Lexical Semantics with Linear Types

Bruno Mery

Lexical Semantics with Linear Types

Bruno Mery[1]

IUT de Bordeaux, Université de Bordeaux, France
LaBRI — CNRS
bruno.mery@me.com

Abstract

We have proposed a framework based upon the λ-calculus with higher-order Intuitionist types for the symbolic computation of the semantic analysis, integrating lexical data. This proposal is sufficient for many phenomena and accurately incorporates lexical semantics by the means of type theory, but some issues linger in the linguistic data. In the present paper, we revisit this proposal with a version of the λ-calculus based upon higher-order linear types, that aims to resolve those issues and present an integrated framework for meaning assembly.[1]

1 Introduction

In the computation of Natural Language Semantics for use in NLP applications, there are two main approaches being pursued : the symbolic, compositional approach that is derived from the Montagovian program (from [Montague, 1974]), in which the meaning of the utterance is derived from the combination of single meaningful postulates associated with each word (most often by the means of β-reduction in a version of the λ-calculus), and the numeric, statistical approach that use contextual information and rules acquired from corpora (using strategies similar to hidden Markov models). It is generally held that the former is closer to the structure of the sentence and able to process correctly complex issues of binding, anaphora, and quantification scope, and that the latter is more informative as it is able to derive precise senses for the words, disambiguating between different meanings that can be conveyed by the same lexical item. There are also many hybrid approaches that try to combine the two, but statistical methods and heuristics are prevalent in most of the real-world applications.

Another approach that has gained traction in the theoretical circles is the integration of complex lexical data and compositional rules in the symbolic analysis. The principle is that words are no longer atomic lexical items, but provide enough information in order to influence their context, and be influenced by other lexical items in return. The use of *co-compositionality* means that the assembly of the meaning of an utterance from its constituting lexical items is more complex than simple application, but, in many cases, the precise determination of the sense attached to polysemic words can be done in a straightforward, deterministic way. Statistical methods, and specifically semi-automated learning, are still useful, but are restricted to the construction of such a complex lexicon.

This alternative approach, pioneered by Nunberg and Sag (see eg. [Nunberg, 1993]) and Pustejovsky (in [Pustejovsky, 1991]), has come to be known as the Generative Lexicon Theory. Thoroughly developed by Pustejosvky in [Pustejovsky, 1995], criticised and enriched by many linguists since then, it has become a reference in linguistic theories. There have been many early attempts to use the principles of the generative lexicon in symbolic semantic analyses for NLP applications (such as

[1]The present paper presents work in progress by the author, currently employed at a full-time teaching position in higher education, and thus still preliminary. This work relates to ΛTY_n, a framework developed at LaBRI and LIRMM by many different people including Christian Bassac, Richard Moot and Christian Retoré from 2007 on. As a research team, we would like to thank Nicholas Asher, Daisuke Bekki, Stergios Chatzikyriakidis, Robin Cooper, Thomas Icard, Luo Zhaohui, Koji Mineshima and many others from long and helpful discussions and collaborations. We are indebted to Frédéric Lang, Anaïs Lefeuvre and Livi Real for working with us on this project. The author would like to extend his gratitude to the NLCS organising committee and the reviewers of the present publication for their kind and helpful comments and the opportunity to return to NLCS.

[Pustejovsky and Asher, 2000], [Gupta and Aha, 2005] and [Pinkal and Kohlhase, 2000]), but the sheer complexity of the concepts involved prevented really satisfactory analyses for a long while.

More recently, there have been several proposals that use "New" Type Theories (logical frameworks of higher order, in the tradition of Curry) in order to accurately integrate lexical semantics (and some additional phenomena) in their analysis. They comprise the Type Theory with Records ([Cooper, 2007]), Dependant Type Theory ([Bekki and Asher, 2012]), Type Presuppositions ([Asher, 2011]), Unified Type Theory ([Luo et al., 2013]), and our very own λ-calculus with higher-order types and many sorts ([Retoré, 2014]). All these proposals use a form of higher-order typing, binding or quantification in order to provide mechanisms for *co-composition*, *sub-typing* and *type coercions*.

