A Montagovian Generative Lexicon

Bruno Mery, Christian Bassac, Christian Retoré

FG 2007:
The 12th conference on Formal Grammar
Dublin, Ireland
August 4-5, 2007

Organizing Committee:
Laura Kallmeyer **Paola Monachesi**
Gerald Penn **Giorgio Satta**

CENTER FOR THE STUDY
OF LANGUAGE
AND INFORMATION

1

A Montagovian Generative Lexicon

Bruno Mery, Christian Bassac and Christian Retoré †

Abstract

Generative Lexical Semantics aims, among other things, to model co-compositional meanings that classical Montague semantics is unable to express. This paper outlines a simple extension of Montague semantics which amends the standard compositional mechanisms for the use of specific terms provided by the lexicon, in order to account for this theory. We hope that such a logical system would prove both simpler and easier to implement than the formulations currently used.

Keywords Computational Semantics, Montague Semantics, Type Theory, Lexical Semantics, Generative Lexicon

Montague semantics forms a simple type logic which is often used in computational semantics to express meaning. While this process is well known, polysemous words (i.e., identical lexical items with different yet related semantics) can be difficult to represent. Moreover, recent developments in linguistics lead to think that meaning can be co-compositional, in that the same lexical items often convey different senses depending on what other terms they are applied, or apply to; classical Montague semantics has no lexical devices for that purpose.

†In the ESSLLI 2007 NDTTG workshop, we present a related work, Mery et al. (2007), in which we also use terms of the standard Montague semantics to encode so-called "dot objects". Apart from the general presentation of the model which cannot be avoided, the two papers explore two different ways it operates: here, we use one entry per word with some kind of projection onto the various aspects, whereas, in the solution proposed in the NDTTG workshop, we use several entries per word with translator terms between any two of them. The linguistic constructions examined also differ in the two articles.

FG-2007.
Organizing Committee:, Laura Kallmeyer, Paola Monachesi, Gerald Penn, Giorgio Satta.

Thus, theories have been proposed to express this particular form of polysemy, such as transfers of meaning, by Nunberg (1993), or a generative lexicon, by Pustejovsky (1995).

1.1 Background

1.1.1 The Generative Lexicon

Pustejovsky's theory, the Generative Lexicon (henceforth GL) gives a strongly-motivated model for many cases of lexical polysemy, together with a rich structuration of the meaning of concepts. It uses a hierarchy of types, corresponding to lexical concepts derived from an ontological knowledge base. Each lexical entry includes the number and types of arguments needed (for a predicate), the characterization of the event structure associated with the concept, if any, and the associated *qualia*, or modes of explanation of the concept: what its distinguishing characteristics are (*formal*), what it is made of / part of (*constitutive*), what it can be used for (*telic*), what can cause it to come into being (*agentive*)... the idea being that a word can, under certain conditions, refer to any of its qualia role.
In addition, some lexical entries are of a *complex* (or *dot*) type, expressing two (or more) aspects of different types. For instance, if one supposes that there are physical objects of type P and informational contents of type I, then the lexical item *book* would be of type $I \bullet P$.

1.1.2 Licensing type changes

The semantics supposed by GL is rather intuitive: each predicate contains, in its argument structure, the number and type of arguments needed. If each argument is present with the correct type, then a λ-term is formed by application, like in classical Montague semantics. In addition, the theory licenses certain cases of application ($f^{\alpha\to\tau}\ x^{\beta}$) with $\alpha \neq \beta$ via various *type coercions*: if one of α or β is a subtype of the other, then the application is valid and is called a *subtype coercion*; if α is part of a certain quale of β, even with α and β disjunct, then the application is valid and is called a *true complement coercion*; if β is a complex type $\alpha \bullet \gamma$ for some γ, then the application is valid by $\bullet$-*exploitation*.

1.1.3 Current Formalizations

The formalization of the structures and compositions of these semantics is, however, far from trivial. The original theory, as well as more recent formalisms such as Asher and Pustejovsky (2005), Pustejovsky (2006), or Asher (2007) seem not fully satisfactory and remain open for improvement.

