Advances in the Logical Representation of Lexical Semantics

Bruno Mery, Christian Retoré

Advances in the Logical Representation of Lexical Semantics

Bruno Mery[1] and Christian Retoré[2]

[1] Université de Bordeaux, France
bruno.mery@me.com
[2] IRIT-C.N.R.S., Université de Bordeaux, France
christian.retore@labri.fr

Abstract

The integration of lexical semantics and pragmatics in the analysis of the meaning of natural language has prompted changes to the global framework derived from Montague. In those works, the original lexicon, in which words were assigned an atomic type of a single-sorted logic, has been replaced by a set of many-facetted lexical items that can compose their meaning with salient contextual properties using a rich typing system as a guide.

Having related our proposal for such an expanded framework ΛTY_n, we present some recent advances in the logical formalisms associated, including constraints on lexical transformations and polymorphic quantifiers, and ongoing discussions and research on the granularity of the type system and the limits of transitivity.

1 Introduction

Our general purpose is to update the framework colloquially known as "Montague Grammar", (based on principles described in [14]) used in formal linguistics to provide an analysis of the syntax, semantics and interpretation of the human language, with data pertaining to lexical semantics according to [21].

Our main principles are the following:

- The semantics should retain its compositional aspects.
- The lexical semantics should allow to distinguish between acceptable and unacceptable predications.
- The mechanism should allow some degree of flexibility for different contexts.
- The framework should be easy to define, describe and implement.

The simplest example of situation considered acceptable by standard Montagovian analysis is selection restriction that prescribes[1], e.g.:

(1) * The chair barked.

Restriction of selection, polysemy and more elaborate phenomena including the creative use of words and co-predicative statements have prompted the inception of the theory known as the Generative Lexicon, detailed in [21]. In [3] and other related works, we have elaborated a Montague-compatible framework corresponding to that theory and the above principles. It has proved useful in the study of

[1] We do not say that this sentence should never receive a semantical analysis. There are some contexts that can provide a suitable parse, and even in the absence of those, something may be derived (if only that there must exist some unprovided context that makes sense of the utterance). See Section 4.4, p. 13 for more details.

several phenomena, including deverbal nouns ([24]), the virtual traveler ([23]), etc. It is also currently partially implemented in a version compatible with λ-DRT.

Our goal here is to examine some of the mechanisms introduced by our framework, such as constraints on the availability of meanings and the modelling of articles with quantifiers, and to discuss several points pertaining to the type system and its granularity.

1.1 Montague Grammar

Richard Montague, in [14] and other related works, expressed the idea that any human language could be considered as a formal language, and receive a logical analysis that could be interpreted in terms of truth. He had a (rather Chomskyian) universal aim, resulting in the analysis process that would come to be called Montague Grammar, and was largely adopted after his death.

An abridged version of this process is the following:

1. The *syntax* of the sentence is analysed using an appropriate formalism, such as categorial grammars. This provides a binary syntactic tree that indicates, minimally, at each node which subtree (the predicate) applies to which other (the argument).
2. A *lexicon* provides a semantic term that corresponds to every word, for example as a simply-typed λ-term. The sentence is analysed as a logic formula by associating to each leaf of the syntactic tree (individual words) their corresponding term, and by using the tree to apply each argument to its predicate.
3. The formula can then be computed by β-reduction and interpreted, for example, as a truth value.

Two common uses of the Montagovian analysis are:

- Using a lexicon with two types, **e** for all entities and **t** for truth values; formulae are of type **t** and receive a boolean interpretation.
- Using the above lexicon with the additional type **s**, corresponding to indices for possible worlds; the interpretation is the set of indices satisfying the formula, in Kripke semantics, and can be combined with modal logic.

This is an example of a lexicon of the former kind.

word	***semantic type*** u^* ***semantics :*** λ***-term of type*** u^* x^v *the variable or constant x is of type v*
some	$(e \to t) \to ((e \to t) \to t)$ $\lambda P^{e\to t}\ \lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^e (\wedge^{t\to(t\to t)} (P\ x)(Q\ x))))$
club	$e \to t$ $\lambda x^e (\texttt{club}^{e\to t}\ x)$
defeated	$e \to (e \to t)$ $\lambda y^e\ \lambda x^e\ ((\texttt{defeat}^{e\to(e\to t)}\ x)y)$
Leeds	e Leeds

Figure 1: A simple semantic lexicon

Considering the sentence

(2) Some club defeated Leeds

Assuming the underlying syntactic structure is

(some (club)) (defeated Leeds)

