Handling logical polysemy within simple type theory

Christian Bassac, BRUNO MERY, Christian Retoré

November, 1^{st} 2007

Prélude in Pauillac

INRIA Bordeaux Sud-Ouest, Signes
LaBRI (CNRS et Université de Bordeaux)
ERSS (CNRS et Universités de Toulouse et de Bordeaux)

Tentative outline after merging slide series at night

- Generative Lexical Semantics
- Linking Lexical Data to Montague Semantics
- Model Outline and examples
- Tightening optional terms and type structure
- Beyond the Lexicon

Computational Semantics

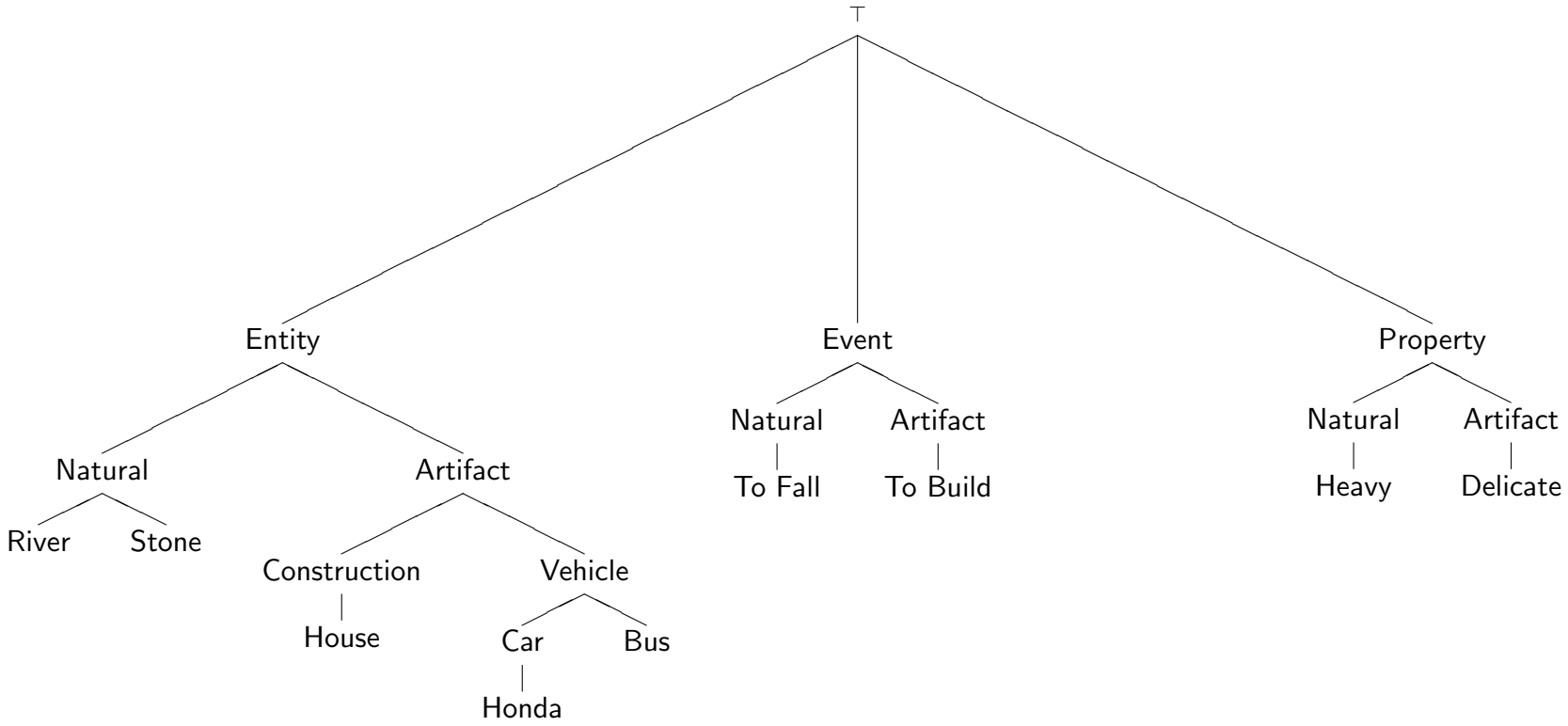
- Computing the meaning of sentences via the composition of the meaning of each lexical item.
- Classical Montague semantics uses straightforward λ-calculus, yet
- Phenomena such as logical polysemy are not so easy to account for:
- While the compositional model is sufficient, the lexical information is not.
- Our goal : to smoothly integrate lexically-induced morphisms within the Montagovian Framework

