

We enrich the previous grammar to include relative clauses

Lexical entries for french relative pronouns

Entry	Properties
qui	*pronoun*, *rel_pro*, *masculine*, *feminine*, *singular*, *plural*, *1st*, *2nd*, *3rd pers.*, *nom*
que	*pronoun*, *rel_pro*, *masculine*, *feminine*, *singular*, *plural*, *1st*, *2nd*, *3rd pers.*, *acc*

Logical vocabulary for wh-movement and agreement in relatives

$$wh\_path(x, y) := subj(x, y) \vee obj(x, y)$$

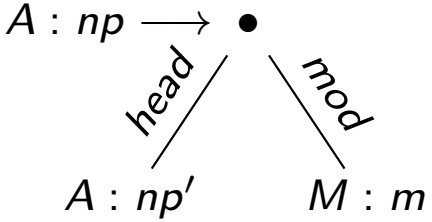
$$ext(x) := rel\_pro(x)$$

$$rel(r) := \exists np.mod(np, r) \wedge \exists v.verb(v) \wedge head(r, v)$$

$$ap\_agr(a, p) := pers\_agr(a, p) \wedge numb\_agr(a, p) \wedge gend\_agr(a, p)$$

$$antecedent(a, p) := \exists np, r.head^*(np, a) \wedge mod(np, r) \wedge wh\_path(r, p)$$

Adding relative clauses (2/3)



$A : a \longrightarrow pronoun : p$

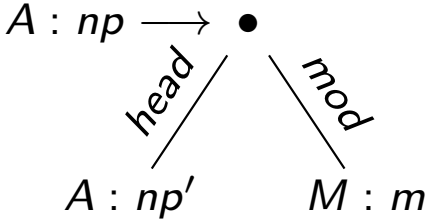
$M : m \longrightarrow C : r$

$\mathtt{np} := \mathtt{np}'.\mathtt{m}$

$\mathtt{a} := \mathtt{p}$

$\mathtt{m} := wh\_path(r, \mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$

Adding relative clauses (2/3)



$A : a \longrightarrow pronoun : p$

$M : m \longrightarrow C : r$

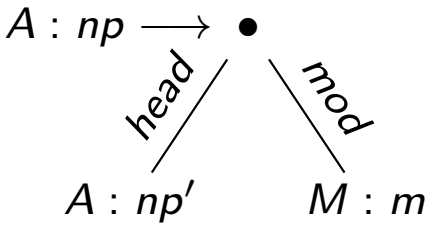
$\neg rel\_pro(np')$

$\mathtt{np} := \mathtt{np}'.\mathtt{m}$

$\mathtt{a} := \mathtt{p}$

$\mathtt{m} := wh\_path(r, \mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$

Adding relative clauses (2/3)



$A : a \longrightarrow pronoun : p$

$M : m \longrightarrow C : r$

$\neg rel\_pro(np')$

$\mathtt{np} := \mathtt{np}'.\mathtt{m}$

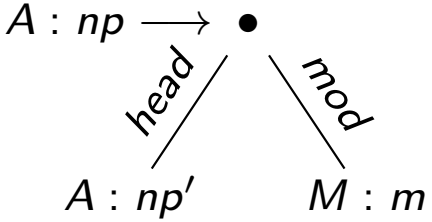
$\mathtt{a} := \mathtt{p}$

$\mathtt{m} := wh\_path(r, \mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$

Filtered out sentences - [NP+MOD] (SUJ+VB+OBJ)

*[la pomme ([qui (que regarde Paul)] tombe)]

Adding relative clauses (2/3)



$A : a \longrightarrow pronoun : p$

$M : m \longrightarrow C : r$

$$rel\_pro(p) \Rightarrow \exists r.rel(r) \wedge wh\_path(r,p) \wedge \exists a.antecedent(a,p) \wedge ap\_agr(a,p)$$

$$\neg rel\_pro(np')$$

$\mathtt{np} := \mathtt{np}'.\mathtt{m}$

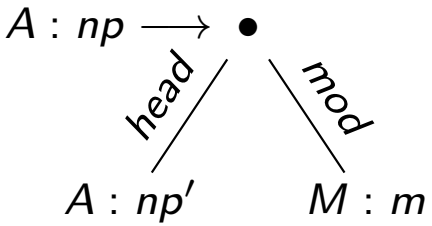
$\mathtt{a} := \mathtt{p}$

$\mathtt{m} := wh\_path(r, \mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$

Filtered out sentences - [NP+MOD] (SUJ+VB+OBJ)

*[la pomme ([qui (que regarde Paul)] tombe)]

Adding relative clauses (2/3)



$A : a \longrightarrow pronoun : p$

$M : m \longrightarrow C : r$

$$rel\_pro(p) \Rightarrow \exists r.rel(r) \wedge wh\_path(r,p) \wedge \exists a.antecedent(a,p) \wedge ap\_agr(a,p)$$

$\neg rel\_pro(np')$

$\mathtt{np} := \mathtt{np}'.\mathtt{m}$

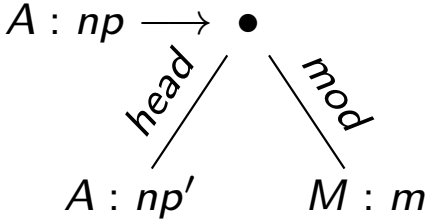
$\mathtt{a} := \mathtt{p}$

$\mathtt{m} := wh\_path(r, \mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$