In particular, the semantics of complex (*dot*) types has been the subject of many successive formalizations. It looks like this phenomenon (in which a single lexical item seems to have two or more *aspects* of different types) cannot be expressed by straightforward rules because its representations are manifold, and each case is subtly different from any other.

For instance, the term "book" can be thought of a compound containing two different aspects (informational content and physical object) which are easily accessible, but are not equal: to speak of a book as a physical item always assumes there is an information contained[1], whereas some uses of the term (e.g., a book in preparation) do not suppose any physical representation. That use of a complex type is logically different from compounds such as Event•Information (which is used in Pustejovsky (2005) for lectures, lessons and the like), where the event is always present and its informational content is in the background...

1.2 Our proposal

1.2.1 Basic assumptions

We are strongly inclined to think that most of the additions to GL can accurately be modeled without the addition of new logical rules to classical Montague semantics. In particular, we contend that the phenomenon named "dot object" by GL can be accounted for without loss as a collection of given phenomena, which can each be treated without resorting to some special set of rules. We believe that using the standard Montague mechanisms with specific additional terms can prove a better approach.

1.2.2 Logical Foundations

Thus, our model is based upon classical Montague semantics, simply-typed λ-calculus, and a hierarchical typing system such as detailed in Pustejovsky (2006) : $\top$ is the universal type with three main subtypes, *Entity*, *Event*, *Property*, the various subtypes of which form the complete type partially ordered set. We also have two additional mechanisms, supported by the lexicon:

- *self-adaptation*: a lexical item has access to a number of optional terms that allow it to change types, and
- *selection-projection*: a predicate may select for an argument of any type and, afterwards, attempt to enforce a certain type upon it, using additional terms for this purpose.

[1] With the exception of blank notebooks, which are usually specified to be so.

This system is compatible with the ideas from both Nunberg (1993) and Pustejovsky (1995) that the lexicon licenses, for each entry, some type-shifting or type-coercing operations. It does not change the general rules for Montague semantics, and thus complexity, soundness or correctness do not change either. The difference is that each lexical item contributes *at least* a term, which *must* be used exactly once (as per usual), plus a (finite) number of *optional* terms which might be used, if necessary, to change the type of the first term. To illustrate this point, the lexical entry for "town" could provide the following terms (supposing that items such as *New York* have the type $Town$):

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

In a GL framework, those terms would appear within the *argument structure* of the lexical entry concerned. Here, the gs are selection-projection operators that provide access to the multiple aspects a town might be predicated of: thus, $(g_1\ x)$ would reduce to the actual physical location y of the town, $(g_2\ x)$ to a representing institution z such as the city council, $(g_3\ x)$ to a set of people p such as its inhabitants, etc.

1.2.3 Complex types equivalents

First of all, the formalization we use is more restricted than the one assumed in Pustejovsky (2005). Thus, many phenomena (such as grinding or complex events) are not considered to be "dot objects" at all, only the most canonical examples (i.e., "town") remain quite comparable.

Then, in our formalism, a "dot object" would simply be a lexical term with its rich structure, its associated λ-term and type, but containing additional terms that might be used after the selection of the argument in order to refer to one of its aspects. Within each of the aspects, such terms might be provided for reference to different aspects, or the original item.

1.3 A modular system

1.3.1 Logical and structural levels of interpretation

The model supposes at least two different parts : a logical system, using λ-calculus and type-coercing operators to provide logical forms akin to Montague semantics, and a rich structure of information, containing lexical data. For instance, Gupta and Aha (2003) present a formalism which can adequately represent this structure.

Specifically, when a certain type-coercing term is used to compute the logical form of the sentence, the following operations might be performed:

- *a minima*, the *type* of the target term is changed;
- the λ-*term* itself might be modified (i.e., the number and types of a predicate's arguments);
- the data structure associated with the item (i.e., its qualia) might be changed.