Pustejovsky's Generative Lexicon

- In Pustejovky GL book, the lexicon contains enough information to derive the related senses of a word. For that purpose, lexical entries provide :
  1. The item's *argument structure* in the form of a typed λ-term.
  2. The associated *event structure*, when needed.
  3. Various references to related concepts are included in the *qualia structure*, namely a *formal quale* referring to the properties of the item, a *constitutive quale* referring to parts and components, an *agentive quale* referring to its origin, and a *telic quale* referring to its use or purpose.
  4. An *inheritance structure* that places the entry in a hierarchy of types.

- Various elements enable the derivation of related senses and speech figures such as metonymy.

An ontology-based hierarchy of types



Composition *via* type coercion

The exploitation of the rich lexical data is made through additional mechanisms in compositional semantics such as *type coercion*, which professes that, when an argument of type α is applied to a predicate expecting to receive one of type β, instead of resulting in a type error, the application is valid in the following cases :

1. One of α and β is a subtype of the other: *type accommodation*, and allows to consider a $Honda$ as a $Vehicle$.

2. One of the qualia of β is of type α: qualia-exploitation. *A 2 GHz computer*.

3. β is actually a compound type with α as one of its aspects. This is $\bullet$-exploitation (items that present this behaviour are called "**dot objects**")

Typical examples of dot objects

- *The book was a huge pain to lug home and turned out to be very uninteresting.*
- *The lunch was delicious but took for ever.*
- *?? The tuna we had yesterday night was lightning fast and delicious*
- *Dublin is a coastal and mostly Catholic city*.
- *Bordeaux is willing a new bridge, despite the opposition of the major.*
- *Mary read the tube's wall / Cherokee.* (dot-introduction)

Current Formalizations

- The original Generative Lexicon is detailed but not really formalized.
- Subsequent works by Asher and Pustejovsky, propose type-theoretical formalisms that model the theory.
- Those models have stumbled upon the peculiar properties of multi-aspectual items, “dot objects”, and have lead to introduce many complex rules to the foundations of the logical systems involved.
- Thus, to this day, they remain quite complicated for formal and computational applications both.

Arguments against coercion – *Transfers of Meaning*

- As Nunberg (and many others) pointed out, simple coercion (i.e. considering that the term as having changed types) is unsatisfactory.
- Blindly applying coercion rules makes the relation between the original and final term disappear.
- In some cases, the denotation is strange, as in *A is parked out back*, or multi-aspectual words.
- Quantificational and individuation problems also apply.
- Some more complex logical system is needed.

Integrating lexically induced references to composition

- The problem is to link references induced by GL, in a meaningful manner, to logical forms.
- The resulting system should behave like standard logics of composition.
- Pustejovsky builds a type composition logic that ressembles Montague, and introduces additional variables in parallel to the computing of lexical inferences, using axiomatic rules.
- Our principle is to keep as close to Montague computational semantics as possible, and to deduce the links induced from the lexicon on a case-by-case basis.

Adding Lexical Information to Montague Semantics

- Contrary to the previous approaches, we tried to keep the actual semantics of composition as close to current Montague-standard frameworks as possible.

- Our goal, rather than identifying *canonical morphisms* such as the "dot" type construction, is to enable the system to use *specific morphisms* as provided by lexically-derived information.

- Thus, to each λ-term as provided by a lexical entry, we add additional, *optional* terms representing those morphisms. Each of these terms might then be employed to change the behaviour and typing of the core term.

A model based on Montague with specific morphisms

- Our base : simply-typed λ-calculus, Montague composition with optional accomodation according to the ontological hierarchy of types.

- Lexically-induced references are modeled within that calculus as *optional terms*, that might be provided together with the main λ-term of a given lexical entry and be used to modify it.

- Thus, the change of type and nature of the terms is not achieved through coercion, but through the application of explicitly defined, lexical, specific morphisms.