Concerns of complexity and decidability have been raised over the use of higher-order logics. Partial implementations and illustrations of the constraints in the use of the expressive power of such systems have alleviated some of those; however, the formal characterisation of the fragment actually used remains a work in progress. Other problematic points include the coverage of such systems (the treatment of additional phenomena is detailed on a regular basis), and the use of features outside of the formal systems in order to patch in some specific idiosyncrasies.

The present paper aims at addressing some of these concerns by proposing an alternative to ΛTY_n that uses a λ-calculus with higher-order Intuitionist linear types. We will demonstrate that this approach is adequate to model the variations in the felicity of co-predicative utterances, and does not imply the use of external features. It also provides a single, well-known formal setting for meaning assembly.

2 Our Type-Theoretical Framework for Lexical Semantics

We will first detail ΛTY_n as a theory of meaning composition, in the tradition of Montague.

2.1 Principles of the Approach

In constructing ΛTY_n, we aimed at producing a system that integrates as many features of lexical semantics as possible while remaining useable, i. e. sufficiently similar to recent re-interpretations of Montagovian semantics to be plugged into our text analysis workflow. Such a system should be sufficiently flexible to keep up to date with the evolution of linguistic studies, and to add support for various phenomena.

We thus adopted a compositional paradigm in which meaning is computed along the lines of the syntactic structure of the utterance. More precisely, composition in Montague-style semantics proceeds as follows : words are given syntactic types (according to syntactic and grammatical categories), and are assembled using a grammar-like formalism such as A/B grammars. This is the first analysis where typed terms are combined in order to form well-typed sentences that yield the syntactic structure. The "meaning" derived from this first step of calculus is that structure, generally depicted as a tree (or sometimes forest).

The second step is *meaning assembly*, wherein the semantics is calculated. The lexicon provides semantic types and terms associated to words (or lexemes), and the Curry-Howard isomorphism states that there is a correspondence between syntactic and semantic types, up to a point (and up to the integration of designs between the formalisms used). The syntactic structure is used to seed the basic layout of the semantics terms in a compositional way, i. e. every term is the argument or predicate of a term that corresponds to the words that shared the corresponding syntactic relationship in the actual utterance.

Meaning assembly consists of the computation, by means of function application, of the formulae given by that transcription. Semantic types are (propositional, Intuitionist) formulae, terms being proofs of those formulae, and the Curry-Howard isomorphism ensuring that everything follows smoothly. At the point of meaning assembly then, there are two notions of formulae : the propositional ones corresponding to the types of (individual or composed) terms, and the result of the combination of the terms themselves.

The latter formulae, when in normal closed form, as a result of β-reduction (or cut-elimination at the proof-theoretic level) can be interpreted as the meaning as the semantics analysis, usually in its first-order predicate logic form. That representation can, in its turn, be interpreted in a model-theoretic way if needed.

The Montagovian analysis uses **e** and **t** as base types. In order to account for the many distinctions driving selection restrictions, we adopt a many-sorted logic, i. e. a logic with many base types in addition to **t** that are effectively used to represent different kinds of **e** (entities). Our principles are, as Pustejovsky suggested, that the types given to lexemes in the lexicon guide the compositional process, and that some *creative use of words* is used to create new meaning in event of a type clash.

2.2 Main Features

Work on this framework is continuing (see e. g. [Bassac et al., 2010], [Mery, 2011], [Mery and Retoré, 2013] and most recently [Retoré, 2014] and [Mery et al., 2015a]).

As it stands now, the logic used is the second order Intuitionist logic based on the logic of types with n sorts described in [Muskens, 1996]. The"second-order" aspect symbolised by the Λ represents the ability to use type-level abstraction, application and quantification, and is based on Jean-Yves Girard's System-F (see [Girard, 1972]). Our calculus for meaning assembly is the λ-calculus with Intuitionist, second-order types of many sorts, while our final logic used for representations is the many-sorted higher-order predicate calculus.

