Some pragmatic and semantic applications of the system F

Christian Retoré

LaBRI-C.N.R.S. & Université de Bordeaux

with the help of M. Abrusci, R. Moot and many others

S.-J. Conrad, N. Asher, Z. Luo,

C. Beyssade, F. Corblin, A. Lecomte, D. Nicolas, C. Pollard, ...

1. A non-standard point of view

As opposed to main stream "formal semantics" in linguistics this talk neither deal with truth nor with reference:

- too difficult (vagueness)
- other interpretations (i.e. interaction, proofs and refutations) seem to be more adequate

We are simply thinking about the

logical syntax of human language semantics

which logical language would be a not-so-bad approximation of "meaning", and how do we compute semantic representation in the language from sentences or discourses.

2. Syntax of semantics, logical syntax

Montague's view:

syntax $\longrightarrow$ (logical form) $\longrightarrow$ **truth, reference in possible worlds**

our view:

syntax $\longrightarrow$ **logical form** $\longrightarrow$ interaction models, proofs and refutations (later on)

this talk

3. A necessary step before discourse and dialogue study

(1) You live in Marseilles?

(2) Yes, place Jean Jaurès.

(3) How did you come to CIRM?

(4) By bike.

(5) I did not know you like that much sports.
Any snow downtown?

To know what answers what, you need to know:

- who does what
- the lexical relations (sports / bike)
- some world knowledge (Luminy is over a hill) — is it lexical or not?

Part I

The usual framework: Montague semantics

4. An improper use of the name "Montague"

In what follows:

- no intentionality operators....
- no set theoretic semantics...
- no possible worlds...

which are usually associated with "Montague".

We only keep Montague's work on "compositionality" that is the **computation of a logical formula** from

- a syntactic analysis
- the lexicon — mapping words to partial formulae

5. Back to the roots: Montague semantics. Types.

Simply typed lambda terms

$$types ::= e \mid t \mid types \rightarrow types$$

chair , *sleep* $e \rightarrow t$
likes transitive verb $e \rightarrow (e \rightarrow t)$

6. Back to the roots: Montague semantics. Syntax/semantics.

$(\text{Syntactic type})^*$	$=$	Semantic type	
S^*	$=$	t	a sentence is a proposition
np^*	$=$	e	a noun phrase is an entity
n^*	$=$	$e \rightarrow t$	a noun is a subset of the set of entities
$(A \setminus B)^* = (B / A)^*$	$=$	$A \rightarrow B$	extends easily to all syntactic categories of a Categorial Grammar e.g. a Lambek CG

7. Back to the roots: Montague semantics. Logic within lambda-calculus 1/2.

Logical operations (and, or, some, all the,.....) need constants:

Constant	Type
$\exists$	$(e \rightarrow t) \rightarrow t$
$\forall$	$(e \rightarrow t) \rightarrow t$
$\wedge$	$t \rightarrow (t \rightarrow t)$
$\vee$	$t \rightarrow (t \rightarrow t)$
$\supset$	$t \rightarrow (t \rightarrow t)$

8. Back to the roots: Montague semantics. Logic within lambda-calculus 2/2.

Words in the lexicon need constants for their denotation:

likes	$\lambda x \lambda y\ (\text{likes}\ y)\ x$	$x : e,\ y : e, \text{likes} : e \rightarrow (e \rightarrow t)$
« likes » is a two-place predicate		

Garance	$\lambda P\ (P\ \text{Garance})$	$P : e \rightarrow t,\ \text{Garance} : e$
« Garance » is viewed as the properties that « Garance » holds		

9. Back to the roots: Montague semantics. Computing the semantics. 1/5

1. Replace in the lambda-term issued from the syntax the words by the corresponding term of the lexicon.
2. Reduce the resulting λ-term of type t its normal form corresponds to a formula, the "meaning".

