Categorial minimalist grammars

Christian Retoré, Université Bordeaux 1

with Alain Lecomte U. Grenoble II and Maxime Amblard U. Bordeaux 1

GEOCAL Logic $\otimes$ Linguistics 15 février 2006

Equipe Signes
INRIA-Futurs & LaBRI-C.N.R.S.
& dép. sciences du langage, Université Bx3 Michel de Montaigne

Contents

1. General remarks

1.1. Syntax boundaries

- Inflectional morphology
  - for: depending on the language syntactic construction with explicit words OR inflection
  - against: different techniques (finite state automata, transducers — although in Navajo....)
- Logical semantics (who does what)
  - for: rather syntactic phenomena (logical syntax)
  - against: different techniques (e.g. in dependency approach formal grammars $\neq$ dependency graphs)
- prosody (which is related to syntactic structure)
- lexical semantics ($\rightarrow$ restricted selection)
- encyclopaedic knowledge ($\rightarrow$ getting rid off some ambiguities)

1.2. Linguistic theories and their mathematical models

Theories: generative grammar dependency grammar others?

Mathematical models:

- context-free grammars, tree grammars, composition different from substitution and term rewriting (e.g. adjunction in TAGs)
- unification grammars

Algorithmic complexity of parsing

- Unification grammars DCG GPSG HPSG (undecidable parsing)
- Context sensitive unification grammars like LFG (decidable parsing)
- TAGs, Range Concatenation Grammars (polynomial)

Given a theory, is there a privileged model?
(generative grammar $\longrightarrow$ TAGs?)
Given a model, is there and underlying theory
(cf. exegeses of HPSG by Pollard & Sag)

1.3. Modelling, Parsing, Generation

Parsing or generation?

- As far as analyse is concerned:
  - word order does not mind,
    sentences are grosso modo correct.
  - transformations and empty elements are a challenge
  - what do we do with parse structures?
- As far as generation is concerned
  - word order if crucial.
  - transformations and empty elements are welcome
  - out of what kind of object do we build (parse structure of) sentences.

1.4. Cognitive realism, empirical coverage

What do we model?

- corpora? normed language from some norm?
- examples representative of language faculty
  (internal language of X)
- pathological examples (like magma study for physics)

Problems

- linguistic resources
  (annotated corpora, grammars)
- are the solution to specific phenomena compatible
- surgeneration (never adressed in main stream NLP:
  corpora are a priori assumed to be correct)

1.5. Convergence vers une forme de grammaires catégorielles

Structure MLL
linear λ-calculus

$\swarrow$ $\searrow$

Phonological Form
(word order,...)

Logical Form
(logical structure)

- Pollard 2004: High-Order Categorical Grammar
- De Groote 2001: Abstract Categorial Grammars
- Muskens 2003: Lambdas, Language and Logic
- Lecomte Retoré 2001 Minimalist Categorial grammars
- Perrier 2001: Interaction Grammars

1.6. Generative grammars

Usual criticisms:

- transformations are algorithmically untractable (analyse / génération)
- derivation $\longrightarrow$ representations levels $\longrightarrow$ conditions on each
- what is Logical Form?

Awards:

- links between languages (principles and parameters)
- transformations: links between related sentences (questions / answers)
- syntax and semantics (coreference, (generalized) quantifier scopes)

1.7. Outcome of Stabler's formalisation of the minimalist program

- Good computability: polynomial, like LCFRS simple positive RCG)
- Derivational formalisation (generative-enumerative syntax)
  and representational formalisation (model-theoretic syntax)
  (cf. Pullum et Scholz 2001)

Mönnich, Morawietz et Michaelis (2001-2004):
Set of an MG parse trees =
image by a binary relation definable in monadic second order logic
of a set of regular tree definable in monadic second order logic
(hence can be analyzed with pushdown of pushdown automaton)

Un problème: difficulté d'écrire des grammaires lexicalisées.