1.3.2 Discourse integration

In addition, the use of type-changing term licenses later reference to the derived object further along in the discourse. Thus, the system is easy to interface with discourse-oriented formalisms such as DRT, at least with respect to one of the main issues of discourse analysis: determining which objects are available for reference in, e.g., anaphora resolution. This is a major concern of Asher's analysis of GL.

1.4 Transfer modes

In this section, we present the two possible ways of applying type-coercing operators in our model, and the fundamental differences they lead to.

In order to formalize these two ways, we shall amend λ-calculus such that any term $T : \alpha$ stands for :

$$(T_0 : \alpha \ , F = \{f_i : \tau_i \rightarrow \alpha\} \ , G = \{g_i : \sigma_i \rightarrow \alpha\})$$

We shall use F_T and G_T for accessing the sets of additional terms. Then, application is redefined for two minimal cases :

1.4.1 Adaptation

An expression such as:

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

might be reduced into:

- $\lambda x : \alpha.(P_0\ x) \quad (y_0 : \beta)$, if α and β are equal or compatible (classical application with optional accommodation);
- $\lambda x : \alpha.(P_0\ x) \quad ((f_i\ y_0))$, if there exists some $f_i : \beta \rightarrow \alpha \in F_y$ (self-adaptation of the argument by the means of an optional constant).

The reduction might prove non-deterministic at this point, if there are several rewritings available. Note that, for α and β identical or compatible, $\lambda x : \alpha.(P_0\ x)(y_0 : \beta)$ reduces, as is standard, to $(P_0\ y_0)$. The system is thus equivalent to classical single-typed λ-calculus with accommodation when, for all T, $F_T = G_T = \emptyset$.

For example, consider an object x of type Car. Supposing its *constitutive* quale includes an object of type $Engine$, we would have a term $f : Car \rightarrow Engine,\ f \in F_x$. Supposing that some use of "powerful" can apply to objects of type $Engine$, we could have a derivation:

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

That mechanism suffices to licence most cases of qualia-exploitation, as well as type accommodation and some particular lexical rules such as *grinding*, where an object is subject to an irrevocable change, such as $Herb \rightarrow Drink$, which is modeled by a type-coercing operator, as in the following phrase (we consider that $Bitter$ is of type $Drink \rightarrow \underline{t}$):

(2) Some bitter tea
$\lambda y : Drink\ .\ (Bitter\ y) \quad (x : Herb)$
$\lambda y : Drink\ .\ (Bitter\ y) \quad ((f\ x) : Drink)$

1.4.2 Type changes after selection

An expression such as[2]:

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

might be reduced into:

- $\lambda x : \top.(P_0\ (\Pi_\alpha\ x)) \quad (y_0 : \beta)$, if α and β are equal or compatible;
- $\lambda x : \top.(P_0\ (\Pi_\alpha\ x)) \quad ((f_i\ y_0))$, if there exists some $f_i : \beta \rightarrow \alpha \in F_y$;
- $(P_0\ (g_i\ y_0))$, if there exists some $g_i : \beta \rightarrow \alpha \in G_P \cup G_y$.

The last possibility is "change after selection": the constant type-coercing term is applied only to the local copy of the argument as selected by the predicate. It is still possible that reduction might prove non-deterministic, as many distinct rewritings could be available.

This mechanism can express the construct known as *co-predication*: if two or more predicates select the same lexical item, they may force different types upon it . When *self-adaptation* is required, conversely, the item changes its type for *every* reference used, co-predication is impossible and sentences such as (3) are usually unfelicitous:

(3) ?? This angora fits neatly on your shoulders and died of old age

As an example of selection and type change, we might have the types $Town$ (T), $People$ (P) and $Locus$ (L), with terms $g_1 : Town \rightarrow Locus$ and $g_3 : Town \rightarrow People$ which, as illustrated earlier, establish the relationship between a town, its geographical situation and its inhabitants. Then we could have:

[2] Π_τ being, roughly speaking, a "projection" to τ.