The analysis of the semantics is thus:

$$\Big(\big(\lambda P^{e\to t}\ \lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^{e}(\wedge(P\ x)(Q\ x))))\big)\big(\lambda x^{e}(\texttt{club}^{e\to t}\ x)\big)\Big)$$
$$\Big(\big(\lambda y^{e}\ \lambda x^{e}\ ((\texttt{defeated}^{e\to(e\to t)}\ x)y)\big)\ Leeds^{e}\Big)$$
$$\downarrow\beta$$
$$\big(\lambda Q^{e\to t}\ (\exists^{(e\to t)\to t}\ (\lambda x^{e}(\wedge^{t\to(t\to t)}(\texttt{club}^{e\to t}\ x)(Q\ x)))))$$
$$\big(\lambda x^{e}\ ((\texttt{defeated}^{e\to(e\to t)}\ x)Leeds^{e})\big)$$
$$\downarrow\beta$$
$$\big(\exists^{(e\to t)\to t}\ (\lambda x^{e}(\wedge(\texttt{club}^{e\to t}\ x)((\texttt{defeated}^{e\to(e\to t)}\ x)Leeds^{e})))\big)$$

The end result is the logical form of the sentence, that corresponds to the following formula in predicate calculus: $\exists x : e\ (\texttt{club}(x)\ \wedge\ \texttt{defeated}(x, Leeds))$.

The analysis in Montague Grammar is thus comprised of *two* levels of logical analysis:

- The logic of assembly (glue logic, compositional calculus). Here, simply-typed λ-calculus with base types e and t. The terms provide a proof to the formula given by the types in intuitionist logic, à la Curry-Howard.
- The logic of semantic representations. Here, higher-order predicate calculus. The resulting formula can be interpreted as a truth value.

Montague Grammar provides a semantic representation. It can be interpreted in terms of truth, possible worlds where the representation is true, etc. This provides an analysis of the possible realisations of a sentence. Our aim, conversely, is to refine the level of meaning assembly in order to take into account precisely *what* is said, and *how* this is said at the lexical level.

1.2 Polysemy and the felicity of sentences

Yeoshua Bar-Hillel famously used the following example in 1960 to illustrate the difficulties of Machine Translation:

(3) a. The box was in the pen.
b. The pen was in the box.

In (3a), *pen* is a container; in (3b), *pen* is a simple tool. This should not come as a surprise to any linguist; the notion that words have sometimes radically different meaning that do not always get carried after translation is known as *polysemy*. What it meant was that the use of an atomic lexicon for translation or other tasks of natural language processing was insufficient. Some additional mechanisms were needed that would allow to choose, in the view of the context, the correct meaning for every word; in his report, Bar-Hillel said that this required encyclopeadic knowledge, and that was *surely utterly chimerical and hardly deserve[d] any further discussion*.

However, those examples are hardly the most complicated problem posed by polysemy. *Pen* as a container and *pen* as a tool are examples of *contrastive ambiguity*, that is, of two words having completely different meanings but the same pronunciation and/or writing. There are various ways to deal with that ambiguity.

More problematic is the case of *logical polysemy*, in which a word can express several meanings that are related logically, yet different. A *bank* can be a geographical feature or a financial institution (this is contrastive ambiguity), but, in the latter case, it can variously refer to a building, a society or some unnamed person playing a specific role in that society.

Consider the following:

(4) * The chair barked.

(5) The bank killed my account.

Not only do we want our framework to be able to tell that (4) is not acceptable in most contexts (because *chair* is not a valid argument for the predicate *to bark*) while (5) is (*even though bank* and *account* are not associated with *to kill* in a normal sense), we also want it to be able to tell us what process is at work in (5).

2 Our Generative Framework

2.1 The Generative Lexicon

In [20], James Pustejovsky argued that logical polysemy is an essential feature of human languages, allowing a speaker to make the language evolve by the way of creative use of words. This means that the lexicon cannot be enumerative as additional word senses are added all the time; he proposed that the lexicon should contain enough information to generate for each word the correct meaning it would receive in context. As this represents a set of mechanisms for word sense generation rather that fixed meanings, he called it the Generative Lexicon. As described in [21], the framework is Montagovian, with the following features:

Types in the terms – Every term is typed, as in Montague grammar, but using a rich typing system. The point here is that application is constrained, and some predicates may only apply to a specific kind of argument. Thus, the logic implicitly used by the generative lexicon is TY_n, the typed logic with n sorts, which has $n+1$ types: $\mathbf{t}$ for truth values, and as many types as needed that replace the Montagovian e. Those types are not arbitrary, but constructed according a hierarchy of concepts under the form of an ontology.