The difference with basic semantic analysis that encompasses the main contribution of lexical semantics to the construction of meaning can be summarised as follows: to lexemes are associated as the meaning-assembly start not a single typed term, but a collection of such: a single "main" term with associated type, and zero to finitely many additional ones. We use these additional terms in case of a type mismatch to provide transformations to the terms in the meaning assembly phase, and the resulting logical representation incorporates those transformations as modifiers to the meaning of the terms.

More specifically, the transformations are used to resolve *type mismatch* situations such as the following:

(1) The book is heavy.

(2) The book is interesting.

No matter what the type for *book* is, *heavy* is a predicate restricted to arguments that are physical objects while *interesting* is restricted to information (or facts). Lexemes such as those that have multiple facets are called *dot objects* in the Pustejovskian tradition, and have provided some of the most difficult phenomena to model in a logically sound way. We simply have a type *Book*, a type φ and a type $Info$ that are three different sorts, and the lexical entry for *book* provides, in addition to the expected main λ-term, the following two additional ones: ${f_\varphi}^{Book\to\varphi}$, that turns terms of type *Book* into terms of type φ, and ${f_{Info}}^{Book\to Info}$, that turns terms of type *Book* into terms of type $Info$.

In the above examples, composition proceeds as normal until a type clash occurs. In example (1), it occurs between $(\text{the book})^{Book}$ and $\text{is\_heavy}^{\varphi\to\mathbf{t}}$. However, the optional term f_φ, made available by the lexical entry for *book*, can be used to resolve the type mismatch in the following way : $(\text{is\_heavy}\ (f_\varphi\ (\text{the book})))$, which has a felicitous typing and yields semantics consistant with our expectations[2].

This very simple mechanism allows us to take into account a number of different linguistic phenomena.

The main difference between our approach and those of Asher, Bekki, Luo… is that the type-changes transformations that are called *coercions* in most of the literature are, almost always, given as relations between *types*. Our transformations are optional terms given in the lexicon, and thus are introduced by individual *words*. This is a system guided by *types*, but enabled by terms at the lexical level, which allows us to model specific idiosyncrasies and hypostasis, among other advantages.

However, a case where type-driven systematic coercions is useful is the derivation of *ontological relations* such as hyperonymy. Consider the following:

(3) The Honda moved.

(4) The car moved.

(5) The vehicle moved.

(6) The object moved.

(7) The thing moved.

Clearly, the meaning of (7) is implied in (6), while (5) is more precise than (6), etc. Lexical semantics is often based upon an *ontological hierarchy* of types, a tree, forest or lattice consisting of *sub-typing* relations. (There is some disagreement between different type-theoretical views here, depending on the typing of the argument and of the instantiation strategy used in the semantics of the determiner; see further on.)

Pustejovsky calls examples (3-7) instances of *type accomodation*: *Honda* is a subtype of *Car* which is a subtype of *Vehicle* which is a subtype of φ which is a subtype of $\mathbf{e}$, and any subtype may be used anywhere the original type can be used (again, with subtle issues depending on determiner and DP semantics). This is formalised as a *coercive mechanism*, as rules at the logical level, in [Soloviev and Luo, 2001] and [Luo et al., 2013]. Briefly, having a type A subtype of B is written $A \preccurlyeq_c B$ and means that there is some coercion $c^{A\to B}$ turning objects of type A into objects of type B. As detailed in [Retoré, 2014], we can use the same mechanism to introduce type-driven coercions for ontological sub-typing, while keeping our term-driven framework for all other transformations.

A specific issue being disputed in recent type-theoretical developments is that of the typing of common nouns. In the classical Montagovian analysis, nouns are *predicates*, i. e. typed as $\text{cat}^{\mathbf{e}\to\mathbf{t}}$. Nouns represent the property of being an individual represented by such a predicate, and determiners select the appropriate individual, with a typing such as $a^{(\mathbf{e}\to\mathbf{t})\to\mathbf{t}}$. The combination is a determiner phrase of type $\mathbf{e}$.

