Towards a logical model of some aspects of lexical semantics

Christian Retoré

2010, January 25th

LaBRI-CNRS, INRIA,Université de Bordeaux

joint work with Bruno Mery and Christian Bassac

Lexical Semantics within Compositional Semantics

- A not-so-recent problem: polysemy
- Sense disambiguation and lexical semantics
- Linguistic and background knowledge
- The advent of the Generative Lexicon
- A gap within the formalism

Typical examples from Pustejovsky's Generative Lexicon

- Qualia
  - *A quick cigarette* (telic)
  - *A partisan article* (agentive)
- Dot Objects
  - *An interesting book* (I)
  - *A heavy book* (φ)
  - *A large city* (T)
  - *A cosmopolitan city* (P)
- Co-predications
  - *A heavy, yet interesting book*
  - *Paris is a large, cosmopolitan city*
  - ? *A fast, delicious salmon*
  - ?? *Washington is a small city and signed a trade agreement with Paris*

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 $e \rightarrow (e \rightarrow t)$

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

(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
...	$=$	...	extends easily to all syntactic categories when a CG is used

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

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

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

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

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”.

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

word	***semantic type*** u^* ***semantics :*** λ***-term of type*** u^* x_v ***means that 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	$e \rightarrow t$ $\lambda x_e(\mathtt{statement}_{e\rightarrow t}\ x)$
speak_about	$e \rightarrow (e \rightarrow t)$ $\lambda y_e\ \lambda x_e\ ((\mathtt{speak\_about}_{e\rightarrow(e\rightarrow t)}\ x)y)$
themselves	$(e \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t)$ $\lambda P_{e\rightarrow(e\rightarrow t)}\ \lambda x_e\ ((P\ x)x)$

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

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

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

$$\downarrow \beta$$

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

$$\downarrow \beta$$

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

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):

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

This is the semantics of the analyzed sentence.

More general types and terms. Many sorted logic. TY_n

Extension to TY_n without difficulty nor suprise: e can be divided in several kind of entities (a kind of a flat ontology).

More general types and terms. Second order types.

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, $T_1 \rightarrow T_2$ is also a type.

More general types and terms. Second order terms.

- A variable of type T i.e. $x : T$ or x^T is a *term*. [For each type, a denumerable set of variables of this 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, .

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).

More general types and terms. A second order example.

How to coordinate over any type of entities

predicates $P^{\alpha \to t}$ and $Q^{\beta \to t}$ over entities of respective kinds α and β

when we have a morphism from any type ξ to α and one from ξ to β?

$$\Lambda \xi \lambda x^{\xi} \lambda f^{\xi \to a} \lambda g^{\xi \to b}.(\text{and } (P\ (f\ x))(Q\ (g\ x)))$$

One can even quantify over the predicates P, Q and the types α, β to which they apply:

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

Principles of our lexicon

- Remain within reach of Montagovian compositional semantics
- Allow both predicate and argument to contribute lexical information to the compound
- Integrate within existing discourse models

We advocate a system based on *optional modifiers*.

Overview of the Lexicon

How much information should a lexicon store ?

- Basic compositional data: number, type, optional character of arguments
- Lexical data for adaptations: qualia, dot objects...
- Constraints on modifiers induced by lexical data
- Interpretation(s) of each term

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
  - The result is close to a language-independent hierarchy of concepts

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

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

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
  - $\Lambda\alpha\beta\lambda x^{\alpha}y^{\beta}f^{\alpha\to A}g^{\beta\to F}.((\mathrm{eat}^{A\to F\to t}\ (f\ x))\ (g\ y))$
  - Paris^{T}

The Terms: Optional Morphisms

- Each a one-place predicate
- Used, or not, for adaptation purposes
- Each associated with a constraint : *local, global,* $\varnothing$
  - $\left(\frac{Id^{F\to F}}{\varnothing}, \frac{f_{grind}^{Living\to F}}{global}\right)$
  - $\left(\frac{Id^{T\to T}}{\varnothing}, \frac{f_L^{T\to L}}{\varnothing}, \frac{f_P^{T\to P}}{\varnothing}, \frac{f_G^{T\to G}}{global}\right)$