Ontological hierarchy – The constraints in application are type-driven. Thus, types are an important mean to categorise terms in the systems. The organisation of types is hierarchical and ontological, as subtypes are understood to be compatible with constraints that would require their supertype. This inheritance hierarchy is based upon dual hyponymy-hyperonymy relations, with *vegetable* being a subtype of *food*, in itself a subtype of *physical object*, and so on. The construct of such an ontology is a subject in itself for Pustejovsky. In [19], he argues that another feature of the Generative Lexicon, Qualia, forms more robust guideline that existent ontological resources for lexical semantics (notably *WordNet*). As such, the ontology proposed in GL remains an outline of a work in progress.

Qualia – The most distinguishing feature of the Generative Lexicon is the inclusion of the staple Aristotelician *Qualia*. James Pustejovsky, citing [17], applies this old philosophical concept to words and lexemes. And, while those concepts are insufficient to describe a physical phenomenon, they are indeed associated to *the idea* that humans have about a phenomenon. They are ontological concepts, part of metaphysics, of our understanding of the world rather than of the world itself, and as such quite close to linguistic concepts.

Thus, in the Generative Lexicon, each lexical item comprises a Qualia structure, divided in four parts: the *formal quale*, pertaining to intrinsic properties; the *constitutive quake*, pertaining to components or materials; the *agentive quale*, pertaining to the origin; and the *telic quale*, pertaining to the purpose.

It would, when complete, be able to deal with phenomena such as:

Contrastive polysemy : the "accidental" polysemy between, e.g., *bar* as a place and as a physical object. The refined typing should be enough to distinguish between the two, as one would be typed $Location$ and the other $Physical$.

Accommodation : hyponymic relations, where a word is taken to be its ontological supertype, are included in the system. Thus, locutions such as *my Honda* can be applied to predicates that require cars or physical objects.

Qualia exploitation : the four qualia are designed to be used in place of the word itself if required by context or sentence structure. Thus, a *naïve article* should be parsed using the agentive quale because *naïve* takes an agent as its argument. This particular case is a *type coercion*: the predicates calls for a specific type which is incompatible with the one of the argument, but there is a transformation available to coerce the argument into the required type via qualia exploitation.

Lexical transformation : some other operations can be devised within the lexicon. [6], for instance, suggests *grinding*, which can make available to the semantics the meaning of something that has undergone a destructive operation. *Delicious chicken*, for example, indicates that the lexical item referred to as *chicken* should not be considered as having an *animal* type, but a *food* type, as allowed by a grinding operation (specifically, cooking).

Facettes : some words can refer to different aspects, or *facettes*, and have proved very difficult to model in the Generative Lexicon. They are the object of specific mechanisms. The canonical example is *book*, as the word can refer to either a *physical* object, made of leather, cardboard and paper, or its *information* content that can be written or read.

However, the Generative Lexicon was incomplete. The case study is linguistically impressive and well motivated, but the assembly language is not really described, and while every structure is detailed, a correct way to perform the desired derivation is simply assumed. There is no formal guide that can distinguish between the different cases, no algorithm that would choose the correct construction in an analysis. In fact, what is missing is an actual model for composition.

As such, the Generative Lexicon was not directly suited for formal or computational implementation. It also famously lacked a comprehensive logical framework for complex types and co-predicative sentences. The author was aware of the latter fact and started working on a logically sound system in [22], that would later be continued by Asher.

Several solutions have then been proposed, including ours (in [13], [12], [11], [3]), systems by Cooper ([5]), Asher ([1], [2]) and Luo ([27], [9], [10]).

2.2 The Logic and Calculus

Our own proposal is based on the following elements:

- The description language (the logic used for final semantic representations) is the many-sorted higher-order predicate calculus.
- The calculus used for the computation of such formula, and thus meaning assembly, is the λ-calculus with second order types, Girard's System-F.
- The lexicon associate each word with a single main λ-term (as usual) and finitely many optional terms.
- The infelicity of sentences (in case of a problem of selection) is evidenced by type mismatches during application.
- The use of lexical information to resolve type mismatches is provided by the optional terms associated to the words, or main λ-terms, and guided by the types, rather than derived entirely from the type system.

In brief, our logic of assembly based on System-F and TY_n (described in [18]) comprises the following elements:

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

Terms encode proofs of quantified propositional formulae:

- A variable of type T i.e. $x : T$ or x^T is a *term*, and there are 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 $\Pi\alpha.T$ whenever α is a type variable, and $\tau : T$ a term without any free occurrence of the type variable α in the type of a free variable of τ.

The latter restriction is the usual one on the proof rule for quantification in propositional logic: one should not conclude that $F[p]$ holds for any proposition p when assuming $G[p]$ — i.e. having a free hypothesis of type $G[p]$.