10. Back to the roots: Montague semantics. Computing the semantics. 2/5

word	***semantic type*** u^* ***semantics :*** λ***-term of type*** u^* x_v *the variable or constant* x *is of type* v
some	$(e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)$ $\lambda P_{e\rightarrow t}\, \lambda Q_{e\rightarrow t}\, (\exists_{(e\rightarrow t)\rightarrow t}\, (\lambda x_e(\wedge_{t\rightarrow(t\rightarrow t)}(P\ x)(Q\ x)))$
statements	$e \rightarrow t$ $\lambda x_e(\texttt{statement}_{e\rightarrow t}\ x)$
speak_about	$e \rightarrow (e \rightarrow t)$ $\lambda y_e\, \lambda x_e\, ((\texttt{speak\_about}_{e\rightarrow(e\rightarrow t)}\ x)y)$
themselves	$(e \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t)$ $\lambda P_{e\rightarrow(e\rightarrow t)}\, \lambda x_e\, ((P\ x)x)$

11. Back to the roots: Montague semantics. Computing the semantics. 3/5

The syntax (e.g. a Lambek categorial grammar) yields a λ-term representing this deduction simply is

((some statements) (themsleves speak_about)) of type t

12. Back to the roots: Montague semantics. Computing the semantics. 4/5

$$\Big(\big(\lambda P_{e\to t}\ \lambda Q_{e\to t}\ (\exists_{(e\to t)\to t}\ (\lambda x_e(\wedge(P\ x)(Q\ x)))))\big)$$
$$\big(\lambda x_e(\texttt{statement}_{e\to t}\ x)\big)\Big)$$
$$\Big(\big(\lambda P_{e\to(e\to t)}\ \lambda x_e\ ((P\ x)x)\big)$$
$$\big(\lambda y_e\ \lambda x_e\ ((\texttt{speak\_about}_{e\to(e\to t)}\ x)y)\big)\Big)$$

$$\downarrow \beta$$

$$\big(\lambda Q_{e\to t}\ (\exists_{(e\to t)\to t}\ (\lambda x_e(\wedge_{t\to(t\to t)}(\texttt{statement}_{e\to t}\ x)(Q\ x)))))$$
$$\big(\lambda x_e\ ((\texttt{speak\_about}_{e\to(e\to t)}\ x)x)\big)$$

$$\downarrow \beta$$

$$\big(\exists_{(e\to t)\to t}\ (\lambda x_e(\wedge(\texttt{statement}_{e\to t}\ x)((\texttt{speak\_about}_{e\to(e\to t)}\ x)x)))\big)$$

13. Back to the roots: Montague semantics. Computing the semantics. 5/5

This term represent the following formula of predicate calculus (in a more pleasant format):

$$\exists x : e \, (\texttt{statement}(x) \wedge \texttt{speak\_about}(x, x))$$

This is a (simplistic) semantic representation of the analyzed sentence.

14. Two levels

Logic/calculus for meaning assembly (a.k.a glue logic, metalogic,...) In the standard case: simply typed lambda calculus with two base types.

Logic/language for semantic representations In the standard case: higher-order predicate logic or first-order logic.

This Churchian view (λ-terms as partial formulae) introduced λ-calculus to handle substitution in Hilbert style deduction.

For Curryhowardists (typed λ-terms as proofs) what is the proof a formula viewed as a lambda term of type $\mathbf{t}$: it is a proof of its correctness, that's all.

15. This is too simplistic:

Lexical semantics is out of reach:

- a common noun say "book" (syntax) maps to a unary predicate "book:$\mathbf{e} \to \mathbf{t}$"
- a transitive verb like "read" maps to a binary predicate "read:$\mathbf{e} \to \mathbf{e} \to \mathbf{t}$"
- this will only give you the argument structure of *Mary reads a book.* ($\exists x : \mathbf{e} \, book(x)$ *and* $reads(Mary, x)$)
- but it cannot relate *book* and *read* as any dictionary does.

It says nothing about classical questions: plurals, generalised quantifiers,...

only retain the computational view!

16. Refinements

Much more types in the calculus for meaning assembly....
for filtering out illicit compositions: *Their ten is incredibly fast*.

Complex terms in the lexicon (triggered by context) for overpassing the aforementioned filter e.g. *ten* may be a soccer player.

Thereafter, uniform operations on types like in system F will be welcome.

17. Why types in the syntax of semantics

opposed to Frege's single sort view:
$\forall x : A\ P(x) \Longleftrightarrow \forall x.\ A(x) \rightarrow P(x)$
(impossible for "the majority of", "most of" etc.)

in ancient and especially medieval philosophy (in particular Abu-l Barakāt al Baghdādī, Avicenna):
we assert properties of things as being member of some class (= type?)