Luo, among others, departs from this view by having common nouns typed with atomic types. Thus he has cat^{Animal} and $a^{\alpha\to\alpha}$ for any sort α; the common noun represents a generic individual, which is determined by the determiner.

[2] The specific question of quantification and the interpretation of determiners is not detailed here; we have adapted selection operators from Russell and Hilbert. See [Retoré, 2014].

We argue that having nouns typed as predicates is very useful for determination and quantification purposes, but that having access to the individual is also necessary in other cases (adjectives, adverbs...). We use operators derived from Hilbert's ε for that purpose. We associate a characteristic predicate to all sorts, $\widehat{\alpha}^{\mathbf{e}\to\mathbf{t}}$ where α is the sort, and axiomatise for all such predicates that $\widehat{\alpha}(\iota(\widehat{\alpha}))$. In the end, determiner phrases are typed by a single sort. We have presented a complete account of this system founded on type-theoretic principles in [Mery et al., 2015b].

2.3 Compositionality

The compositional approach to formal semantics, in agreement with cognitive evidence, advocates that the meaning of the discourse is derived from the composition of the meaning of its parts. Crucially, at each composition step from the elementary lexical term (lexeme) to the final formula in normal form (logical representation of the utterance / discourse), each term can still be interpreted and convey constructive meaning. While this is transparent in classical Montague semantics, it is also the case with our additional mechanisms. Here, we will only present an intuitive interpretation of the denotation of adaptative predications.

An intransitive verb such as *to fall* is modelled as a predicate over a sort (physical objects) that can be adapted to other argument types: $\Lambda\alpha\lambda x^{\alpha} f^{\alpha\to\varphi}.(P\ (f\ x))$. The meaning associated to P is that of asserting that x, if it can be coerced into a physical object by some f, enjoys P. Such predicates always provide the identity morphism for the appropriate type, here $id_{\varpi} = \lambda x^{\varphi}.x$, and so the "usual" meaning of the verb can be inferred as $\lambda x^{\varphi}.(P\,x)$. In *The book fell*, the argument will provide the correct transformation in order to infer correctly that the verb applies to the physical aspect of the book. Thus, compositionality and intermediate meanings are preserved through the process of meaning assembly.

This approach is also compatible with accounts of semantics that are centred on context, as with dynamic or continuation semantics. The "slots" provided for transformations allow the context to alter the felicity of predications such as * *The idea fell*, depending if some morphism of type $Abstract \to \varphi$ or similar has been introduced in the discourse.

However, the above mechanisms are not sufficient in order to treat all phenomena, as issues with co-predicative utterances illustrate.

3 Co-Predicative Utterances

Co-predicative utterances, or *co-predications* for short, are phrases in which a polysemous word is used for more that one of its meanings at once. They are among the more difficult problems treated with the help of lexical semantics, and have been used as a test case for such theories.

3.1 The Problem of Co-Predication

Consider the following sentences (classical examples used by Asher, Pustejovsky and us in the past):

(8) The book is heavy and interesting.

(9) London is a lively, sprawling city.

(10) (?) The salmon is lightning-fast and delicious.

All these sentences contain a co-predication: the book in (8) is used by a predicate restricted to physical and another restricted to informative objects, London in (9) has its facets of a group of people and a place with geographical limits being used, while the salmon in (10) is used infelicitously at the same time as an animal and cooked food.

In classical Montagovian analysis, there are no ways to distinguish co-predications. Everything being typed as **e**, there are no selection restrictions. Even when a separate mechanism is used for selection restriction, when the individual predications are fine, their conjunction is also.

In a many-sorted framework, without a specific attention to co-predications, every co-predication is infelicitous, as the two predicates use different types as their arguments.

The difficulty lies in producing a mechanism that allows (8-9), but is able to block (10).