The reduction is defined by two schemes which are quite similar:

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

As an example, in Ty_n we need a first order quantifier per sort (or base type). Here, a single quantifier $\forall$ of type $\Pi\alpha.\ (\alpha \to \mathbf{t}) \to \mathbf{t}$ is sufficient. Indeed, this quantifier can be specialised to specific types, for instance to the base type ζ, yielding $\forall\{\zeta\} : (\zeta \to \mathbf{t}) \to \mathbf{t}$, or even to properties of ζ objects, which are of type $\zeta \to \mathbf{t}$, yielding $\forall\{\zeta \to \mathbf{t}\} : ((\zeta \to \mathbf{t}) \to \mathbf{t}) \to \mathbf{t}$.

2.3 Application Revisited

The types with n sorts normally act as an implementation of selection restriction via application. The mechanisms of lexical semantics that convey additional meanings take place when the application encounter a type mismatch in a situation such as:

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

In that case, the analysis looks for some optional term, available either through the lexical entry of P, τ, or both, that can resolve the situation. It must be some $f^{U\to V}$, that is, the selection of this optional term is guided by the type system but not contained in type-derived rules.

If there is no suitable optional term available, the analysis ends with a type mismatch (or signals a missing operation, see section 4.4). If there is at least one available, it is used to *transform* the argument into a term that is suitable for the predicate, both yielding the expected type and providing a trace for the actual transformation in the meaning assembly phase. If there is more than one suitable transformation, multiple interpretations can occur.

In any case, after the transformation takes place, the type mismatch is corrected in the following way:

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

This leads to the semantic representation $P(f(\tau))$ after reduction.

2.4 Example

The optional terms are used to represent alternate lexical meanings induced from qualia, deverbals, or other alternations, which occur in the case of type mismatches in the usual meaning assembly. The following example is the lexical transformation from a city into the football club commonly associated, using the following (trimmed down) lexical entry:

$$\left\langle \text{Liverpool}^{City}\ ;\ Id = \lambda x^{City}.x\ ,\ f_C^{City\to Club} \right\rangle$$

Here, *Liverpool* is considered as being primarily the proper name for an entity of type *City*, with a transformation that can provide something of type *Club*, that is, the football club commonly associated. Considering:

(6) Liverpool won the cup.

For the sake of simplicity, this can be considered as a simple application of a predicate (*won*) to an argument (*Liverpool*). The predicate is the following:

$$\left\langle \lambda x^{Club}.(\text{won}^{Club \to t}\ x)\ ;\ Id = \lambda x^{Club}.x \right\rangle$$

The application is the following:

$$\lambda x^{Club}.(\text{won}^{(Club \to t}\ x)\ \text{Liverpool}^{City}$$

The type mismatch can be resolved using the optional term f_C, contained in the lexical entry for *Liverpool*, which gives the expected type. The application, using f_C as a transformation, yields:

$$(\text{won}^{Club \to t}\ (f_C^{City \to Club}\ \text{Liverpool}^{City}))$$

The representation in predicate logic is simply $\text{won}(f_C(\text{Liverpool}))$.

3 Recent Advances

3.1 Constraints

One of the main problems with the usual treatment of polysemous words is to distinguish between felicitous and infelicitous co-predications. Consider:

(7) a. The salmon was fast.
b. The salmon was delicious.
c. *The salmon was fast and delicious.
d. The salmon was lighting fast. It is delicious.

(8) a. Liverpool is a large city.
b. Liverpool voted Labour.
c. Liverpool won the cup.
d. Liverpool is a large city and voted Labour.
e. *Liverpool is a large city and won the cup.

Such data is subject to discussion and interpretation as to what is really felicitous or not. However, it has conducted us to provide a mechanism to distinguish between the word meanings (accessible by transformations) that are compatible with others, and those that are not.

We associate every optional term, every transformation and every term constructed in the process of the analysis with a *degree of flexibility* that allow to provide *constraints* associated with problematic transformations. Based on the behaviour seen above, transformation should at the least be distinguished between *rigid* and *flexible* ones. The flexibility of the terms is updated during the calculus and checked against the constraints in each application or conjunction.

As a first approximation, the difference is simply the following:

- Optional terms that are said to be *flexible* (*F*) are compatible with everything else. There is no restriction to their behaviour.
- Optional terms terms that are said to be *rigid* (*R*) are only compatible with themselves. No other transformation may occur to a term they have modified, nor can they apply to a term that has been modified by any other.

The distinction is made within the lexicon. It is also necessary to keep track of this during the composition.