There are less types than logical formulae with a single free variable, they are more constrained, and not any formula defines a comparison class.

One should be cautious: the coexistence of *types* and of *usual formulae* opens the *gates of hell*.

18. A personal view on the border between semantics and pragmatics

- semantics is encoded by the terms:
  they yield formulae by compositionality
- pragmatics is encoded in the types
  they are flexible and determined by the context

19. Terms and types, semantics and pragmatics

A paradox of Frege's view. For the very same Carlotta (two-year old) on can both have

(6) Carlotta is tall. (class: two-year old)

(7) Carlotta is not tall. (class: human beings)

Types should be used to model classes, sorts,... and they are inferred from the context (pragmatic).
With such a view we go for types depending on the context (not on other terms as in dependent types).

Terms (with standard typing) express the possible composition for computing semantic representations. The lexicon tells which transformations are possible (meaning transfers, adaptation of the class).

Part II

Extending the type system

20. More general types and terms. Many sorted logic. TY_n

Extension to TY_n without difficulty nor surprise:
e can be divided in several kind of entities.

It's a kind of flat ontology:
objects, concepts, events,...
multisorted higher-order logic

... but this yields puzzling question: types there could be predicates acting on larger types.

21. More general types and terms. Second order types (Girard's F).

One can also add type variables and quantification over types.

- Constants e and t, as well as any type variable α in P, are types.
- Whenever T is a type and α a type variable which may but need not occur in T, $\Lambda\alpha.\ T$ is a type.
- Whenever T_1 and T_2 are types, $T_1 \to T_2$ is also a type.

22. More general types and terms. Second order terms (Girard's F).

- A variable of type T i.e. $x : T$ or x^T is a *term*. Countably many variables of each type.
- $(f\ \tau)$ is a term of type U whenever $\tau : T$ and $f : T \to U$.
- $\lambda x^T.\ \tau$ is a term of type $T \to U$ whenever $x : T$, and $\tau : U$.
- $\tau\{U\}$ is a term of type $T[U/\alpha]$ whenever $\tau : \Lambda\alpha.\ T$, and U is a type.
- $\Lambda\alpha.\tau$ is a term of type $\Lambda\alpha.T$ whenever α is a type variable, and $\tau : T$ without any free occurrence of the type variable α.

23. More general types and terms. Second order reduction.

The reduction is defined as follows:

- $(\Lambda\alpha.\tau)\{U\}$ reduces to $\tau[U/\alpha]$ (remember that α and U are types).
- $(\lambda x.\tau)u$ reduces to $\tau[u/x]$ (usual reduction).

Reduction is strongly normalising and confluent (Girard, 1971): *every term of every type admits a unique normal form which is reached no matter how one proceeds.*

24. More general types and terms. A second order example.

Given two predicates $P^{\alpha\to t}$ and $Q^{\beta\to t}$
over entities of respective kinds α and β
two morphisms f from ξ to α and g from ξ to β
we can coordinate entities of type ξ:
$\Lambda\xi\lambda x^{\xi}\lambda f^{\xi\to a}\lambda g^{\xi\to b}.$
$$(and\ (P\ (f\ x))(Q\ (g\ x)))$$

Universal closure w.r.t. morphisms, properties, types:
$\Lambda\alpha\Lambda\beta\lambda P^{\alpha\to t}\lambda Q^{\beta\to t}\Lambda\xi\lambda x^{\xi}\lambda f^{\xi\to\alpha}\lambda g^{\xi\to\beta}.$
$$(\mathrm{and}\ (P\ (f\ x))(Q\ (g\ x)))$$
The functions are not really functions: (fx) should be read as x of type ξ viewed as an α object.

Part III

Integrating facets in a compositional lexicon

25. Principles of our lexicon

- Remain within realm of Montagovian compositional semantics (but no models).
- Allow both predicate and argument to contribute lexical information to the compound.
- Integrate within existing discourse models (λ-DRT).

We advocate a system based on *optional modifiers*.