Filtered out sentences - [NP+MOD] (SUJ+VB+OBJ)

*[la pomme ([qui (que regarde Paul)] tombe)]
*(qui tombe)

Adding relative clauses (2/3)



$$A : a \longrightarrow pronoun : p$$

$$M : m \longrightarrow C : r$$

$$\neg rel\_pro(np')$$

$$\begin{aligned} rel\_pro(p) \Rightarrow\ & \exists r.rel(r) \wedge wh\_path(r,p) \\ & \wedge \exists a.antecedent(a,p) \\ & \wedge ap\_agr(a,p) \\ nom(p) \Rightarrow\ & \exists x.subj(x,p) \\ acc(p) \Rightarrow\ & \exists x.obj(x,p) \end{aligned}$$

$$\mathtt{np} := \mathtt{np}'.\mathtt{m}$$

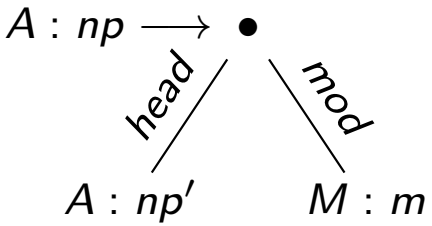
$$\mathtt{a} := \mathtt{p}$$

$$\mathtt{m} := wh\_path(r, \mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$$

Filtered out sentences - [NP+MOD] (SUJ+VB+OBJ)

*[la pomme ([qui (que regarde Paul)] tombe)]
*(qui tombe)

Adding relative clauses (2/3)



$$A : a \longrightarrow pronoun : p \qquad M : m \longrightarrow C : r$$

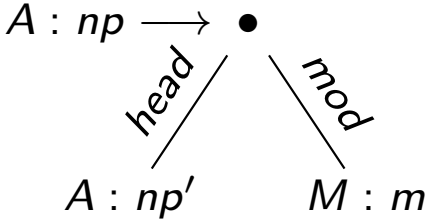
$$\neg rel\_pro(np') \qquad \begin{aligned} rel\_pro(p) \Rightarrow\ & \exists r.rel(r) \wedge wh\_path(r,p) \\ & \wedge \exists a.antecedent(a,p) \\ & \wedge ap\_agr(a,p) \\ nom(p) \Rightarrow\ & \exists x.subj(x,p) \\ acc(p) \Rightarrow\ & \exists x.obj(x,p) \end{aligned}$$

$$\mathtt{np} := \mathtt{np}'.\mathtt{m} \qquad \mathtt{a} := \mathtt{p} \qquad \mathtt{m} := wh\_path(r, \mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$$

Filtered out sentences - [NP+MOD] (SUJ+VB+OBJ)

*[la pomme ([qui (que regarde Paul)] tombe)]
*(qui tombe)
*[la pomme (que tombe)] ; *[la pomme (qui Paul regarde)]

Adding relative clauses (2/3)



$$A : a \longrightarrow pronoun : p$$

$$M : m \longrightarrow C : r$$

$$\begin{aligned} rel\_pro(p) \Rightarrow\ & \exists r.rel(r) \wedge wh\_path(r,p) \\ & \wedge \exists a.antecedent(a,p) \\ & \wedge ap\_agr(a,p) \\ nom(p) \Rightarrow\ & \exists x.subj(x,p) \\ acc(p) \Rightarrow\ & \exists x.obj(x,p) \end{aligned}$$

$$\neg rel\_pro(np')$$

$$\exists! p.wh\_path(r,p) \wedge rel\_pro(p)$$

$$\mathtt{np} := \mathtt{np}'.\mathtt{m}$$

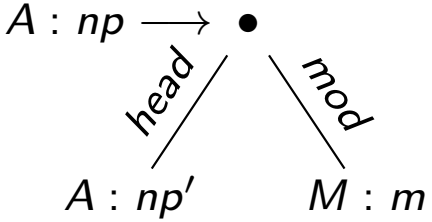
$$\mathtt{a} := \mathtt{p}$$

$$\mathtt{m} := wh\_path(r,\mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$$

Filtered out sentences - [NP+MOD] (SUJ+VB+OBJ)

*[la pomme ([qui (que regarde Paul)] tombe)]
*(qui tombe)
*[la pomme (que tombe)] ; *[la pomme (qui Paul regarde)]

Adding relative clauses (2/3)



$$A : a \longrightarrow pronoun : p \qquad M : m \longrightarrow C : r$$

$$\begin{aligned} rel\_pro(p) \Rightarrow\ & \exists r.rel(r) \wedge wh\_path(r,p) \\ & \wedge \exists a.antecedent(a,p) \\ & \wedge ap\_agr(a,p) \\ nom(p) \Rightarrow\ & \exists x.subj(x,p) \\ acc(p) \Rightarrow\ & \exists x.obj(x,p) \end{aligned}$$