2. Syntax and semantics in categorial grammars (reminder)]

2.1. Syntactic categories

$$\mathcal{B} = \{S, sn, n, ...\}$$

$$F ::= \mathcal{B} \mid F \setminus F \mid F \,/\, F$$

if $u : A$ and $f : A \setminus B$ then $uf : B$ (AB and Lambek)
if $u : A$ et $f : B \,/\, A$ then $fu : B$ (AB and Lambek)
if $u : A$ et $uf : B$ then $f : A \setminus B$ (Lambek only)
if $u : A$ et $fu : B$ then $f : B \,/\, A$ (Lambek only)

2.2. Semantic types

Church 1930, Curry 1940, Montague 1970

2.2.1. Logical formulae in simply typed λ-calculus with 2 basic types:

- individual e
- truth values t
- n-ary predicate : $e \rightarrow (e \rightarrow (e \rightarrow (\cdots \rightarrow t)))$
- n-ary function : $e \rightarrow (e \rightarrow (e \rightarrow (\cdots \rightarrow e)))$
- logical constants $\left| \begin{array}{rcl} \wedge, \vee, \Rightarrow & : & t \rightarrow (t \rightarrow t) \\ \exists, \forall & : & (e \rightarrow t) \rightarrow t \end{array} \right.$

2.2.2. Syntactic categories and semantic types

S^*	$=$	t	sentence: truth values / propositions
sn^*	$=$	e	individual
n^*	$=$	$e \rightarrow t$	unary predicate
$(A \setminus B)^* = (B \,/\, A)^*$	$=$	$A^* \rightarrow B^*$	propagation to every formula

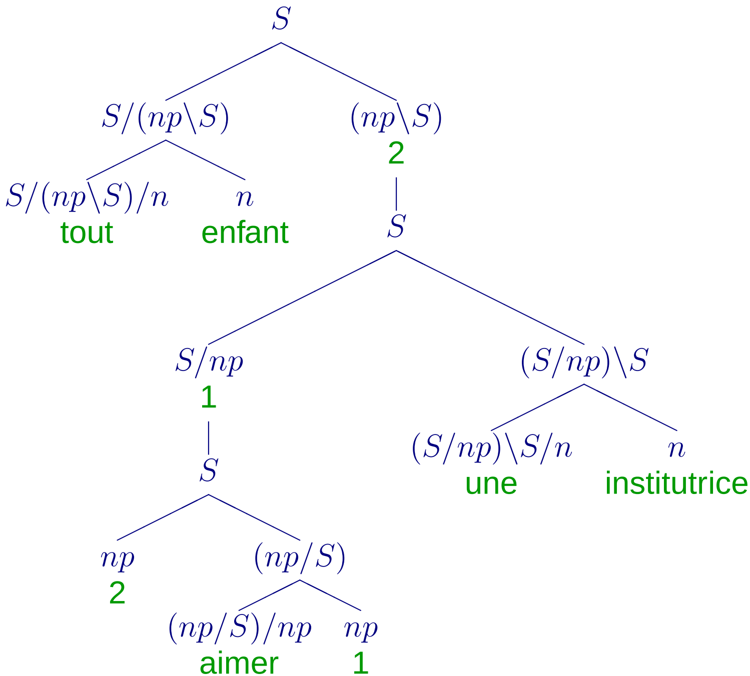
2.2.3. Lexicon: example

$aimer$	$(np\backslash S)/np$	$e \rightarrow e \rightarrow t$	$\lambda x \lambda y.aimer(y, x)$
$tout$	$((S/np)\backslash S)/n$	$(e \rightarrow t) \rightarrow (e \rightarrow t) \rightarrow t$	$\lambda P \lambda Q.\forall x P(x) \Rightarrow Q(x)$
$enfant$	n	$e \rightarrow t$	$\lambda x.enfant(x)$
une	$(S/(np\backslash S))/n$	$(e \rightarrow t) \rightarrow (e \rightarrow t) \rightarrow t$	$\lambda P \lambda Q.\exists x P(x) \wedge Q(x)$
$institutrice$	n	$e \rightarrow t$	$\lambda x.student(x)$