26. The Types

- Montagovian composition:
  - Predicate include the typing and the order of its arguments.
- Generative Lexicon style concept hierarchy:
  - Types are different for every distinct lexical behavior
  - A kind of ontology details the specialization relations between types

Second-order typing, like Girard's F system is needed for arbitrary modifiers:

$$\Lambda\alpha\lambda x^{A} y^{\alpha} f^{\alpha\rightarrow R}.((\mathsf{read}^{A\rightarrow R\rightarrow t}\ x)\ (f\ y))$$

27. The Terms: main / standard term

- A standard λ-term attached to the main sense:
  - Used for compositional purposes
  - Comprising detailed typing information
  - Including slots for optional modifiers
  - e.g. $\Lambda\alpha\beta\lambda x^{\alpha} y^{\beta} f^{\alpha\to A} g^{\beta\to F}.((\mathsf{eat}^{A\to F\to t}\ (f\ x))\ (g\ y))$
  - e.g. Paris^{T}

28. The Terms: Optional Morphisms

- Each a one-place predicate
- Used, or not, for adaptation purposes
- Each associated with a constraint : *rigid,* $\varnothing$

$$* \left(\frac{Id^{F \to F}}{\varnothing}, \frac{f_{grind}^{Living \to F}}{rigid} \right)$$

$$* \left(\frac{Id^{T \to T}}{\varnothing}, \frac{f_{L}^{T \to L}}{\varnothing}, \frac{f_{P}^{T \to P}}{\varnothing}, \frac{f_{G}^{T \to G}}{rigid} \right)$$

29. A Complete Lexical Entry

Every lexeme is associated to an n-uple such as:

$$\left(\text{Paris}^{T}, \frac{\lambda x^{T}.\ x^{T}}{\varnothing}, \frac{\lambda x^{T}.(f_{L}^{T\to L}\ x)}{\varnothing}, \frac{\lambda x^{T}.(f_{P}^{T\to P}\ x)}{\varnothing}, \frac{\lambda x^{T}.(f_{G}^{T\to G}\ x)}{\mathit{rigid}}\right)$$

30. RIGID vs flexible use of optional morphisms

Type clash: $(\lambda x^V.\ (P^{V\to W} x))\tau^U$

$$(\lambda x^V.\ (P^{V\to W} x))\ (f^{U\to V}\tau^U)$$

f: optional term associated with either P or τ
f **applies once to the argument** and not to the several occurrences of x in the function.
A conjunction yields
$(\lambda x^V.\ (\wedge\ (P^{V\to W} x)\ (Q^{V\to W} x))\ (f^{U\to V}\tau^U)$,
the argument is uniformly transformed.
Second order is not needed, the type V of the argument is known and it is always the same for every occurrence of x.

31. FLEXIBLE vs. rigid use of optional morphisms

$(\lambda x^{?}.\ (\cdots(P^{A\to X}x^{?})\cdots(Q^{B\to Y}x^{?})\cdots)\tau^{U}$:
type clash(es) [Montague: $?=A=B$ e.g. e]
$(\Lambda\xi.\lambda f^{\xi\to A}.\lambda g^{\xi\to B}.\ (\cdots(P^{A\to X}(fx^{\xi}))\cdots(Q^{B\to Y}(gx^{\xi}))\cdots))$
$\{U\}\ f^{U\to A}\ g^{U\to B}\ \tau^{U}$
f, g: optional terms associated with either P or τ.
For each occurrence of x
with different $A, B, \ldots$ with different $f, g, \ldots$ each time.

Second order typing:
1) anticipates the yet unknown type of the argument
2) factorizes the different function types in the slots.

The types $\{U\}$ and the associated morphism f are inferred from the original formula $(\lambda x^{V}.\ (P^{V\to W}x))\tau^{U}$.

32. Standard behaviour

ϕ: physical objects

small stone

$$\overbrace{(\lambda x^{\varphi}.\ (\text{small}^{\varphi\to\varphi}\, x))}^{\text{small}}\ \overbrace{\tau^{\varphi}}^{\text{stone}}$$

$$(\text{small}\ \tau)^{\varphi}$$

33. Qualia exploitation

wondering, loving smile

$$\overbrace{(\lambda x^{P}.\ (\text{and}^{t\to(t\to t)}\ (\text{wondering}^{P\to t}\ x)\ (\text{loving}^{P\to t}\ x)))}^{\text{wondering, loving}}\ \overbrace{\tau^{S}}^{\text{smile}}$$

$$(\lambda x^{P}.\ (\text{and}^{t\to(t\to t)}\ (\text{wondering}^{P\to t}\ x)\ (\text{loving}^{P\to t}\ x)))(f_a^{S\to P}\tau^{S})$$

$$(\text{and}\ (\text{loving}\ (f_a\ \tau))\ (\text{loving}\ (f_a\ \tau)))$$

34. Facets (dot-objects): incorrect copredication

Incorrect co-predication. The rigid constraint blocks the copredication e.g. $f_g^{Fs \to Fd}$ cannot be **rigidly** used in

(8) ?? The tuna we had yesterday was lightning fast and delicious.