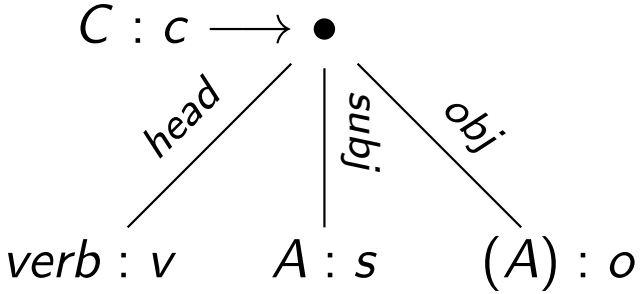
$$\neg rel\_pro(np') \qquad \exists!p.wh\_path(r,p) \wedge rel\_pro(p)$$

$$\mathtt{np} := \mathtt{np}'.\mathtt{m} \qquad \mathtt{a} := \mathtt{p} \qquad \mathtt{m} := wh\_path(r,\mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$$

Filtered out sentences - [NP+MOD] (SUJ+VB+OBJ)

*[la pomme ([qui (que regarde Paul)] tombe)]
*(qui tombe)
*[la pomme (que tombe)] ; *[la pomme (qui Paul regarde)]
*[la pomme (Paul regarde Marie)] ; *[Paul (qui que regarde)]

Adding relative clauses (3/3)

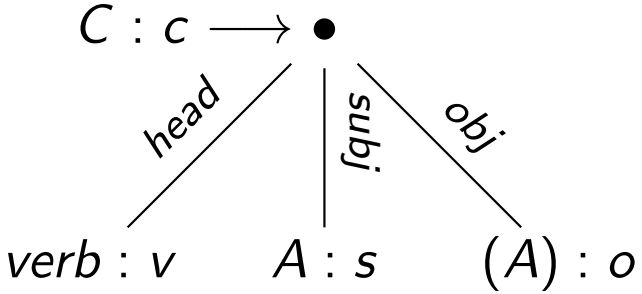


$$M : m \longrightarrow C : r$$

$$\mathtt{c} := \begin{bmatrix} ext(s) & \longrightarrow & \mathtt{v.o} \\ ext(o) & \longrightarrow & \mathtt{s.v} \\ else & \longrightarrow & \mathtt{s.v.o} \end{bmatrix}$$

$$\mathtt{m} := wh\_path(r, \mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$$

Adding relative clauses (3/3)

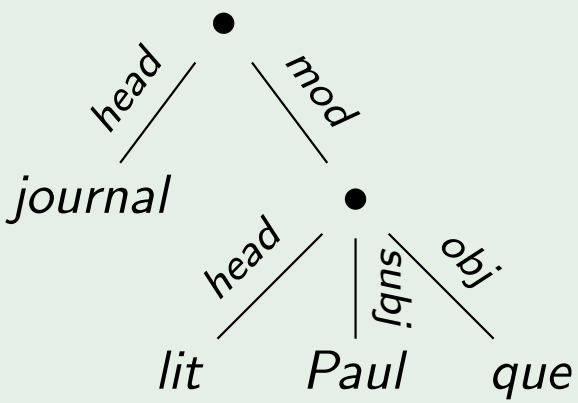


$$M : m \longrightarrow C : r$$

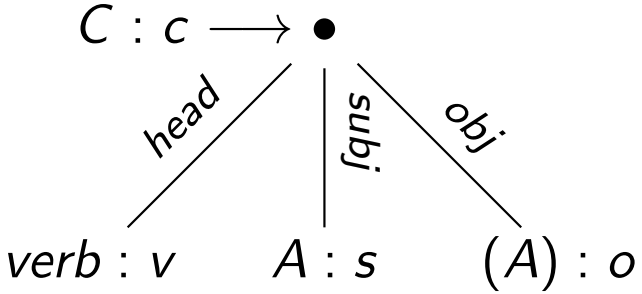
$$\mathtt{c} := \begin{bmatrix} ext(s) & \longrightarrow & \mathtt{v.o} \\ ext(o) & \longrightarrow & \mathtt{s.v} \\ else & \longrightarrow & \mathtt{s.v.o} \end{bmatrix}$$

$$\mathtt{m} := wh\_path(r, \mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$$

Wh- movement account



Adding relative clauses (3/3)

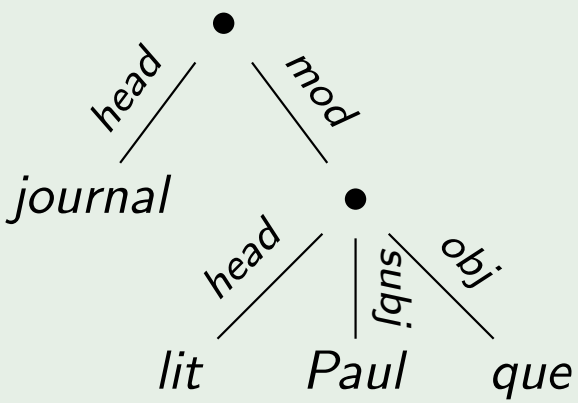


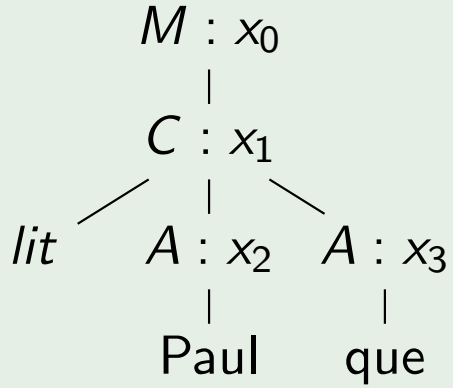
$$M : m \longrightarrow C : r$$

$$\mathtt{c} := \begin{bmatrix} ext(s) & \longrightarrow & \mathtt{v.o} \\ ext(o) & \longrightarrow & \mathtt{s.v} \\ else & \longrightarrow & \mathtt{s.v.o} \end{bmatrix}$$