For instance, (8e) would be treated in the following way. Considering a slightly expended entry for *Liverpool*:

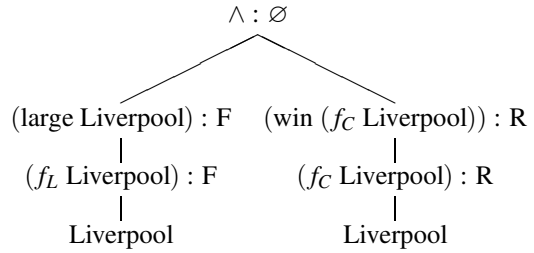
$$\left\langle \text{Liverpool}^{City} \; ; \; \underset{F}{Id = \lambda x^{City}.x} \; , \; \underset{F}{f_P^{City \to People}} \; , \; \underset{F}{f_L^{City \to Location}} \; , \; \underset{R}{f_C^{City \to Club}} \right\rangle$$

The identity transformation and the possibility to access either people or geographical location are flexible transformations (one can easily say (8d)), the access to the football club is rigid. Simplifying slightly, we can summarise our example as a conjunction of two predicates, one pertaining to locations, the other of football clubs:

$$\left\langle \lambda x^{Location}.(\text{large}^{Location \to t}\ x) \; ; \; \underset{F}{Id = \lambda x^{Location}.x} \right\rangle$$

$$\left\langle \lambda x^{Club}.(\text{win}^{Club \to t}\ x) \; ; \; \underset{F}{Id = \lambda x^{Club}.x} \right\rangle$$

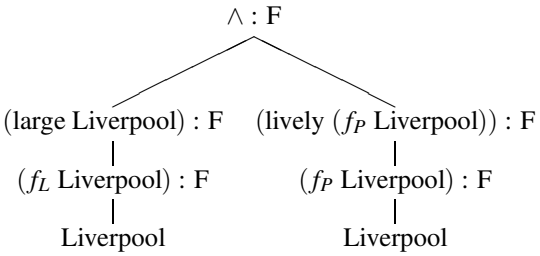
When parsing flexibility, it is impossible to apply simultaneously the flexible and rigid transformations:



On the other and, a *correct* co-predication such as (8d) is made possible by the use of flexible optional terms as transformations. With a predicate pertaining to *People*, such as:

$$\left\langle \lambda x^{People}.(\text{lively}^{People \to t}\ x) \; ; \; \underset{F}{Id = \lambda x^{People}.x} \right\rangle$$

The conjunction is acceptable because only flexible transformations are used:



The resolution of a type mismatch in a simple application is trivial. Co-predication, however, requires a conjunction between two predicates having different types as targets. This is accomplished by using a polymorphic version of *and*:

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

This operator takes the conjunction of two arbitrary predicates P and Q taking two different types α and β as their argument, and provides a guide for applying two possibly different optional terms as transformations for any argument of arbitrary type ξ. The application becomes the following:

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

$$/\ \text{large}^{Location\to\mathbf{t}}\ \text{lively}^{People\to\mathbf{t}}\ \{City\}\ \text{Liverpool}^{City}\ f_L^{City\to Location}\ f_P^{City\to People}$$

This reduces to:

$$(\text{and}^{\mathbf{t}\to\mathbf{t}\to\mathbf{t}}\ (\text{large}^{Location\to\mathbf{t}}\ (f_L^{City\to Location}\ \text{Liverpool}^{City}))\ (\text{lively}^{People\to\mathbf{t}}\ (f_P^{City\to People}\ \text{Liverpool}^{City}))$$

and yields the wanted representation in predicate calculus, of

$$\text{and}(\text{large}(f_L(\text{Liverpool})),\text{lively}(f_P(\text{Liverpool})))$$

3.2 Syntax-Driven Constraint Relaxing

However, some rigid transformations exhibit different behaviours when used in a longer syntactic unit than a sentence. Anaphoric reference can make sentences such as (7d) felicitous, while (7c) is not.

A first refinement would then be to distinguish between transformations that are *flexible* (of degree 1), *semi-flexible* (of degree 2), and *rigid* (of degree 3). The semi-flexible transformations would have the behaviour of rigid transformations, but their use would be destroyed by syntactic constructs such as anaphoric reference.

3.3 Articles and Quantifiers

The usual view in Montague Grammar (as seen in section 1.1) is that common nouns such as *book* are accurately modelled as predicates (the set of properties that means to be a book), and that articles are quantifiers that can point to individuals satisfying such predicates (and thus *a book* is an entity). The usual treatment, using existential, universal or generalised quantifiers is questionable in the standard Montagovian theory, and becomes very problematic when used with n sorts.

Hilbert's operators provide an interesting alternative to generalised quantifiers. Described in [8], ε and ι can be used to represent some aspects of determiners, and have been used in this fashion by von Heusinger in [7] and [26]. They can be used in order to model the existential quantification and dynamically linking.