(9) ?? Liverpool defeated MU and is and important harbour.

35. Facets, correct co-predication. Town example 1/3

T town L location P people
$f_p^{T \to P}$ $f_l^{T \to L}$ k^T København

København is both a seaport and a cosmopolitan capital.

36. Facets, correct co-predication. Town example 2/3

Conjunction of $cospl^{P\to t}$, $\mathrm{cap}^{T\to t}$ and $port^{L\to t}$, on k^{T}
If $T = P = L = e$, (as in Montague)
$(\lambda x^{e}(\mathrm{and}^{t\to(t\to t)}((\mathrm{and}^{t\to(t\to t)}\ (\mathrm{cospl}\ x)\ (\mathrm{cap}\ x))\ (\mathrm{port}\ x)))\ k.$
Here is the λ-term for *AND* between three predicates over different kinds $P^{\alpha\to t}$, $Q^{\beta\to t}$, $R^{\beta\to t}$

$$\Lambda\alpha\Lambda\beta\Lambda\gamma\ \lambda P^{\alpha\to t}\lambda Q^{\beta\to t}\lambda R^{\gamma\to t}\ \Lambda\xi\lambda x^{\xi}\ \lambda f^{\xi\to\alpha}\lambda g^{\xi\to\beta}\lambda h^{\xi\to\gamma}.\ (\mathrm{and}(\mathrm{and}\ (P\ (f\ x))(Q\ (g\ x)))(R\ (h\ x)))$$

f, g and h convert x to **different** types.

37. Facets, correct co-predication. Town example 3/3

AND applied to P and T and L and to $cospl^{P\to t}$ and $cap^{T\to t}$ and $\mathsf{port}^{L\to t}$ yields:

$\Lambda\xi\,\lambda x^{\xi}\,\lambda f^{\xi\to\alpha}\lambda g^{\xi\to\beta}\lambda h^{\xi\to\gamma}.$
$(\mathsf{and}(\mathsf{and}\,(cospl^{P\to t}\,(f_p\,x))(cap^{T\to t}\,(f_t\,x)))(\mathsf{port}^{L\to t}\,(f_l\,x)))$

We now wish to apply this to the type T and to the transformations provided by the lexicon. No type clash with $cap^{T\to t}$, hence $id^{T\to T}$ works. For L and P we use the transformations f_p and f_l.

$(\mathsf{and}^{t\to(t\to t)}$
$\quad(\mathsf{and}^{t\to(t\to t)}$
$\quad\quad(\mathsf{cospl}(f_p\,k^T)^P)^t)(\mathsf{cap}(\mathsf{id}\,k^T)^T)^t)^t(\mathsf{port}\,(f_l\,k^T)^L)^t)^t$

38. Fictive motion, virtual traveller

Corpus of travels in the Pyrenees (French, XVII-XX centuries)...

(10) nous descendons, pendant un quart d'heure, la vallée de l'Esera. *we descend, for a quarter of an hour, the Esera valley.*

(11) La lune, qui éclaire notre marche, nous fait découvrir sur la droite un sentier qui serpente. *The moon, which lightens our steps, allows us to discover a winding path on our right.*

(12) Il nous conduit sur un petit plateau, au milieu de sapins, au-dessus et à quelque distance du torrent de Ramun. *It leads us to a small plateau, surrounded by firs, at some distance of and above the Ramun torrent*.

"Il" (it) in sentence 12 refers to "un sentier qui serpente" imposes anaphora resolution before coercion → constraints on the possible interpretations *Background*(10,11) and *Narration*(11,12).

39. More examples...

Rhetoric: important & hard to infer. Virtual? Wait for 16!

(13) Nous partimes pour Barèges à 8 heures du matin par une fort jolie route qui nous conduisit à Lourdes.
We left (PS) for Barèges at 8 in the morning, taking a very pretty road which led (PS) us to Lourdes.