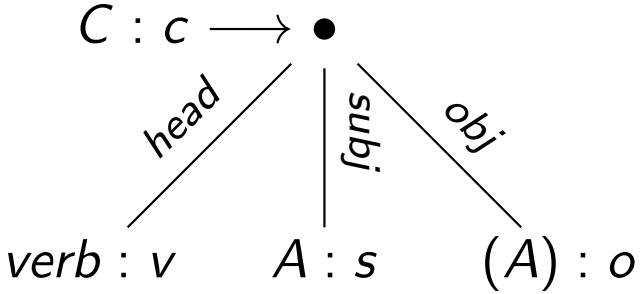
$$\mathtt{m} := wh\_path(r, \mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$$

Wh- movement account





Adding relative clauses (3/3)

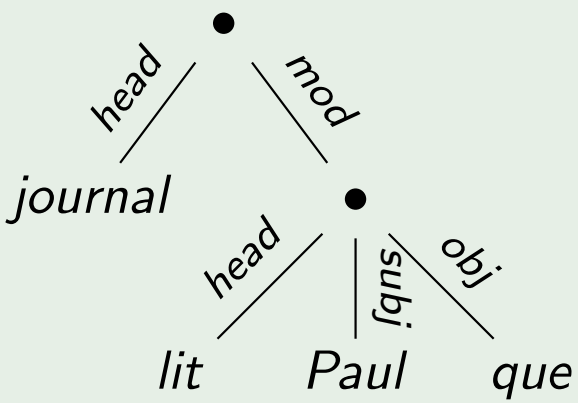


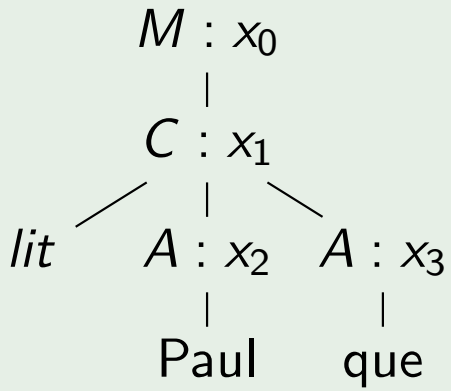
$$M : m \longrightarrow C : r$$

$$\mathtt{c} := \begin{bmatrix} ext(s) & \longrightarrow & \mathtt{v.o} \\ ext(o) & \longrightarrow & \mathtt{s.v} \\ else & \longrightarrow & \mathtt{s.v.o} \end{bmatrix}$$

$$\mathtt{m} := wh\_path(r, \mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$$

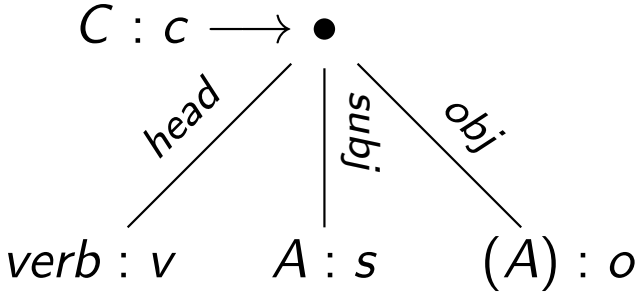
Wh- movement account





$$\mathtt{x}_2 = \mathtt{Paul}$$
$$\mathtt{x}_3 = \mathtt{que}$$

Adding relative clauses (3/3)

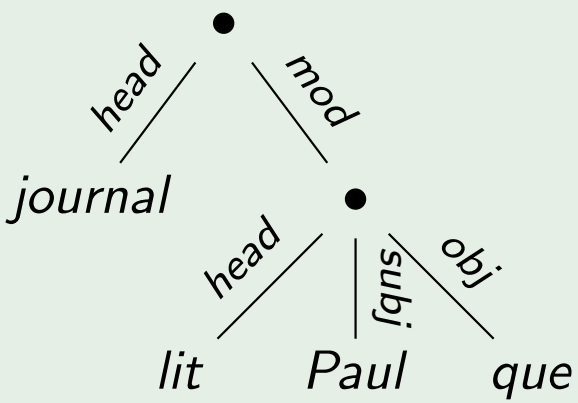


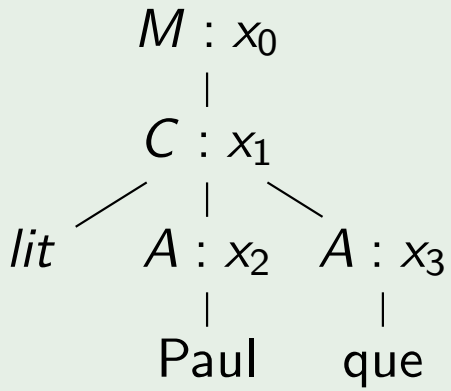
$$M : m \longrightarrow C : r$$

$$\mathtt{c} := \begin{bmatrix} ext(s) & \longrightarrow & \mathtt{v.o} \\ ext(o) & \longrightarrow & \mathtt{s.v} \\ else & \longrightarrow & \mathtt{s.v.o} \end{bmatrix}$$

$$\mathtt{m} := wh\_path(r, \mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$$