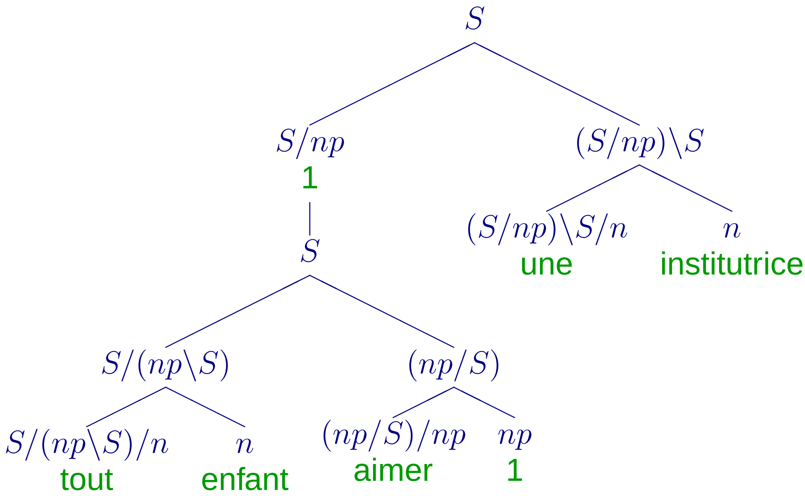
2.2.4. Parsing example:

Two syntactic analyses
for two possible readings.

1. (tout enfant)(λy (une institutrice) (λx aimer (x,y)))
   $\forall z\ enfant(z) \wedge (\exists s\ instit(x) \Rightarrow aimer(z,s))$
2. (une institutrice)(λx (tout enfant)(aimer x))
   $\exists s\ (instit(x) \Rightarrow \forall z enfant(z) \wedge aimer(z,s))$



S
$S/(np\backslash S)$
$(np\backslash S)$
2
$S/(np\backslash S)/n$
n
tout
enfant
S
S/np
1
$(S/np)\backslash S$
S
$(S/np)\backslash S/n$
n
une
institutrice
np
2
(np/S)
$(np/S)/np$
np
aimer
1


S
S/np
1
$(S/np)\backslash S$
S
$(S/np)\backslash S/n$
une
n
institutrice
$S/(np\backslash S)$
(np/S)
$S/(np\backslash S)/n$
tout
n
enfant
$(np/S)/np$
aimer
np
1

2.2.5. Explanation

Why does it works?

- syntactic analyse = proof in the Lambek calculus
- forgetting directions $\subset$ proof in MLL $\subset$ intuitionnistic logic
- type morphism $\longrightarrow$ intuitionistic proof, lambda-term
- variable := lexical lambda-terms (same type)
- beta redution $\longrightarrow$ proof of $S^* = t$ i.e. a proposition

2.2.6. Critics

- Too restricted syntactic formalism:

  discontinuous constituents: ne...pas

  middle extraction: Le livre que$_i$ [tu lis (t$_i$) ces jours-ci] est Samarcande

- some analyses do not have a semantic counter part

  (type raising is mandatory)

  e.g. Joan: $(e \rightarrow t) \rightarrow t$ and not e because of

  Joan et tous les invit:'es sont partis.

  an analysis with Joan: sn has no semantic counterpart

- the syntactic cateogry of the quantifiers depends on their syntactic position

  "tout" has a syntactic type for subject position

  another for object position, etc. pour celle objet, etc...

3. Stabler minimalist grammars

3.1. Overview

- based on the minimalist program
- lexicalised grammars
- generative capacity : MC-TAG / MCFG
- polynomial parsing
- principle and parameters approach to language variation

- relation bewteen related sentences
  (by movement and transformations):

  ⋆ questions
  (1)Combien de livres que Tabucchi a écrit aime-t-il?
  (2) Il aime trois livres que Tabucchi a écrit.

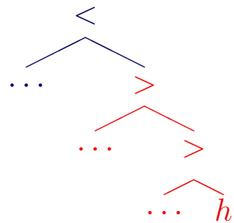
  ⋆ passif
  (3) Ce livre a été écrit par Pavese

  (4) Pavese a écrit ce livre.