3.2 Co-Predications in our Current System

We treat the co-predication issue using a specific operator for conjunction, the polymorphic "and" defined as follows:

$$\Lambda\alpha\beta\tau\lambda P^{\alpha\to\mathbf{t}} Q^{\beta\to\mathbf{t}} x^{\tau} f^{\tau\to\alpha} g^{\tau\to\beta}.(\wedge^{\mathbf{t}\to\mathbf{t}\to\mathbf{t}}\ (P\ (f\ x))\ (Q\ (g\ x)))$$

This operator allows the conjunction of predicates bearing on different sorts to the same argument, if there are transformations that can accommodate both predicates.

Example (8) is thus computed as

$$\text{AND is\_heavy}^{\varphi\to\mathbf{t}}\ \text{is\_interesting}^{Info\to\mathbf{t}}\ (\text{the book})^{Book} f_{\varphi}{}^{Book\to\varphi} f_{Info}{}^{Book\to Info}$$

which reduces to

$$(\wedge^{\mathbf{t}\to\mathbf{t}\to\mathbf{t}}\ (\text{is\_heavy}\ (f_{\varphi}\ (\text{the book})))\ (\text{is\_interesting}\ (f_{Info}\ (\text{the book}))))$$

In addition, in order to deal with incompatible co-predications, we have a mechanism in the lexicon that attaches to every transformation a label (either *Flexible* or *Rigid*). *Rigid* transformations are destructive transformations, such as the ones in (10) that turns a salmon into food through the process of cooking. A simple set of rules ensures that such examples are not computed. However, this is rather unsatisfactory, as those features and rules are outside of the scope of the logical system itself.

3.3 Outstanding Issues

This framework is sufficient to deal with most co-predication issues, but not all. First, the use of an external system to reject infelicitous co-predications is unfortunate and inelegant.

More importantly, it also fails to predict felicity in many cases. Sentences such as (10) are clearly infelicitous, but can be rewritten as:

(11) The salmon was lightning-fast. It is delicious.

Thus, it appears that rather simple syntactic means (here, anaphoric reference) can alter the felicity of co-predications. There is also the example of city names, apparent in utterances such as:

(12) London is a huge city.

(13) London has castigated Moscow.

(14) (?) London is a huge city. They have castigated Moscow.

(15) (*) London is a huge, Moscow-castigating city.

City names have an extremely high degree of polysemy, and can refer to a number of different things depending on context. Co-predications between any two of those different meanings have a felicity which is variable in function of many parameters. A more complete account of the possible combinations is forthcoming (and outside the scope of this article), but this illustrates the difficulty of characterising coherent behaviours for co-predications.

Moreover, those examples necessarily rely upon the subjective appreciation of the authors and the limited data available. They are contestable, and might be idiosyncratic (personal communications from Asher and Luo, as well as anecdotical evidence, suggest significant differences in behaviour between English and Chinese). While a thorough empirical study of those issues, including a survey of native speakers of several languages, is worthy to conduct, we do not need to wait such in order to complete a framework that can provide such behaviours, while being adaptive enough to integrate updated linguistic data.

4 Linear Second-Order Types

4.1 Linear Logic

Linear Logic is an alternative to Classic and Intuitionist Logics developed by Jean-Yves Girard in [Girard, 1987], and detailed in [Girard, 1995]. It has garnered much attention in the field of formal logic for its simplicity, its geometrical interpretation and its relevance in category theory.

It differs from other logics in that hypothesises are used *exactly once* in order to form the conclusion. Where, in Classical Logic, from a proof of A and one of $A \to B$, one can deduce a proof of B and still be able to use the hypothesises for the rest of the deduction, in Linear Logic, if from a proof of A and one of $A \multimap B$, one deduces a proof of B, neither A nor $A \multimap B$ can be used afterwards. The possibility of duplicating some elements is constrained to exponent operators such as !, and thus non-linearity is clearly marked in the logic.