A Complete Lexical Entry

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

$$\left(\mathrm{Paris}^T\ ,\ \frac{\lambda x^T.\ x^T}{\varnothing}\ ,\ \frac{\lambda x^T.(f_L^{T\to L}\ x)}{\varnothing}\ ,\frac{\lambda x^T.(f_P^{T\to P}\ x)}{\varnothing}\ ,\ \frac{\lambda x^T.(f_G^{T\to G}\ x)}{global}\right)$$

Global vs local use of optional morphisms. GLOBAL

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.

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.

Global vs local use of optional morphisms. LOCAL

Type clash(es): $(\lambda x^{?}.\ (\cdots(P^{A\to X}x^{?})\cdots(Q^{B\to Y}x^{?})\cdots)\tau^{U}$ [$? = A = B$ e.g. $e \to t$]

$(\Lambda\xi.\lambda f^{\xi\to A}.\lambda g^{\xi\to B}.\ (\cdots(P^{A\to X}(f x^{\xi}))\cdots(Q^{B\to Y}(g x^{\xi}))\cdots))\{U\}\ f^{U\to A}\ g^{U\to B}\ \tau^{U}$

f, g: optional terms associated with either P or τ.

This can be done for all the occurrences of x and different α and different f can be used each time.

Second order typing is required to anticipate the yet unknown type of the argument and to factor the different types for f that will be use in the slots.

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

Standard behaviour

ϕ: physical objects

small stone

$$\overbrace{(\lambda x^{\varphi}.\ (\mathrm{small}^{\varphi\rightarrow\varphi} x))}^{\mathrm{small}}\ \overbrace{\tau^{\varphi}}^{\mathrm{stone}}$$

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

Qualia exploitation

wondering, loving smile

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

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

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

Facets (dot-objects): incorrect copredication

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

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

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

T town $\quad L$ location $\quad P$ people

$f_p^{T \to P} \quad f_l^{T \to L} \quad k^T$ København

København is both a seaport and a cosmopolitan capital.

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

Conjunction of $cospl^{P\to t}$, $\mathrm{cap}^{T\to t}$ and $port^{L\to t}$, applied to tk^T

If $T = P = L = e$, (Montague)
$(\lambda x^e(\mathrm{and}^{t\to(t\to t)}((\mathrm{and}^{t\to(t\to t)}\ (\mathrm{cospl}\ x)\ (\mathrm{cap}\ x))\ (\mathrm{port}\ x)))\ k$.

AND between three predicates over different kinds $P^{\alpha\to t}$, $Q^{\beta\to t}$, $R^{\beta\to t}$

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

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

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 $\mathrm{port}^{L\to t}$ yields:

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

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.

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

Importing an existing lexicon

- Main type and argument structure: main λ-term
- *Qualia*-roles: local modifiers
- Dot objects: local modifiers
- Some specific constructions are global modifiers (e.g. grinding).
- Inheritance structure: local modifier $\rightarrow$ parent

The calculus, summarized

- First-order λ-bindings: usual composition
- Open slots: generate all combinations of modifiers available
- As many interpretations as well-typed combinations

Paris is an populous city by the Seine river

$$((\Lambda\xi\ .\ \lambda x^{\xi} f^{\xi\to P} g^{\xi\to L}\ .\ (\mathrm{and}\ (\mathrm{populous}^{P\to t}\ (f\ x))\ (\mathrm{riverside}^{L\to t}\ (g\ x))))$$

$$\{T\}\ \mathrm{Paris}^{T}\ \lambda x^{T}\ (f_P^{T\to P}\ x)\ \lambda x^{T}\ .\ (f_L^{T\to L}\ x))$$

Logical Formulæ

- Many possible results
- Our choice: classical, higher-order predicate logic
- No modalities

$$\mathrm{and}(\mathrm{populous}(f_P(\mathrm{Paris}),\ \mathrm{riverside}(f_L(\mathrm{Paris})))$$

Intermezzo: my favorite puzzle. Situation.