Wh- movement account

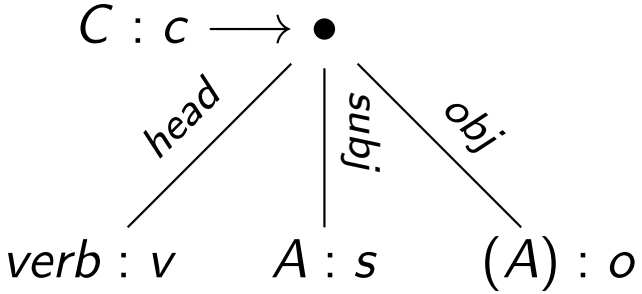




$$\mathtt{x}_1 = \mathtt{Paul.lit}$$

$$\mathtt{x}_2 = \mathtt{Paul}$$
$$\mathtt{x}_3 = \mathtt{que}$$

Adding relative clauses (3/3)

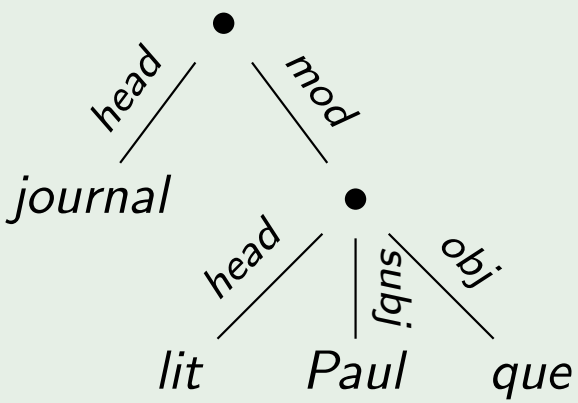


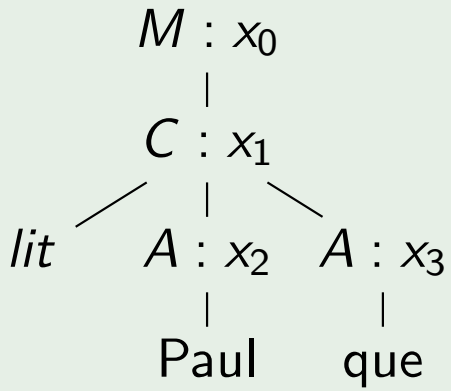
$$M : m \longrightarrow C : r$$

$$\mathtt{c} := \begin{bmatrix} ext(s) & \longrightarrow & \mathtt{v.o} \\ ext(o) & \longrightarrow & \mathtt{s.v} \\ else & \longrightarrow & \mathtt{s.v.o} \end{bmatrix}$$

$$\mathtt{m} := wh\_path(r, \mathbf{p}) \wedge ext(\mathbf{p}) \longrightarrow \mathbf{p}.\mathtt{r}$$

Wh- movement account





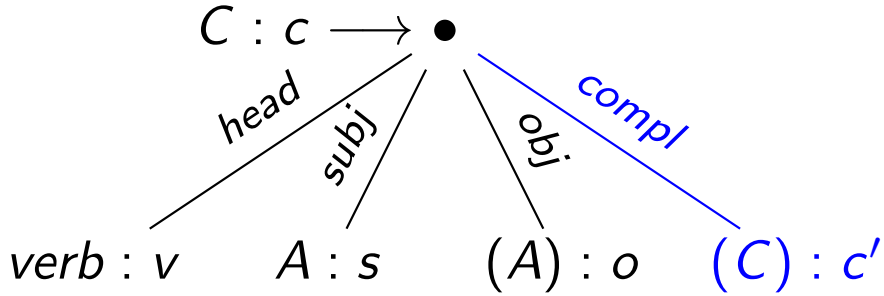
$$\mathtt{x}_0 = \mathtt{que.Paul\ lit}$$

$$\mathtt{x}_1 = \mathtt{Paul.lit}$$

$$\mathtt{x}_2 = \mathtt{Paul}$$

$$\mathtt{x}_3 = \mathtt{que}$$

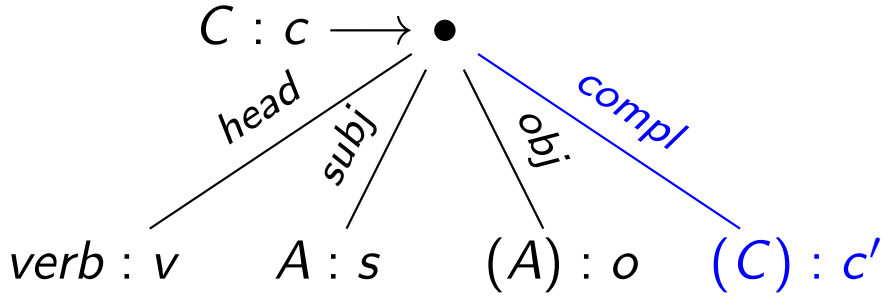
Adding complement clauses (1/2)