- some question are raised:
  possible or impossible coreference
  il=Tabucchi (1) possible (2) impossible

3.2. Analysis structures

- binary trees
- leaves: list of features
- internal node : ”$<$” or ”$>$” leading to the head
- maximal projection of h: largest tree whose head is h



3.3. Lexicon

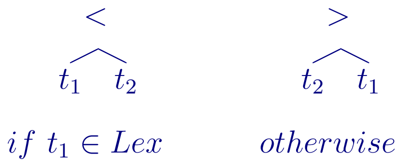
Features

- base d, n, v
- select =d for d in base
- licensees -case, -wh
- licensors +case, +CASE, +wh, +WH

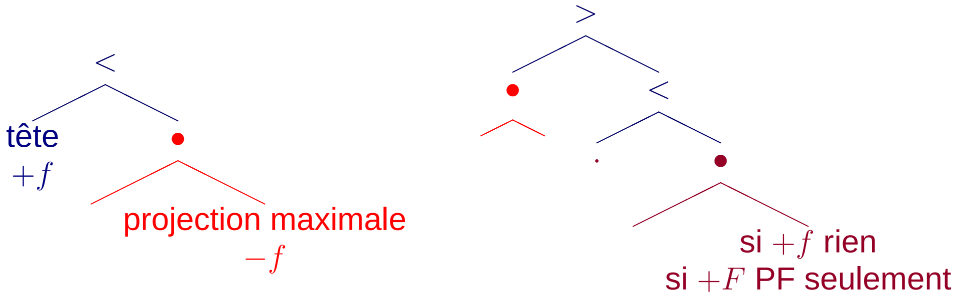
Lexicon : list of features /mot/ (mot)

3.4. Generative rules

MERGE



MOVE



3.5. Lexicon example

$aimer$	$=d\ +case\ =d\ v$
une	$=n\ d\ -case$
$institutrice$	n
$tout$	$=n\ d\ -case$
$enfant$	n
$infl$	$=v\ +case\ t$
$comp$	$=v\ c$

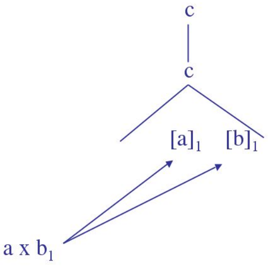
Exemple à faire au tableau

4. Categorial Minimalist Grammars

Only elimination rule

AB or Lambek grammars

commutative product



Merge

Move

[*Partially commutative linear logic*, de Groote, 1996]

4.1. Some differences:

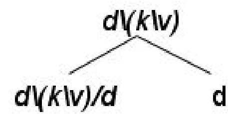
- internal subject hypothesis

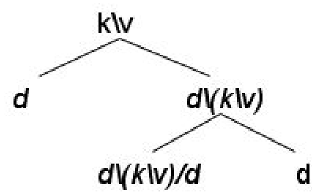
  (like in Radford 97 ands some other minimalist papers)
- commutative product

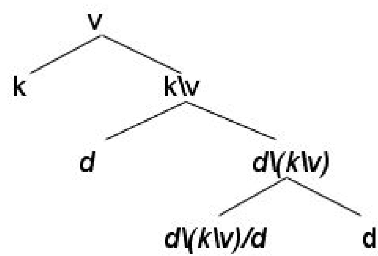
  set of features instead of list of features

4.2. Example of a categorial minimalist lexicon

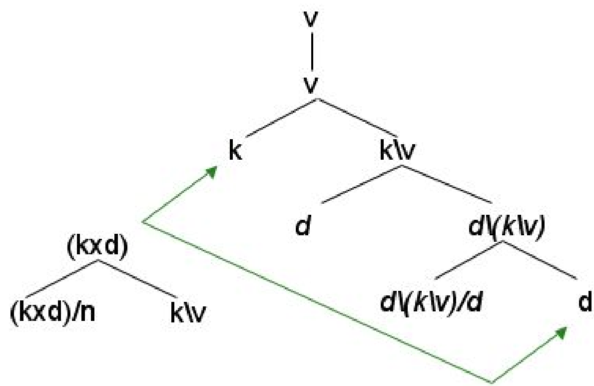
$aimer$	$(d\backslash k_1\backslash v)/d_1$	$=d\ +case\ =d\ v$
une	$k \times d/n$	$=n\ d\ -case$
$institutrice$	n	n
$tout$	$k \times d/n$	$=n\ d\ -case$
$enfant$	n	n
$infl$	$(k\backslash v)/v$	$=v\ +case\ t$
$comp$	v/c	$=v\ c$