In Montague Grammar, ε is of type $(\mathbf{e}\to\mathbf{t})\to\mathbf{e}$, making it a good choice for the semantics of the indefinite article *a*: if *book* is of type $(\mathbf{e}\to\mathbf{t})$, *a book* is of type $\mathbf{e}$. But our framework, based on principles borrowed from the Generative Lexicon, uses many types and *book* would be typed as $(Readable\to\mathbf{t})$, with *Readable* being an ontological type common to most printed works, and the work *book* providing optional terms in order to access the physical object associated and the informational content.

Our indefinite determiner is modelled as a polymorphic, many-sorted version of Hilbert's ε: a constant ε of type $\Lambda\alpha.(\alpha\to\mathbf{t})\to\alpha$. As the original operator, given a predicate with some arbitrary target types, it provides an individual of that type.

However, a different approach argued by Luo is that common nouns simply provide a base type; see section 4.2 for discussion.

3.4 Linear Formulation for more Flexible Constraints

The incompatibility of some facets is, up to now, accounted for by the rigidity of some transformations. Such transformations are exclusive: they exclude any other transformation. This is somehow too schematic and it is possible that the actual incompatibilities are more subtle than this. Assuming that there are four transformations associated with a word, f,g,h,k it is *a priori* possible — although we do not have examples of this yet — that the compatible subsets are $\{k\},\{f,g\},\{g,h\},\{h,f\}$ but not $\{f,g,h\}$, or even more sophisticated configurations.

Such explicit subsets might be formulated using a single transformation which produces the compatible subsets. If these were types, linear logic would easily express these possibilities, e.g. with $!k\&(!f\otimes!g)\&(!g\otimes!h)\ \&(!h\otimes!f)$ or the alternative using $!a\otimes!b\equiv!(a\&b)$ — second order intuitionnistic linear logic faithfully encodes system-F, as studied in [25]. We are presently studying whether the proof terms of [4] for intutitionnistic linear logic could be used in the same manner, with transformations as constants, i.e. as black boxes of the corresponding types.

This seems to be a promising direction for handling subtle incompatibilities — and at the same time we are looking for linguistic data with such tricky incompatibilities.

4 Discussions

4.1 Transitivity

The issue of the transitivity of transformations is a different issue that of constraints that can apply to them. Consider:

(9) a. Liverpool won the cup.
b. It had hired several oversea players.
c. This caused it to go bankrupt.
d. *Liverpool was thus forced to resign.

In (9a), *Liverpool* is used in the sense of the football club, via a first transformation. The *club* itself receives a second transformation in (9b) (despite the first transformation being rigid: this is not a case of conflict between several senses, but of sequential changes), that makes accessible some unnamed person responsible for the management of the club. This can carry for a bit, but there are clearly limits, as seen in (9d).

We can remark that idioms (that is, linguistic differences) can make some transformations possible:

(10) a. My car has a punctured tyre.
b. *My car is punctured.
c. Ma voiture est crevée.

If (10a) is possible, (10b) should be under the assumptions of [21] that a *tyre* is an important constitutive part of a *car*. It is not, but the French (10c) is, which is a literal translation of (10b). This illustrate the importance of making the lexical transformations on a linguistical rather than ontological basis.

However, consider:

(11) a. My car has four injectors.
b. One of them is clogged.
c. ? My car is clogged.

(11a) followed by (11b) is felicitous. However, in the context of (11a), (11c) can be understood, if a bit strange, but it is clearly infelicitous by itself.

Keeping track of successive transformations and what can be ultimately derived from a word is thus as necessary as the evaluation of what transformations are possible simultaneously.

4.2 Types, Predicates and Granularity

There are two closely related issues with regard to the design of the type system. The first is wether nouns should be represented as being individuals of some base type or predicates, the second is the definition of the set of base types. There are several possibilities for the latter:

- The only sort is **e**. This preclude any type-based lexical treatment.
- There is a finite, small number of sorts that serve as base types, derived from ontological properties, as envisioned by [21] and [2] (e.g. $Physical$, $Information$, $Agent$, $Artifact\ldots$ All told, two or three dozen at most).
- There is a finite and relatively small number of sorts that are defined by the common restrictions of selection gathered from the grammar of a language (*animate*, *human*, *male*... about 200 in total).
- There is a finite but large number of base types: one for every common noun, and many other derivable Σ-types ([10]).
- Every possible formula with a free variable is a type. This could lead to a circular definition (as types are used to define formulae).

For some of these sets of base types, having nouns as individuals rather than predicates is a sound approach, specifically in the case where every noun defines its own type. Our view is that for some finite and reasonable number of base types, the two approaches can be non-contradictory.

It is possible to envision both possibilities simultaneously, by associating to every base type α the predicate *to be of type* α, written $\hat{\alpha}$, which is of type $\mathbf{e} \rightarrow \mathbf{t}$. In that case, it is possible to quantify over individuals of a given type using this construct and the polymorphic ε given in section 3.3.