$$some(o) \Rightarrow transitive(v)$$
$$none(o) \Rightarrow intransitive(v)$$
$$sv\_agr(s, v)$$
$$some(c') \Leftrightarrow compl\_cl(v)$$

$$\mathrm{c} := \begin{bmatrix} ext(s) & \longrightarrow & \mathrm{v.o.cc} \\ ext(o) & \longrightarrow & \mathrm{s.v.cc} \\ else & \longrightarrow & \mathrm{s.v.o.cc} \end{bmatrix} \quad where\ \mathrm{cc} = \begin{bmatrix} some(c') & \longrightarrow & que.c' \\ else & \longrightarrow & \varepsilon \end{bmatrix}$$

Adding complement clauses (1/2)



$$some(o) \Rightarrow transitive(v)$$
$$none(o) \Rightarrow intransitive(v)$$
$$sv\_agr(s, v)$$
$$some(c') \Leftrightarrow compl\_cl(v)$$

$$\mathrm{c} := \begin{bmatrix} ext(s) & \longrightarrow & \mathrm{v.o.cc} \\ ext(o) & \longrightarrow & \mathrm{s.v.cc} \\ else & \longrightarrow & \mathrm{s.v.o.cc} \end{bmatrix} \quad where\ \mathrm{cc} = \begin{bmatrix} some(c') & \longrightarrow & que.c' \\ else & \longrightarrow & \varepsilon \end{bmatrix}$$

Addition to the lexicon

Entry	Properties
dit	*verb*, *intransitive*, *compl_cl*, *singular*, *3rd pers*
disent	*verb*, *intransitive*, *compl_cl*, *plural*, *3rd pers*

Adding complement clauses (2/2)

Updated logical definition of *wh_path* (Island constraints)

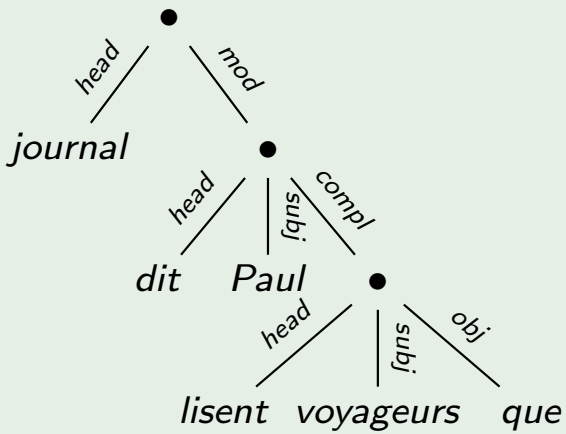
$$wh\_path(x, y) := subj(x, y) \vee compl^{*}\, obj(x, y)$$

Adding complement clauses (2/2)

Updated logical definition of *wh_path* (Island constraints)

$$wh\_path(x, y) := subj(x, y) \vee compl^{*}\, obj(x, y)$$

Final linearization example

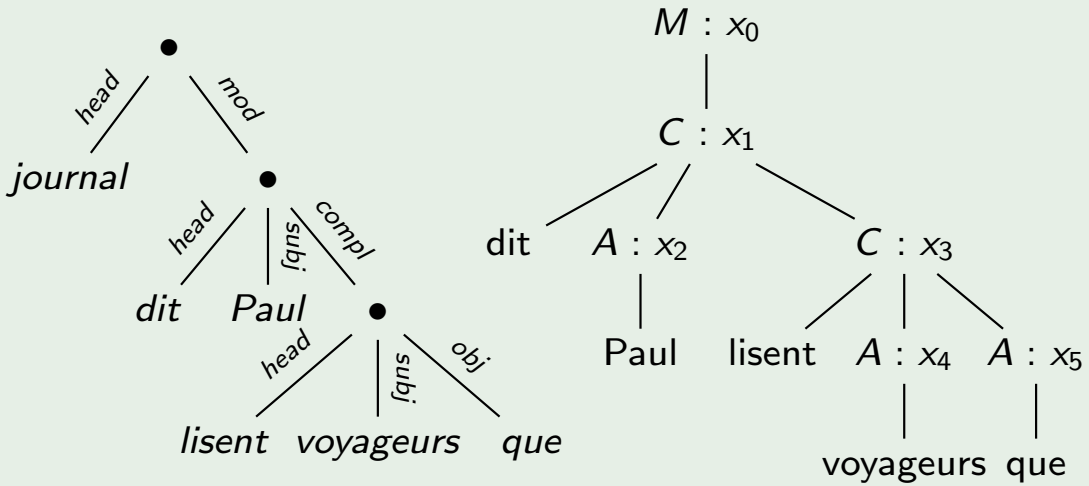


Adding complement clauses (2/2)