(14) (...) qui va en se resserrant jusqu'à Pierrefite, où les routes de Lux et de Cauterets séparent.
(...) which goes shrinking along the way, up to Pierrefite, where the roads to Lux and to Cauterets split.

(15) Celle de Lux entre dans une gorge qui vous mène au fond d'un précipice et traverse le gave de Pau.
The one to Lux enters a gorge which leads you to the bottom of a precipice and traverses the Gave de Pau.

(16) (...) Après une longue marche, l'on arrive à Barèges à 6 heures du soir. *After a long walk, we arrive in Barèges at 6 in the evening.*

40. A simple example

To sum up, we often see examples like:

(17) The road goes up for two hours.

And sometimes, no one follows the path....they simply speak about it while passing by.
Here as well, a type conflict triggers some lexical resources:

$$\left(P^{hu \rightarrow \mathbf{t}}\left(u^{path}\right)\right) \qquad hu(man) \neq path$$

The lexicon as a dictionary tells us that this construct is allowed and means someone following the path moves upwards for two hours.

Without the duration,no need of a virtual traveller following the path: the "path" could be a parametrized path whose third component increases.
Hence the u^{path} produces a x^{human}, and if u was an argument of P but of type *human*:

1. The scope of the human could not encompass P.
2. Properties or the path like *tarred*, would become properties of the traveller (cf. Lucky Luke)

Hence transformation apple not to the type *path* but to the type-raised version $(path \rightarrow \mathbf{t}) \rightarrow \mathbf{t}$ and yields a type-raised human $(hu \rightarrow \mathbf{t}) \rightarrow \mathbf{t}$ which can have the wider scope. Here $\mathbf{t}$ should be $\mathbf{t} \rightarrow \mathbf{v}$ because we have event variables.

$p : path$ — picked up by a function finding an element from a formula.

kind of coercion

$$h = \lambda Q^{(path \to \mathbf{v} \to \mathbf{t}) \to \mathbf{v} \to \mathbf{t}} \lambda P^{hu \to \mathbf{v} \to \mathbf{t}} \\ (Q\ (\lambda c^{path} \lambda e^{\mathbf{v}}\ \forall (\lambda v^{hu}\ suivre(e, v, c) \Rightarrow ((P\ v)\ e))))$$

$$(h\ p) = ((\lambda Q^{(path \to \mathbf{v} \to \mathbf{t}) \to \mathbf{v} \to \mathbf{t}} \lambda P^{hu \to \mathbf{v} \to \mathbf{t}} \\ (Q\ (\lambda c^{path} \lambda e^{\mathbf{v}}\ \forall (\lambda v^{hu}\ suivre(e, v, c) \Rightarrow ((P\ v)\ e)))))) \\ (\lambda P^{path \to \mathbf{v} \to \mathbf{t}} \lambda e^{\mathbf{v}} (P\ p^{path}\ e)))$$

$$=_{\beta} \lambda P^{hu \to \mathbf{v} \to \mathbf{t}} \lambda e^{\mathbf{v}}\ \forall (\lambda y^{hu} suivre(e, y, p^{path}) \Rightarrow ((P\ x)\ e))$$

$$goes\_up = \lambda x^{hu} \lambda e^{\mathbf{v}} up(e, x)$$
$$((h(the\ path))goes\_up) =$$
$$((\lambda P^{hu \to \mathbf{v} \to \mathbf{t}} \lambda e^{\mathbf{v}}\ \forall(\lambda y^{hu} suivre(e, y, p^{path}) \Rightarrow ((P\ x)\ e)))$$
$$(\lambda x^{hu} \lambda e^{\mathbf{v}} up(e, x)))$$
$$=_{\beta} \lambda e^{\mathbf{v}} \forall(\lambda y^{hu} suivre(e, y, p^{path}) \Rightarrow up(e, y))$$

In practice we do not proceed like this in the implementation. We use the representation of formulae called λ-DRT: it better follows human language structure and allows formulae to move to the top most level (presuppositions).

41. Plurals

For plurals, one can define have maps from properties to internal integers, and operators are able to switch from collective readings:

- The three of us moved the piano. (collective reading is likely)
- The twelve students passed the exam. (each of them)
- The committees met. (each committee met or they all met)

42. Generalized quantifiers

Quite common in natural language, but very difficult to model (vagueness).