A shelf.

Three copies of *Madame Bovary*.

The collected novels of Flaubert in one volume (L'éducation sentimentale, Madame Bovary, Bouvard et Pécuchet)

One copy of *Jacques le fataliste*.

The volume also contains *Trois contes: Un coeur simple, La légende de Saint-Julien, Salammbô*

Intermezzo: my favorite puzzle. Questions.

- I carried down all the books to the cellar.
- Indeed, I read them all.
- How many books did you carry?
- How many books did you read?

Critics

- The classical solution with products: $\langle p_1(u), p_2(u)\rangle = u$
- (Asher's solution with pullbacks) too tight relation type structure / morphisms (only and always canonical morphisms) and unavoidable relation to product
- (Ours) not enough relation types/morphisms (no relation at all), typing does not constrain morphims,

Linear alternative

Direct representation with monoidal product $A \otimes B$ and replication !

- $A \otimes B$
  - without $\langle p_1(u), p_2(u) \rangle = u$
  - without canonical morphism
  - but the type of a transformation relates to the structure of the type.

- Types of morphims in a linear setting ($\vdash$ being $\multimap$) either:
  - <u>irreversible:</u> $A \multimap U$ since $A \not\multimap U \otimes A$
  - <u>reusable:</u> $A \rightarrow B = (!A) \multimap U$ since $(!A) \multimap U \otimes (!A)$

This leads to general questions....

Which logic for semantics? Linear Logic?

Two kind of logics:

- glue language?
  - usually base types e,t constructor $\rightarrow$
  - not rich enough
  - composition better handled with linear types
- language of semantic representation
  - usually undefined, fragment of Higher Order Logic
  - too rich, but not enough fine grained enough. Linear logic?

A natural representation (too natural?)

In the usual system we use the following: if the lambda constants are connectives, quantifiers and relational or functional symbols, then every closed temr of type t is a formula, etc.

What about a closed term of type $e \to (e \otimes t)$ and other complex types.

Interpretation, models

- usually possible worlds
- too large, uncomputable, ...
- no well defined, unless free or categorical semantics
- can we use models of linear logic (of formulae or of composition)

Argument for and against linear logic. For.

For:

- Refined both for semantic representation and as a description of the computation leading to these representations.

- Encode usual formulae and even usual typed lambda calculus.

Argument for and against linear logic. Against.

Against

- As opposed to usual semantics, no good model of first order. Phase valued models unatural, *ad hoc*
- Models of composition computation cannnot handle proper axioms — coherence spaces (Scott domains), ludics (game semantics)

[Both are even needed for maths, hence for linguistics...]

A direction that I am exploring (me but also Melliès, Lamarche,) refinement **sheaves models** of intuistionistic logic (topoi, local notion — Grothendieck, Lawvere, Lambek)

Yet more general questions. 1

performance / competence

cognitive experiments versus formal computational complexity

Algorithmic complexity not adapted. Logical complexity:

- $\rightarrow$ nesting $(e \rightarrow t) \rightarrow t$
- quantifier alternation
- order (individuals, predicates, predicates of predicates,...)

Yet more general questions. 2

How things are and works / How a specific language describes this

Ambiguity: does the lexicon (e.g. qualia structure) describe

- the world of the discourse universe (ontology)
- or a language dependent ontology:

Ma voiture est crevée. even *J'ai crevé. (une roue de ma voiture est crevée).*
** Ma voiture est bouchée. (le carburateur)* or ** Ma voiture est à plat. (la batterie)*

Cross linguistic comparisons?
book and *livre* are already different wrt. the quantificational puzzle.