Updated logical definition of *wh_path* (Island constraints)

$$wh\_path(x, y) := subj(x, y) \vee compl^{*}\, obj(x, y)$$

Final linearization example

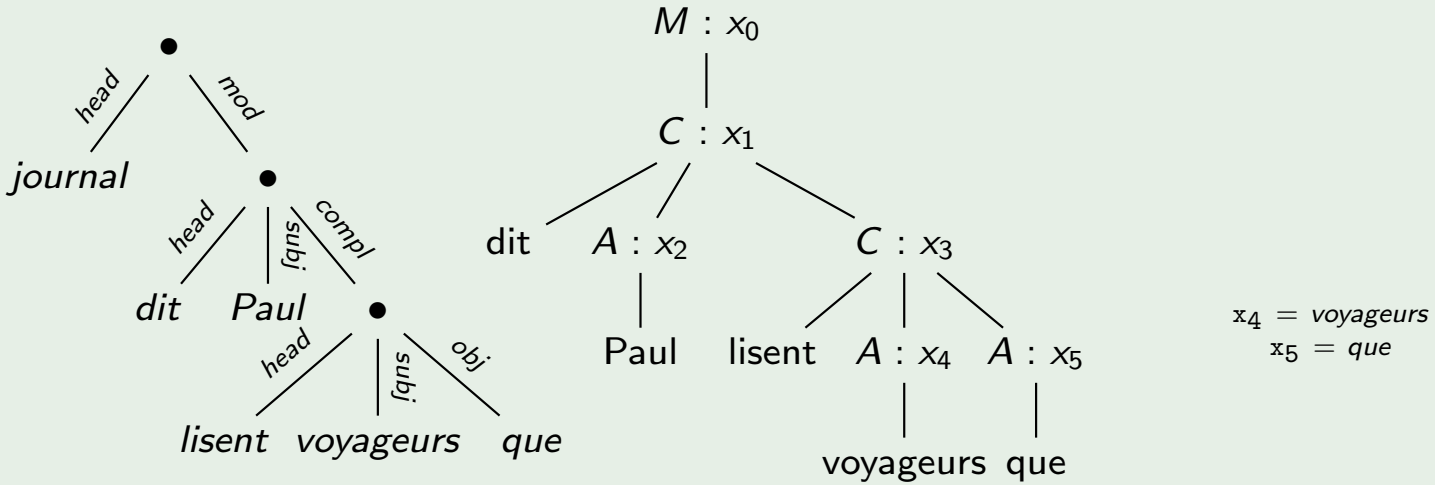


Adding complement clauses (2/2)

Updated logical definition of *wh_path* (Island constraints)

$$wh\_path(x, y) := subj(x, y) \vee compl^{*}\, obj(x, y)$$

Final linearization example

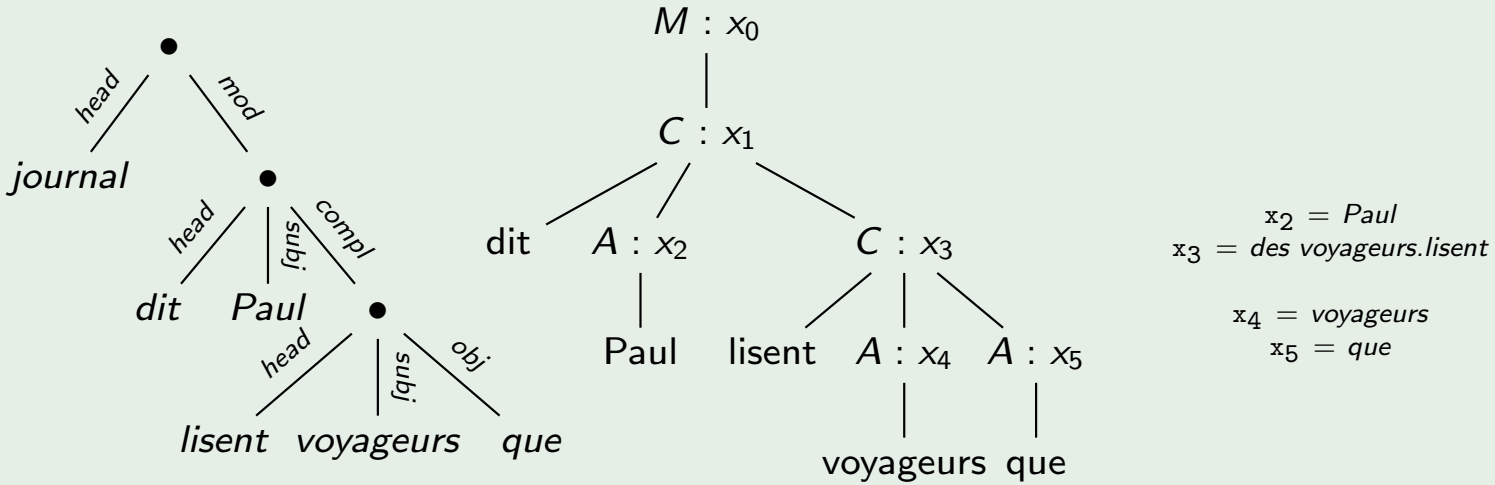


Adding complement clauses (2/2)