The Various Aspects of $Town$

Considering an item, *New York*, of assumed type $Town$:

Main λ-term	*Optional terms*
$x : Town$	$g_1 : Town \rightarrow Locus$ $g_2 : Town \rightarrow Institution$ $g_3 : Town \rightarrow People$ . . .

Each of the gs representing the relation that links the concept of $Town$ to its geographical location, ruling body, inhabitants, or other salient aspects.

The various aspect of books $Book$

Thus, an item *book* could be conceivably of two types, $PhysBook$ or $InfoBook$ (the physical object and informational content that can both be denoted using that term). They would convey the following additional terms :

$$\begin{array}{ll} x{:}PhysBook & g_1{:}PhysBook \rightarrow InfoBook,\ f_T{:}PhysBook \rightarrow Event \\ y{:}InfoBook & g_2{:}InfoBook \rightarrow Entity,\ f_A{:}PhysBook \rightarrow Agent \end{array}$$

Variant: non *book* type.

Here, g_1 is an anchor to the informational content of a (physical) book, while g_2 leads to an entity instancing the (conceptual) book. f_T refers to the *telic* (i.e., the event of *reading* the physical book) while f_A refers to the *agentive* (i.e., the *author* of the book as information).

Two modes of applying optional terms

- There are (at least) two ways of applying optional terms.
- The semantic changes induced by the lexicon might affect *every occurrence of the item* (self-adaptation). . .
- . . . Or *only the occurrence selected by a predicate* (selection-projection).
- The motivation of that distinction is linguistic data pointing out that some constructs allow *co-predication* over distinct aspects of a term, while some do not.

Transfer Modes

- A term might be applied lexical morphisms in two fashions :
  1. *Self-adaptation*, a "destructive" operation that changes every reference in the local context (discounting anaphora).
  2. *Change after selection*, that only affect a particular occurrence and allows co-predicative sentences.

- Accordingly, we decompose a term T of type α into three parts :
  1. The main term $T_0 : \alpha$, that *must* be consumed.
  2. The operators $F = \{f_i : \tau_i \to \alpha\}$ that might be used for *self-adaptation*.
  3. The operators $G = \{g_i : \sigma_i \to \alpha\}$ that might be used to change the item after its selection.

Adaptation Terms

Self-adaptation amounts to reducing an expression such as

$$\lambda x : \alpha.(P\ x) \quad (y : \beta)$$

into

$$\lambda x : \alpha.(P\ x) \quad ((f_i\ y))$$

provided there exists some $f_i : \beta \to \alpha \in F_y$.

Several options might be available.

Adaptation Example

For a $x : Car$, the *constitutive* quale provides a term $f : Car \rightarrow Engine,\ f \in F_x$.
Supposing that $Powerful$ applies to objects of type $Engine$:

A powerful car
$\lambda y : Engine\ .\ (Powerful\ y) \quad (x : Car)$
$\lambda y : Engine\ .\ (Powerful\ y) \quad ((f\ x) : Engine)$

$(Powerful\ (f\ x))$

Qualia-exploitation

Supposing objects of type $Computer$ have CPU-type objects included in their *constitutive* quale, with an adaptation term $f{:}Computer \rightarrow CPU$ to access such processing units. If clock-related predicates only apply to objects of type CPU, the following derivation is available :

A 2-GHz computer
$\exists x{:}Computer \ / \ (\lambda y{:}CPU.(Clock\ y)) \quad (x)$
$\exists x{:}Computer \ / \ (\lambda y{:}CPU.(Clock\ y)) \quad (f\ x)$

Grinding

As well as generic qualia exploitation, adaptation can be used to represent lexical rules such as *grinding*. Supposing we have an operator $f : Herb \rightarrow Food$ modeling the use of herbs in cooking, we would have :

Freshly prepared lemongrass
$\exists x{:}Herb \ / \ (\lambda y{:}Food.(Fresh\ y)) \quad (x)$
$\exists x{:}Herb \ / \ (\lambda y{:}Food.(Fresh\ y)) \quad (f\ x)$

Adaptation is destructive

The point of using *adaptation terms* is that the operation considered is destructive, and that every occurrence of the item (modulo anaphoric references) is changed. Thus, the sentence below is generally unfelicitous :

?? The tuna we had yesterday night was lightning fast and delicious

This is because we attempted to use *adaptation* with two different, incompatible types, upon a single variable.