(4) Dublin is a coastal and mostly Catholic city
$(\lambda y : \top.(Coast\ (\Pi_L\ y)) \wedge \lambda z : \top.(Catholic\ (\Pi_P\ z)))\quad (x : T)$
$(Coast\ (\Pi_L\ x))\ \wedge\ (Catholic\ (\Pi_P\ x))$
$(Coast\ (g_1\ x)\ \wedge\ (Catholic\ (g_3\ x)))$

This change of type after selection, which enables to reference the newly selected aspect, can be used to analyze some of the most complex puzzles involving co-predicative sentences.

1.4.3 Choosing the right transfer mode

The difference between the two transfer modes is that applying the change after selection licences full co-predication. Thus, the definition of the lexically-induced operators will have to take at least two things into account: whether multiple aspects of the argument can coexist in a predication, and whether a predicate needs type change after or before selection. In GL, co-predication is mostly used with complex types and a number of well-identified phenomena, and the difference is quite straightforward.

1.5 Comparison with the current approach

After this brief outline of our methods, let us review the differences with such formalizations as Asher and Pustejovsky (2005). Our goal is to show that the complex rules used in those formalizations to solve co-predication and quantificational puzzles are not so strictly necessary. The chief difference between the methods lies with coordinate co-predications over "dot objects", which we express using our "type change after selection".

1.5.1 Selection, transfer, and coercion

The principles behind our approach of those phenomena go back to Nunberg (1993), who had already suggested that the meaning of the selected argument in itself does not change. Rather, it is the *predicate* that could select any argument and will use one of its properties. Pustejovsky and Asher's opposition to this analysis is based on overgeneration: if the predicate can coerce its argument, then phrases such as "to read a flower" would be valid.

Their formalization, then, uses dot objects and additions to the logical system, with elementary rules that modify the *application* of a term to another. Apart from the notational problems of these rules, the fact that *elementary rules* contain predicates such as "the smallest subformula within a term ϕ which was responsible for the original meaning of a variable x", and for which, unfortunately, no algorithm is provided (we are inclined to think that the complexity analysis of such an al-

gorithm would necessarily be non-trivial), or the fact that these rules need external inference modules in order to work, does not look so good for the overall efficiency of Pustejovsky and Asher's solution.

So, admitting that transfers of meaning can over-generate sentences, and that the current dot object formalization is not so adequate, why should our approach prove any better ?

From our point of view, a meaning transfer makes more sense than a type change for co-predicative uses. Indeed, when Pustejovsky and Asher use dot objects together with a predicate, they need to introduce a variable corresponding to the selected aspect, and to leave the original object intact in order to make other predications – which only differs from the transfer approach in that the selected property, in the latter, is supposedly known. Of course, a predicate cannot turn any argument into any type, so as to avoid over-generation. This is why, in our methods, the type forcing by the predicates after selection of an argument is effective only when the argument licenses the operation: informally, this really would be equivalent to the complex types as they are outlined in Pustejovsky (1995), but this use of type-coercing operators *does not change the basic rules of the logic system.*

The main difference between the two methods is thus that the type coercion is enabled by *generic terms* through the addition of *specific type construction rules* into the system in the first case, and by *specific terms* through the *usual compositional rules* in our case. The interpretation of the applied type-coercing term will provide the specific relation between the aspects referred to, rather than the generic (and thus very underspecified) link that exists in the case of a compound type. On the other hand, we lose quite a convenient product construction.

The expressive power of our formalism depends on the operators licensed for each item. If one simply allows as many self-adaptation terms as there can be valid instances of qualia exploitation, type accommodation or grinding rules for the lexical item concerned, and as many selection-coercion terms as there are different types in a "dot object" (none if the concerned item is not a "dot"), then our system is fully equivalent to Pustejovsky and Asher's, and in particular does not over-generate sentences. With some adjustments, it might express slightly more about the trickiest co-composition phenomena, though.