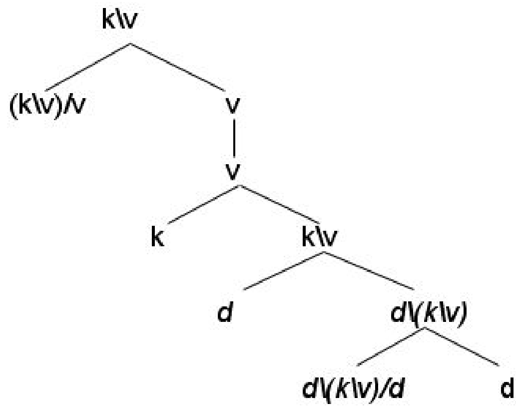

d\(k\v)
d\(k\v)/d
d

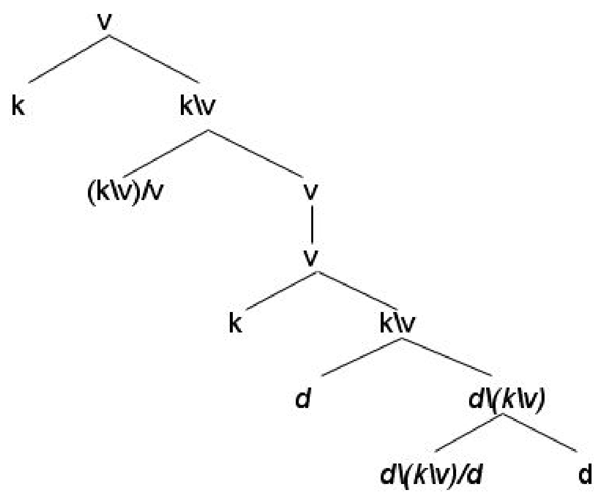

k\v
d
d\(k\v)
d\(k\v)/d
d



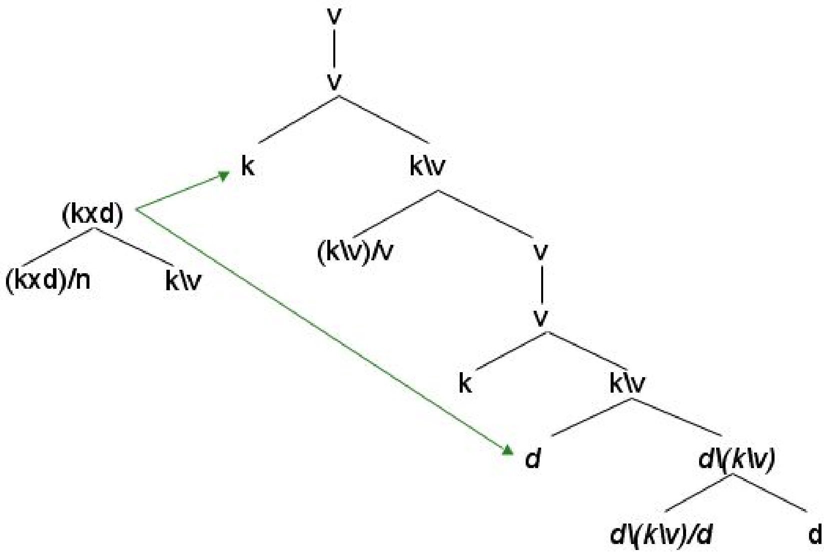
v
k
k\v
d
d\(k\v)
d\(k\v)/d
d

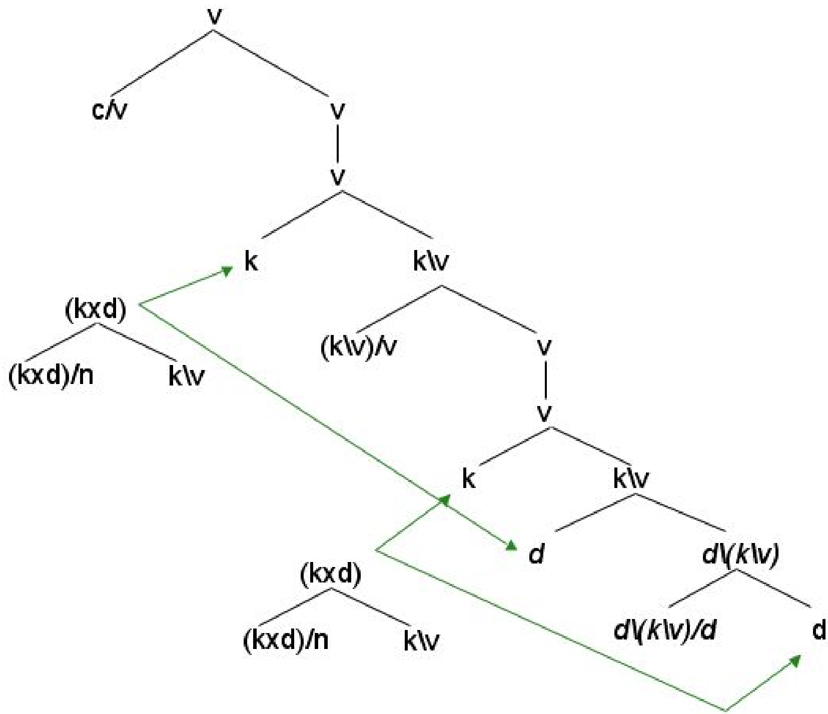

v
v
k
k\v
d
d\(k\v)
(kxd)
(kxd)/n
k\v
d\(k\v)/d
d


k\v
(k\v)/v
v
v
k
k\v
d
d\(k\v)