Changing Types after Selection

- The operation is to allow some predicates to select arguments of *any* type, and, *afterwards*, to try and obtain a specific type, using optional terms if necessary.

- Thus, a problematic expression such as :

$$\lambda x : \top.(P\ (\Pi_\alpha\ x)) \quad (y : \beta)$$

  would be reduced into :

$$(P\ (g_i\ y))$$

  provided there exists some $g_i : \beta \to \alpha \in G_P \cup G_y$.

Projection after selection

- Each term T can also convey a number of distinct terms g_i, such that

$$\text{the expression } (\lambda x{:}\top.(P\ (\Pi_\alpha\ x))\quad (y{:}\beta)$$

$$\text{can be reduced into } (P\ (g_i\ y))$$

$$\text{if there is some } g_i \text{ with type } \beta \to \alpha \text{ available to } y \text{ or } P.$$

- $(\lambda x{:}\top.(P\ (\Pi_\alpha\ x))$ expresses the fact that the predicate P selects for some x of *any type*, and attempts *afterwards* to enforce the type α, by means of an additional term if necessary.

Co-predication with projection

Conversely, applying terms after the selection of the variable allows constructions such as *co-predication*.

Supposing objects of type $Town$ have available, as an alternative aspect, objects of type $People$, we can have an operator $g{:}Town \rightarrow People$ representing the link between a city and its inhabitants. Then the following derivation is available :

Boston is a large city that mostly votes Democrat
$\exists x{:}Town \ / \ ((\lambda y{:}Town.(City\ y)) \wedge (\lambda z{:}\top.(Vote\ (\Pi_{People}\ z) \quad (x)$
$\exists x{:}Town \ / \ ((City\ x) \wedge (Vote\ (\Pi_{People}\ x)))$
$\exists x{:}Town \ / \ ((City\ x) \wedge (Vote\ (g\ x)))$

Variants

There are two variants in the type encoding:

- $InfoOfBook \mapsto PhysOfBook \quad (PhysOfBook \mapsto InfoOfBook)$
- $Book \mapsto PhysBook \quad (InfoBook \mapsto Book) \qquad Book \mapsto PhysBook \quad (InfoBook \mapsto Book)$

Still these two approaches do not provide a structure for the types of the so-called dot-types.

Differences of Approach

Asher & Pustejovsky	Our model
Canonical Morphisms	Specific Morphisms
Additional Rules	Additional Terms
Conjunctive •-types	Single types
Uniform collection of aspects	Hierarchical aspects
Type enumeration in the lexicon	Morphisms detailed in the lexicon

We will examine some classical examples...

Case 1 : *Books*

A *book* consists of two immediate aspects, the physical object and the informational content.

The book was a huge pain to lug home and turned out to be very uninteresting.

$$book : Book,\ g_1 : Book \rightarrow PhysBook, g_2 : Book \rightarrow InfoBook \in G_{book}$$
$$\lambda x : Book.(ThisBook\ x), (\lambda y :\top.(Huge\ (\Pi_{Physical}\ y)))(x) \wedge$$
$$(\lambda z :\top.(Uninteresting\ (\Pi_{Info}\ z)))(x)$$

which is derived as

$$\lambda x : Book.(ThisBook\ x), (Huge\ (g_1\ x) \wedge (Uninteresting\ (g_2\ x))$$

Case 2 : $Meals$

Lunch was delicious but took forever.

Asher and Pustejovsky use $Event \bullet Food$ for meals.

Our opinion is that meals are primarily of type $Event$, with some degree of access to other aspects (including food as a whole and specific parts of the experience).

Supposing we refer to food, we use $g : Event \rightarrow Food$ and get, after derivation :

$$\lambda x : Event.\ (Lunch\ x), (Delicious(g\ x)) \wedge (Forever\ x)$$

Case 3 : *Readings*

In the previous cases, the aspects are intrinsic to the argument. The predicate might also provide some additional terms, accounting for the phenomenon Asher and Pustejovsky named •-*type introduction* occurring in :

Mary read the subway wall

We take *read* to be $\lambda x : A,\ y : \top(Read\ x\ (\Pi_{Info}\ y))$.
The term making "readings" from any suitable objects is

$$g : Entity.Artifact \rightarrow Info$$

.