1.5.2 Examples

Finally, let us present the way that examples from Asher and Pustejovsky (2005) could be treated using our system.

Informational content, physical instances

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

Several treatments are possible. An easy way to envision this case is to have a term "book" of type $Book$ which could then be derived into a term of type $PhysBook$ which would represent an underspecified, individual, artifactual object with a cover, a set of pages, etc. . . and into a term representing the informational content, of type $InfoBook$. We then have type-coercing operators $g_1 : Book \rightarrow PhysBook$ and $g_2 : Book \rightarrow InfoBook$.

For the target sentence, we have

$$\lambda x : Book.(ThisBook\ x), (\lambda y : \top.(Huge\ (\Pi_{Physical}\ y)))(x) \wedge$$
$$(\lambda z : \top.(Uninteresting\ (\Pi_{Info}\ z)))(x)$$

which is derived as

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

Complex event types

(6) Lunch was delicious but took forever.

This sentence has prompted Pustejovsky and Asher to use the type Event • Food for meals. The problem with this interpretation is that there are many other aspects than these two which can be referenced – e.g., some would say that "delicious" does not correspond to food as such, but rather to the perception by the speaker of a sub-event of the lunch, the act of eating (compare "Lunch was great. . . ").

In addition, a "lunch" is a scenario comprising several aspect which cannot be directly referenced (in a co-predicative phrase such as "lunch's dessert was delicious but took forever"): the different parts of the lunch, the settings, the cook, etc. The fact that these aspects can be present contextually (as in "in yesterday's dinner, the lighting was perfect"), and not directly, hints that the aspectual structure is *nested*: the *lunch* takes place at a certain *location*, which has a certain *setting*, comprising some kind of *lighting*, and we can only directly access foremost elements of such a hierarchy (i.e., the intrinsic components of *lunch* itself).

So, we assume that the main entry for lunch is a subtype of Event, as it is the one aspect of the term which can always be referred to, and a certain number of operators are available, allowing for reference to sub-events, food as a whole, date and time, etc.. . .

For this sentence, we could use either $g_1 : Event \rightarrow Event$ (a type accommodation term linking "lunch" with the sub-event "eating lunch"),

or $g_2 : Event \rightarrow Food$, to keep the interpretation assumed by the original article. Anyway, after derivation, we shall have

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

Complex type introduction

(7) Mary read the subway wall

Examples like this one induced the inclusion of •-Introduction mechanisms into the theory. The idea is that "read" selects Physical•Information arguments, and thus that an object subtype of Physical would be coerced into this complex type. But this also generates incorrect phrases such as "read a river", which (generally) have no coherent meaning ; Pustejovsky and Asher then use an inference mechanism to determine whether this •-Introduction rule can be applied or not. The whole mechanism seems inadequate, since a transfer of meaning could also be limited by an inference module, but also because "read" is the *only* convincing example of •-Introducing predicate.

Our idea here is that, while "read" does indeed select a physical object with an informational content, the determining property is that this information has always the possibility to be instanciated into an entity. Thus, if we view "read" as a predicate $\lambda x : Ay : \top(Read\ x\ (\Pi_{Info}\ y))$ (the agent x being the reader), we have at our disposal a type-coercing term, $g : Info \rightarrow Entity.Artifact$[3], which does not come from the operators of y, but rather is contributed by the lexical definition of the type Information. The use of that operator in "read" allows to attach an (underspecified) informational content to any artifactual entity, and to read this information. The entity must, in general, be artifactual, for the information must have been inserted here in order to be extracted. Thus, "read a flower/river" is incorrect, whereas the uses with artifactual entities abound: one might "read"...

(8) a. a binary file
    b. the mayor's expression
    c. the opponent's strategy

In each case, an informational content is supposed and attached to the entity. Typically, a specification of the information would follow in the discourse.