d\(k\v)/d
d



v
k
k\v
(k\v)/v
v
v
k
k\v
d
d\(k\v)
d\(k\v)/d
d


v
v
k
k\v
(kxd)
(k\v)/v
v
(kxd)/n
k\v
v
k
k\v
d
d\(k\v)
d\(k\v)/d
d


v
c/v
v
v
k
k\v
(kxd)
(k\v)/v
v
(kxd)/n
k\v
v
k
k\v
d
d\(k\v)
(kxd)
d\(k\v)/d
d
(kxd)/n
k\v

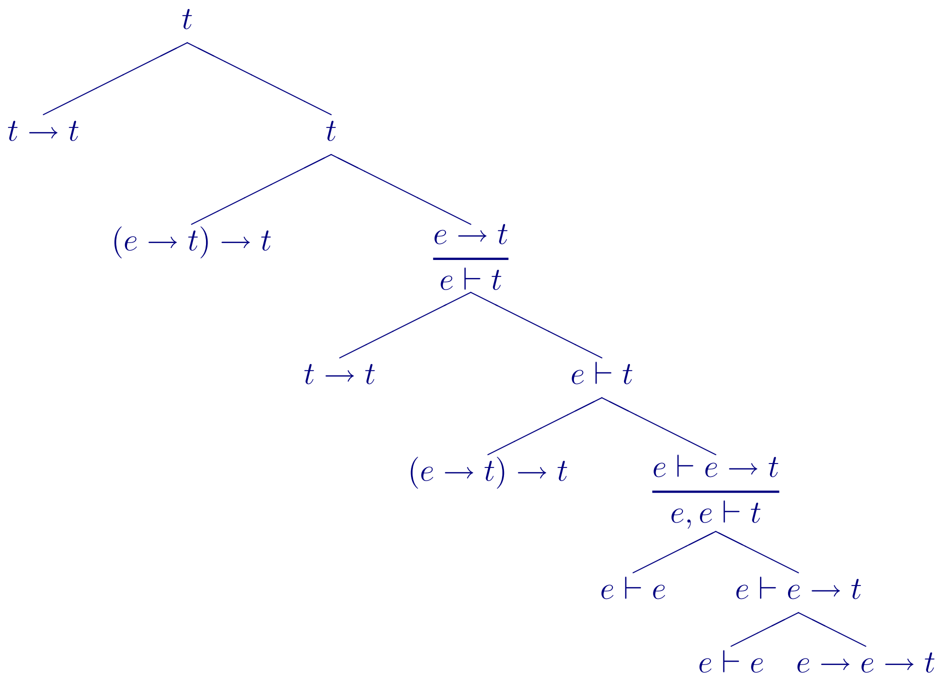
5. Syntax/semantics

5.1. Logical system for semantics

As usual:

$\star$logical formulae as λ-terms.
Base types e and t à la Montague.

BUT moreover
$\star\lambda$-terms with explicit contexts:
list of free variables



t
$t \longrightarrow t$
t
$(e \longrightarrow t) \longrightarrow t$
$\dfrac{e \longrightarrow t}{e \vdash t}$
$t \longrightarrow t$
$e \vdash t$
$(e \longrightarrow t) \longrightarrow t$
$\dfrac{e \vdash e \longrightarrow t}{e, e \vdash t}$
$e \vdash e$
$e \vdash e \longrightarrow t$
$e \vdash e$
$e \longrightarrow e \longrightarrow t$

5.2. Semantic rules

- application : $[\rightarrow]$
- abstraction in the tree hosting the move

$$\frac{\Gamma, z : Z \vdash u : U}{\Gamma \vdash (\lambda z.\ u) : Z \rightarrow U[EXTRACT]}$$

- application, for type raising

$$\frac{\Delta \vdash z : (T \rightarrow U) \rightarrow V \quad \Gamma \cup [x : T] \vdash u : U}{\Delta \cup \Gamma \vdash z(\lambda x.u) : V} [RAISE]$$

- l'application, sans montée de type

$$\frac{\Delta \vdash z : T \quad \Gamma, x : T \vdash u : U}{\Delta \cup \Gamma \vdash (\lambda x.\ u)z : U} [NORAISE]$$

5.3. Syntax/semantics

SYN, syntactic calculus

- connectives: $\times$, $/$, $\backslash$
- only elimination rules (encoding move and merge

SEM, semantic calculus

- connective $\longrightarrow$
- semantic rules (derived rules)

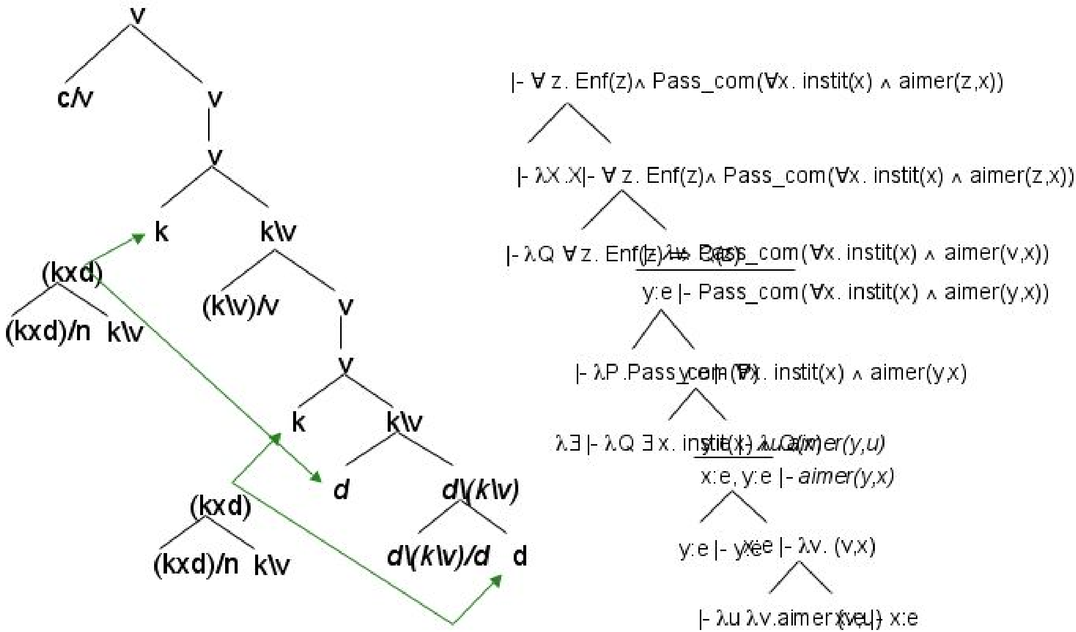
parallel $SYN \parallel SEM$:

$$\begin{array}{ll} syntaxe & semantique \\ & \\ merge : [/][\backslash] & [\rightarrow] \\ move & [Extract] \\ projection & [RAISE] ou [NORAISE] \end{array}$$

- every leaf in SEM has a coindexed part in SYM
- each step and its counterpart are executed
  in the same order in their respective derivations

5.4. Sample lexicon

$aimer$	$=d\ {+case}\ =d\ v$	$k\backslash d\backslash v/d$	$\vdash \lambda x \lambda y.aimer(y,x)$
une	$=n\ d\ {-case}$	$k \times d/n$	$\vdash \lambda P \lambda Q.\exists x P(x) \wedge Q(x)$
$institutrice$	n	n	$\vdash \lambda x.instit(x)$
$tout$	$=n\ d\ {-case}$	$k \times d/n$	$\vdash \lambda P \lambda Q.\forall x P(x) \Rightarrow Q(x)$
$enfant$	n	n	$\vdash \lambda x.enf(x)$
$infl$	$=v\ {+case}\ t$	$(t/k)/v$	$\vdash \lambda P.pass\_comp(P)$
$comp$	$=v\ c$	v/c	$\vdash \lambda P.P$


v
c/v
v
v
k
k\v
(kxd)
(kxd)/n
k\v
(k\v)/v
v
v
k
k\v
d
d\(k\v)
(kxd)
(kxd)/n
k\v
d\(k\v)/d
d
|- ∀ z. Enf(z) ∧ Pass_com(∀x. instit(x) ∧ aimer(z,x))
|- λX .X|- ∀ z. Enf(z) ∧ Pass_com(∀x. instit(x) ∧ aimer(z,x))
y:e |- Pass_com(∀x. instit(x) ∧ aimer(y,x))
x:e, y:e |- aimer(y,x)

6. Results

- syntax/semantics correspondence
  extended to a richer syntactic system.

- a single syntactic category for a quantifier,

  whatever might be its syntactic position.

- understanding movement
  - in the structure which host the moved constituent: λ-abstraction
  - for the moved constituent : type raising

7. In progress: CMG proofnets

Minimalist grammars without movement : bounded pushdown for partial structures, insertion only when there will be no further movement.

Word order can be reconstructed without distinct $\backslash$ and $/$

- first application of a lexical function : argument after fucntion (lexical merge)
- otherwise argument before the function (non lexical merge)

Proof-nets (graphs):

- equivalent formalism
- better for product
  (complicated normal forms)
- avoid co-indexation of hypothesis to be cancelled simultaneously
- better algorithms for constructing analysis
  (e.g. minimizing axioms length, Moot 2004
- formulae $\longrightarrow$ trees taking into account the order of the operations

8. Perspectives

- possible or impossible coreference for anaphora resolution
  incremental calculus of binding principles and small clauses
  or of Reinhardt/Reuland semantic binding (Bonato)
  (1) Carlotta's dog thinks that he hates him.
  (2) * Il$_i$ aime trois livre que Tabuchi $_i$ a écrit.

- semantics of questions
  (Maxime Amblard)
  (3) Quel train Pierre prend?
  (4) Quel train prend Pierre? (plus difficile)

- clitics, clitic climbing with correct control intepretation (in progress, Amblard)

  (similar to Moot/Retoré 2005 for multimodal categorial grammars or to Stabler 2001)

  (5) Je répare ma voiture.

  (6) Je la répare.

  (7) Je sais la réparer.

  (8) Je la fais réparer. (”la” is being repared)

  (9) Je te permets de venir. (”te” viens)

  (10) Je te promets de venir (”je” viens)