Type internal structure and terms

- So far the types of a polysemic item are unrelated, say $Phys, Info, book$.
- Possibly the type changes are not as canonical as Asher says,
  ... but there are neither as unrelated as we say.
- Product works fine but conceptually we do NOT want: $\langle \pi_1(u), \pi_2(u) \rangle = u$
  A logical individual is not the sum of its components.
- The projection only exists in some cases (lexical property).
- What about $A \otimes B$, with projections provided by the lexicon — only when they are possible?

Exploring $\otimes$ for lexical polysemy

- This would leads us to something like *Book:* $Info \otimes Phys$.
- The type change functions would turn out to be projection like:

$$g : Info \otimes Phys \mapsto Phys$$

  (a constant $g : Info^{\perp}$, would do as well with CUT instead of functional application)

- This complex type opens the door to verbs introducing a new aspect of a current object such as read: *I read the tube's wall / her hand // Goodbye Colombus / Meng-Tseu / the history of scholastics / ....*

- Reads introduces the other component: *I read Small World when I started my PhD. I wanted to lend it to Pierre, but I've lost it.*

- Reads really needs an argument having both facets: $Info \otimes Phys$ and if one is missing then some is assumed.

- Lexical function provided by the lexicon $g : Info \mapsto Info \otimes Phys$ but what does it do on terms? Introduce an existential quantifier but we prefer a *Skolem function* g^ι of type $Info \mapsto Phys$: $g(x) = \langle x, g^\iota(x) \rangle$, and each time this function is used, a *new* Skolem symbol is used.

- The freedom in interpreting the Skolem funciton is attested in by discourse mismatch *The opinions on the cover where excessive.*

- Semantics of first order linear logic?

Beyond lexical data

- We covered so far some intuitions in converting GL-induced data into the compositional semantics.

- Here, we would expand on additional uses for the general mechanism introduced here. . .
  - Implementation and efficiency of such a framework
  - Interpretation and inferences
  - Integration of non-lexical data

Towards a functional implementation

- The implementation in functional, symbolic programming is straightforward as application is the only operation needed, Lisp or CaML are well-suited for the task.

- The specific morphisms can be thought of as methods attached to the class that models a type in an object-oriented formalism (such as CLOS or OCaML).

- The difference between *adaptation* and *projection after selection*, as well, can be modeled directly with the difference between *passing a variable by reference* and *passing a variable by value*.

Scope and complexity

- The *grammatical* generative power of the formalism used (e.g., context-free grammars or MCFG) is not changed.
- However, the *semantic* generative power will be greater than the canon simply-typed λ-calculus.
- The precise generative power and increase in computing complexity depends upon the number of choices made available through additional terms.
- Thus, an efficient implementation of the model would have to severely limit the number of operators available for any term and of arguments for any predicate, and probably to use heuristics.

Interpretation choice and scoring

- In the course of the derivation, several interpretation might become available as several operators might be used.

- We might envision a module that *scores* the interpretation according to the sum of a *semantic distance* value contained by each operator, and optionally that uses pragmatic inference to reject impossible interpretations, considering the sentence :

  Philadelphia wants a new bridge but the mayor is opposing it

Adjustments from multiple theoretical sources

- We focused on GL, but the framework is pretty generic by itself.
- If we separate the compositional part and translator modules that integrate specific morphisms as optional terms, we might model :
  - discursive data (as modeled by λ- or S-DRT),
  - situational data (such as non-verbal signs),
  - cultural assumptions (is a *village* likely to contain churches or communal grounds ?),
  - additional pragmatic reasoning.

Expressing idioms

- Data studied so far are mostly language-independant.

- We envision two ways of dealing with language-specific metaphors and idioms:

  1. Listing every expression and use thereof, and defining optional terms that yield the intended meaning as a logical form : costly and difficult, but needed for precise understanding and translation.
  2. Assuming that each syntactically valid sentence is semantically correct, and generating underspecified operators to deal with unresolved type clashes : economical and obviously over-generative.

In conclusion. . .

- We hope to currently have a promising model outline.
- We hope that its simplicity and generality will make it easy for a formalisation and automated computation of logical forms.
- Specific problems remains : quantification, individuation and counting are prominent.
- Use and interpretation of linear logic *with first order*.