Of course, in some contexts, a limited number of natural entities might be "read". In that case, it is one of the arguments, as usual, which contributes the corresponding term – a "shaman" agent might

[3]The subtype of *Entity* denoting artifacts, i.e. lexical entries that have a specified *telic* or *agentive* quale.

thus provide some specific terms in order to read a "flock of birds" or a "riverstream", for instance.

1.6 Beyond lexical meaning

1.6.1 Interpretation choice and scoring

When several terms f_i, g_i are available for use, several defeasible interpretations might be possible, and another module might have to *choose* between the possibilities, resolving logical constraints (as in "The dinner was lousy but the food was all right").

Alternatively, several interpretations might remain possible after all constraints have been resolved. In that case, a notion of "score", associated to the likeliness of using any operator, might help establish which interpretations are most natural or feel more "forced". For example, "an amazing book" might indicate that the book as information (writing, plot...) or as an object (in the case of precious, rare items) is considered to be amazing – both interpretations can certainly be derived, and they are non-contradictory, yet it might be wished to focus on the first one as being more usual.

1.6.2 Integrating more than the Generative Lexicon

While our system is intended to integrate the compositional mechanisms introduced by GL, it is not dependant upon it, as it is formally but an extension to λ-calculus with simple types and accommodation. It could thus be used to model other theories of lexical meaning. More importantly, it could also be used to add to GL some sense of discourse semantics, pragmatics or, for instance, cultural variants of a concept.

If one supposes a generic implementation of our system (which would be straightforward enough using functional and symbolic programming), then the integration of various layers of meaning would be achieved using modules that translate constraints into optional terms for GL or, for example, λ-DRT. The generative power of the resulting implementation could thus cover many aspects.

1.7 Conclusion

In the course of this work, we have proposed a solution for compositional semantics involving lexically specified rather that canonical morphisms. It is our hope that this approach will prove both simpler to formalize and easier to implement than the current formulations used, and will be worth investigating. The use of additional terms for multi-aspectual objects could thus help highlight a hierarchy of the aspects, and a more precise definition of such type compounds in type theory

would also be interesting to pursue (for instance, Nicholas Asher is currently investigating fibered-products for that purpose).

We hope, with the improvements to generality obtained, to express more precisely some of the most difficult semantic phenomena.

Acknowledgements

The work presented here is part of the preliminaries of a doctorate thesis in computer science funded by the French Minister of National Education (*MENSR*), conducted in the University of Bordeaux, at the LaBRI. We are indebted to Nicolas Asher (IRIT – Toulouse) and the people at Signes, as well as the original reviewers, for their many helpful remarks, comments and co-concommital work.

References

Asher, Nicholas. 2007. A type driven theory of predication with complex types. *Fundamenta Informaticæ* To appear.

Asher, Nicolas and James Pustejovsky. 2005. Word Meaning and Commonsense Metaphysics. Semantics Archive.

Gupta, Kalyan Moy and David M. Aha. 2003. Nominal Concept Representation in Sublanguage Ontologies. In *Second International Workshop on Generative Approaches to the Lexicon*.

Marlet, Renaud. 2007. When the Generative Lexicon meets Computational Semantics. In *Fourth International Workshop on Generative Approaches to the Lexicon*.

Mery, Bruno, Christian Bassac, and Christian Retoré. 2007. A montague-based model of generative lexical semantics. In R. Muskens, ed., *New Directions in Type Theoretic Grammars*. ESSLLI, Foundation of Logic, Language and Information.

Nunberg, Geoffrey. 1993. Transfers of meaning. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 191–192. Morristown, NJ, USA: Association for Computational Linguistics.

Pustejovsky, James. 1995. *The Generative Lexicon*. MIT Press.

Pustejovsky, James. 2005. A Survey of Dot Objects. Author's weblog.

Pustejovsky, James. 2006. Type Theory and Lexical Decomposition. Semantics Archive.

Vanier, Jules, Christian Bassac, Patrick Henry, Renaud Marlet, and Christian Retoré. 2006. Toward a knowledge representation model dedicated to the semantic analysis of the sentence. Tech. rep., INRIA.