4.3 Locality and Contextuality

There are many cases where differences between social contexts or speakers can result in different lexica. The most obvious example is in fictional narratives such as mythology, fables, fairy tales, fantasy or science-fiction, where a great part of the vocabulary is attached to the narrative. Some of these works make use of standard vocabulary but change some of the settings (for instance, children's literature makes extensive use of animals as characters, effectively changing their type to *Agent*), some introduce specific words either by amalgamation (*direwolf*, *lightsaber*, etc.) or similarity with historical words (*serjeant*), and some use words that are not related to the reader's lexicon and have him infer their meaning purely from context (Tolkien's Quenda and Sindar, as most readers will not get around to perusing an Elvish dictionary).

Each of those literary contexts modify the lexicon as part as the normal reading process of the work associated. We believe that Fantasy, SF and tales can be modelled as having a slightly *expanded* lexicon, with additional words available and some additional senses available for some words. There are many words that are not used, but they are ignored (as in any other case where the full lexicon is not used for a particular text).

The idea here is that additions to the lexicon can be made locally in the following cases:

- There is a known difference in the literary genre.
- There is an apparent difference in the vocabulary used by different speakers in a dialogue.
- There is a social setting that provides additional terms, such as those used in a trade *jargon*.

Examples – all of the following are from G. R. Martin's *A Dance with Dragons*.

- The point of view is of particular importance. Even outside of dialogue, where the narration is third-person, a specific agent is used, explicitly (Robbin Hobb, George Martin...) or implicitly, for providing atmospheric effects and involving descriptions:

  (12) But the air was sharp and cold and full of fear.

- The text will often use additional words, that do not belong in any language. Here is an explicit example, that include the necessary information for the reader's lexicon:

  (13) Their driver awaited them beside his *hathay*. In Westeros, it might have been called an oxcart, though it [...] lacked an ox. The *hathay* was pulled by a dwarf elephant[...]

- However, the mechanisms of lexical semantics apply equally as well in such a setting. For instance, organisations such as banks continue to have their multiple aspects available, including an unnamed human leader:

  (14) We who serve the Iron Bank face death full as often as you who serve the Iron Throne.

In each case, a slightly expanded and differentiated lexicon would be needed.

4.4 Infelicitous statements

In the view of such examples and many others where specific senses are associated to words in a technical jargon (skeumorphic metaphors in the computer world such as *file*, *document*, *directory*, *window*..., for instance, became mainstream with the technology), there is a danger in taking the view that infelicitous sentences are simply wrongly formed. In our view, the analyser should not necessarily reject sentences that are infelicitous because of a type mismatch with no suitable transformation available, but instead indicate that there must exist such a transformation that is missing from the lexicon, and try to infer its characteristics.

4.5 Other approaches

There have been several other approaches to the modelling of a complete framework for the Generative Lexicon. However, the first ones (including the original formulation by James Pustejovsky and early work by Nicholas Asher) neglected large parts of compositional semantics based on the syntactic structure, including determiners, quantifiers, plurals... Our own approach, as well as the more recent others, have focused on the integration of this aspect.

Asher – The first formulation was arguably [22], which was continued by Nicholas Asher in several works that culminate in [2]. The most advanced of those approaches use *type presuppositions* as a guide for the computation of meaning. The formal aspect of the computation of complex types, however, is still a problem, as is their interpretation in term of categorical fiber-products.

Cooper – In [5], Robin Cooper proposes a complete model that uses record types in Per Martin-Löf's type theory, as well as generalised dynamic quantifiers. We are continuing discussions regarding this approach and possible convergences.

Luo – Luo Zhaohui has been working with Nicholas Asher and has proposed a slightly different logic based on Unified Type Theory, in [27], and recently [9]. His principles are to consider words as types and transformations as coercions in a paradigm of coercitive subtyping. Discussions have proven interesting with regards to many aspects of the logic and type system.

Conclusion

The work presented here is still in progress. We are still looking for a satisfactory implementation of constraints using types in linear logic, and for an account of the limits of transitivity. There are three main areas for future research:

- Exploration of additional phenomena, including nonce words and hypostasis, the relativity of the lexicon to social contexts and agents. Such research is currently undertaken by Cooper.
- Fine-tuning the formal framework, including a formal approach of quantification and syntax-driven composition, as Asher and Luo are doing.
- Implementing and testing a complete framework.

We are making some progress in all three of these areas. We are investigating alternate literary universes. We have made some real progress from the original formulation on the formal compositional model, with accepted publications forthcoming on the use of Hilbert's ε for indefinite articles and on plurals ([16]). We also have a partial implementation in λ-DRT, using Richard Moot's Grail analyser for syntax and semantics ([15]), including applications to the virtual traveler problem ([23]), among others. Much is still missing, however, especially work on the automatic acquisition of rich, well-typed lexica. Progress in that direction remains dependent on further formalisation on the granularity of the type system, and we are optimistic.