Rather viewed as properties of properties than as properties of individuals... of the predicates: "few" "the majority of" "most of" ... as acting one one predicate (as opposed to the standard, maps from pairs of sets to truth values).

A generalised quantifier $\mathscr{Q}$ acts upon a property P of α-objects and yield an α object satisfying P whenever $\mathscr{Q}x.P(x)$ holds (cf. Hilbert's τ, ε).

43. Carlotta is tall and and not tall

We handle this question by type coercion.
Types, not formulae: indeed you can not say Carlotta is tall with respect to the set of girls born on a tuesday, having a sky blue jumper, and whose mother is called Sarah.

Then a coercion in the lexicon provides the right type from her natural type(s) to a possible but undeclared type:

from 2yearOld girls to Human beings, or to her school class, but sometimes to a deictic set that has to be provided by the context: the children playing in from of me.

44. Interrogations, atrocities

I am quite worried by the "mixture" between logical formulae and types. Given a type τ it is tempting to have a predicate $\bar{\tau}(x)$ corresponding to $x : \tau$ but .... $x : \tau$ is a kind of presupposition while $\bar{\tau}(x)$ is simply a formulae has to be stipulated as "more true".

(18) The cat of my neighbour is sleeping on my car.

(19) This cat is my neighbour's.

In the first example there is a presupposition that my neighbour owns a cat. In the second, we could consider that this cat is a cat also is a presupposition. Should they be similarly modelled? When the λ-term is complex?

45. System F for meaning assembly

- syntactic elegance
- the formulae obtained as semantic representation are the usual ones
- less types (constrained) than formulae with a free variable (e.g. types $\sim$ natural comparison classes).
- finite description (e.g. a single constant for the quantifier $\forall$, which is specialized for any type and property)

46. F and subtyping

Subtyping in functional programming is quite complicated with system F despite some attempts by Cardelli et al. (complicated restriction on types) Soloviev et al. (too strong equations).

Subtyping : subtypes of $a \longrightarrow b$ are related to subtypes of a and to subtypes of b.

Does this subtyping has something to do with classes and subclasses as organised in the lexicon? NO

47. F and linguistic subtyping

Subtyping on verb types does not derive from subtyping of its arguments, subject, object, etc. Classifications of "food"and "eaters" does not provide a classification of "eating" verbs (*swallow, taste, appreciate*)?

Worse, our subtyping relations are idiosyncratic :

- not all ontological specialisations are lexically sound OK: *Mon vélo set crevé.* Not OK: *My bike is flat.*
- the transitivity of linguistic subtyping is unclear

48. F vs. Type Theory

- F defined by 4 rules and 2 reduction patterns
- Algorithmic complexity is not an issue (syntax performs parsing, semantics β-reduces the simple terms appearing in the lexicon)
- Martin-Löf TT (used by Z. Luo) many rules, many variants
- Coherence: F more complex.
- dependent types may be useful but so far I was not convinced that they are necessary.

49. Improvement with linear types

Instead of F we can use a linear version of F .

An object with two facets may have type $A \otimes B$.

Access to facets are provided by the lexicon which offers a morphism c when the language allows this coercion — remember access is idiosyncratic.

Exponentials allow to internally encode the difference between *flexible* and *rigid* transformation.
$X, X \multimap Y \not\vdash Y \otimes X$
but with the intuitionistic logic / connectives
$X, X \to Y \vdash Y \& X$

Ongoing work with Ivano Ciardelli (PhD Bordeaux — Bologna).

50. Our type theoretical model at work

Part of the Grail platform by Richard Moot:

- categorial grammar automated extraction from an annotated corpus
- parser using most likely trees
- semantic analysis in λ-DRT (convenient variant for predicate calculus)
  - basic when no semantic lexicon has been typed
  - fully implemented on a small typed lexicon

Lexical transformations partly implemented by Emeric Kien and Samira Kherfellah (internships for the tiny lexicon that has been typed.

51. Conclusion

Confrontation of logic with real linguistic data triggers difficult questions.

At least one of them, the interaction between

- the compositional calculus
- the logic for expressing "meaning"

The logical organisation of a lexicon, and the integration of lexical semantics into compositional semantics and formulae is a difficult question as well.

It is a first step before the ludic analysis of dialogue by A. Lecomte and M. Quatrini.