Updated logical definition of *wh_path* (Island constraints)

$$wh\_path(x, y) := subj(x, y) \vee compl^{*}\, obj(x, y)$$

Final linearization example



x_2 = *Paul*
x_3 = *des voyageurs.lisent*

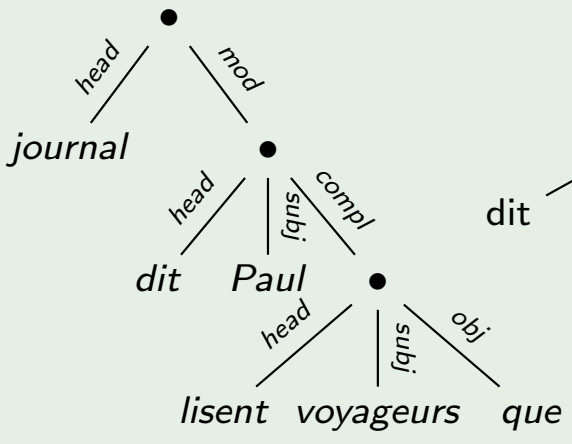
x_4 = *voyageurs*
x_5 = *que*

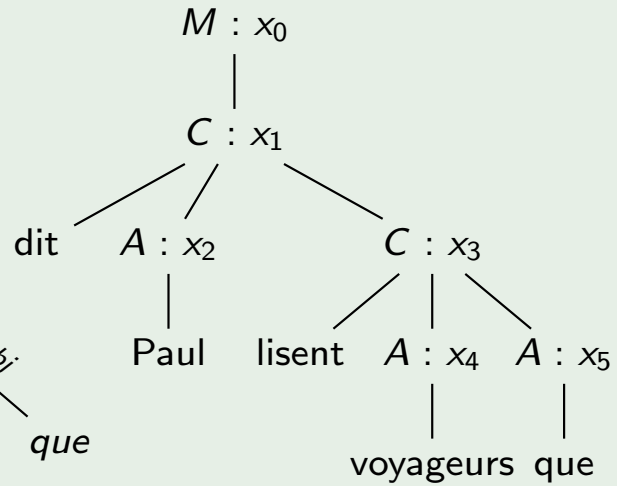
Adding complement clauses (2/2)

Updated logical definition of *wh_path* (Island constraints)

$$wh\_path(x, y) := subj(x, y) \vee compl^{*}\, obj(x, y)$$

Final linearization example





x_1 = *Paul.dit.que.des voyageurs lisent*

x_2 = *Paul*
x_3 = *des voyageurs.lisent*

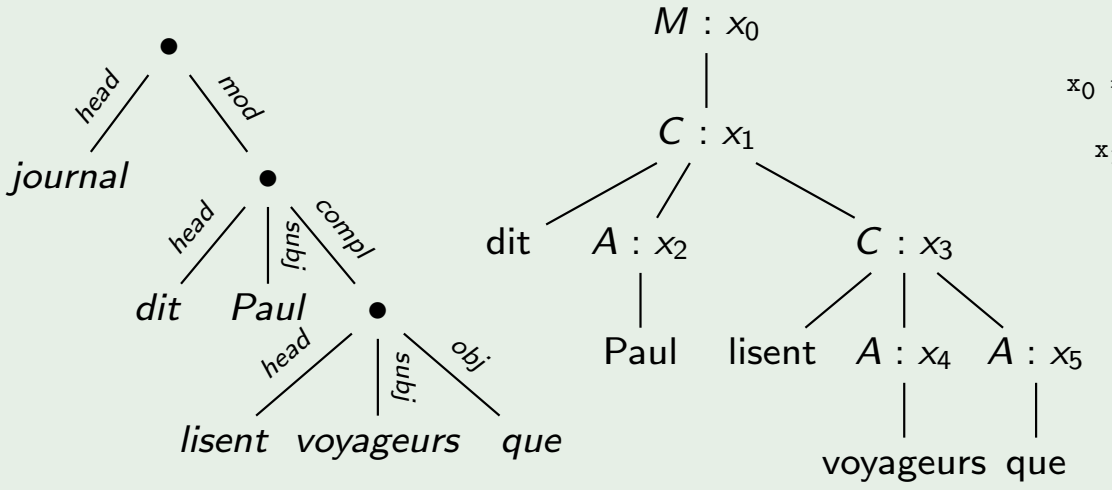
x_4 = *voyageurs*
x_5 = *que*

Adding complement clauses (2/2)

Updated logical definition of *wh_path* (Island constraints)

$$wh\_path(x, y) := subj(x, y) \vee compl^{*}\, obj(x, y)$$

Final linearization example



x_0 = *que.Paul dit que des voyageurs lisent*

x_1 = *Paul.dit.que.des voyageurs lisent*

x_2 = *Paul*
x_3 = *des voyageurs.lisent*

x_4 = *voyageurs*
x_5 = *que*

Conclusion

In summary, we have :

- Provided a logic-based toolset for linguistic description
- Demonstrated possible uses for it through a simple grammar
- Hinted at an effective procedure to compile the descriptions

Conclusion

In summary, we have :

- Provided a logic-based toolset for linguistic description
- Demonstrated possible uses for it through a simple grammar
- Hinted at an effective procedure to compile the descriptions

Possible directions :

- Work towards a usable implementation
- Improve mapping to language semantics
- Implement linguistic knowledge into larger grammars