References

[1] Nicholas Asher. A Type Driven Theory of Predication with Complex Types. *Fundamenta Informaticæ*, 84(2):151–183, 2008.

[2] Nicholas Asher. *Lexical Meaning in Context: a Web of Words*. Cambridge University Press, March 2011.

[3] Christian Bassac, Bruno Mery, and Christian Retoré. Towards a Type-Theoretical Account of Lexical Semantics. *Journal of Language, Logic, and Information*, 19(2), 2010.

[4] P. N. Benton, Gavin M. Bierman, Valeria de Paiva, and Martin Hyland. A term calculus for intuitionistic linear logic. In Marc Bezem and Jan Friso Groote, editors, *TLCA*, volume 664 of *Lecture Notes in Computer Science*, pages 75–90. Springer, 1993.

[5] Robin Cooper. Copredication, dynamic generalized quantification and lexical innovation by coercion. In *Fourth International Workshop on Generative Approaches to the Lexicon*, 2007.

[6] Ann Copestake and Ted Briscoe. Lexical operations in a unification based framework. In *ACL SIGLEX Workshop on Lexical Semantics and Knowledge Representation*, 1991.

[7] U. Egli and K. von Heusinger. The epsilon operator and e-types pronouns. *Lexical Knowledge in the Organization of Language*, pages 121–141, 1995.

[8] D. Hilbert and P. Bernays. *Grundlagen der Matematik. Bd. 2*. Springer, 1939.

[9] Zhaohui Luo. Contextual analysis of word meanings in type-theoretical semantics. In *Pogodalla and Prost*, pages 159–174. 2011.

[10] Zhaohui Luo. Common nouns as types. In *BÃ©chet and Dikovsky*, pages 173–185. 2012.

[11] Bruno Mery. Compositionality and the Lexicon (Lexique organisé pour la composition sémantique). In *Journées Sémantique et Modélisation*, Paris, France, 2009.

[12] Bruno Mery, Christian Bassac, and Christian Retoré. A montagovian generative lexicon. In *Formal Grammar*, 2007.

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

[14] Richard Montague. The proper treatment of quantification in ordinary English. In R. H. Thomson, editor, *Formal Philosophy*, pages 188–221. Yale University Press, New Haven Connecticut, 1974.

[15] Richard Moot. Wide-coverage French syntax and semantics using Grail. In *Proceedings of Traitement Automatique des Langues Naturelles (TALN*, Montreal, 2010.

[16] Richard Moot and Christian RetorÃ©. Second order lambda calculus for meaning assembly: on the logical syntax of plurals. In *Coconat*, Tilburg, Netherlands, 2011.

[17] Julius M. Moravcsik. How do words get their meanings ? *The Journal of Philosophy*, LXXVIII(1), January 1982.

[18] Reinhard Muskens. Meaning and Partiality. In Robin Cooper and Maarten de Rijke, editors, *Studies in Logic, Langage and Information*. CSLI, 1996.

[19] J. Pustejovsky. Type construction and the logic of concepts, 2001.

[20] James Pustejovsky. The generative lexicon. *Computational Linguistics*, 17(4):409–441, 1991.

[21] James Pustejovsky. *The Generative Lexicon*. MIT Press, 1995.

[22] James Pustejovsky and Nicolas Asher. The Metaphysics of Words in Context. *Objectual attitudes, Linguistics and Philosophy*, 23:141–183, February 2000.

[23] L. PrÃ©vot R. Moot and C. RetorÃ©. Un calcul de termes typÃ©s pour la pragmatique lexicale – chemins et voyageurs fictifs dans un corpus de rÃ©cits de voyages. In *Traitement Automatique du Langage Naturel – TALN 2011*, pages 161–166, Montpellier, France, 2011.

[24] L.-M. Real-Coelho and C. RetorÃ©. A generative montagovian lexicon for polysemous deverbal nouns. In *4th World Congress and School on Universal Logic – Workshop on Logic and Linguistics*, Rio de Janeiro, 2013.

[25] C. Retoré, Christian. Le système F en logique linéaire. Mémoire de D.E.A. (dir.: J.-y. girard), Université Paris 7, 1987.

[26] K. von Heusinger. Choice functions and the anaphoric semantics of definite nps. *Research on Language and Computation*, 2:309–329, 2004.

[27] T. Xue and Z. Luo. Dot-types and their implementation. In *BÃ©chet and Dikovsky (2012), pages 234-249*, 2012.