In computer science, Linear Logic is interesting because of the intuitive *resource interpretation* associated: rather than being abstract propositions that have some truth value in an eidetic space, the base elements of the logic can be thought of as *resources*. An atom A is a single instance of a resource. The linear implication $A \multimap B$ represent the ability to convert an instance of A into an instance of B, consuming both the resource and the converter in the process. Abundance of a resource is noted $!A$, etc. This is obviously well-suited to many real-world applications.

We will not delve into much detail for the syntax and semantics of Linear Logic (see [Braüner, 1996] for an introduction), but will consider Intuitionist Linear Logic, a fragment thereof that employs the following connectives:

- $\multimap$: linear implication ($A \multimap B$ consumes an A and yields a B).
- $\otimes$: multiplicative conjunction ($A \otimes B$ means that both A and B have to be used).
- &: additive conjucntion ($A\&B$ means that either A or B can be used, the formula that consumes this can ”choose” which).
- !: exponential ($!A$ means as many and as little instances of A as needed).

4.2 Computational Interpretation

Although Linear Logic is often used for its geometrical or proof-net interpretations, it is possible to define computational interpretations for it. The seminal work of Samson Abramsky ([Abramsky, 1993]) has illustrated this possibility, detailing terms, operational semantics and evaluation strategies for a computational interpretation of Intuitionist, second-order Intuitionist and classical Linear Logic.

Specifically, it is possible to define a version of λ-calculus where the terms are typed with formulae from Linear Logic rather that Intuitionist Logic.

Standard λ-calculus is the computational interpretation of the Curry-Howard Correspondence for formulae of Intuitionist Logic. Abramsky introduced such a calculus for Linear Logic (and several of its fragments), which was later studied and refined in e. g. [Benton et al., 1993]. The term calculus in itself is quite similar to standard λ-calculus, with abstraction and application unchanged. It presents a direct implementation process for linear functional programming.

4.3 Our Proposal for Linear Second-Order Types

The diverse possibilities of compatibilities for co-predicative utterances, explored above in section 3.3, has led us to consider using the improved power provided by Linear Logic, rather than a simple fragment of Intuitionist logic for types.

To be more precise, we want to propose a system based on Linear Second-Order ΛTY_n for meaning assembly, i. e. a version of λ-calculus that uses linear, Intuitionist, second-order, many-sorted types. Meaning assembly proceeds as it always has but constrained by linear types, and the features specific to linear logic will disappear after composition so that the (normal, closed) resulting term can be transcribed as a formula in regular predicate logic (with Intuitionist types).

Linear Logic is well-suited to the meaning assembly phase. In particular, the intuitive resource interpretation fits well with the imagined process of composition, where the meaning of arguments and predicates are consumed to form the semantics of the complex predication as a whole. In fact, simple composition for simple predications are easily transcribed using linear types in the usual fashion:

(16) The dog barked.

(17) (*) The chair barked.

(18) The sergeant barked.

With *to bark* (non-transitive version) being a predicate selecting arguments of type *Dog*, *the dog* a determiner phrase of that type, *the chair* of type φ and *the sergeant* of type $Human$, the terms for every sentence are:

- (16): $\Lambda\tau\lambda x^{\tau} f^{\tau\multimap Dog}.(\text{bark}^{Dog\multimap \mathbf{t}}\ (f\ x))\quad \text{the\_dog}^{Dog}$
  This reduces to $(\text{bark}\ (id_{Dog}{}^{Dog\multimap Dog}\text{the\_dog}))$, or more simply (bark the_dog).
- (17): $\Lambda\tau\lambda x^{\tau} f^{\tau\multimap Dog}.(\text{bark}^{Dog\multimap \mathbf{t}}\ (f\ x))\quad \text{the\_chair}^{\varphi}$
  This has no correct typing, assuming (as in a normal context) that there is no way to turn (generally inanimate) physical objects into dogs. (Dogs can be envisioned as physical objects, not the other way around.)
- (18): $\Lambda\tau\lambda x^{\tau} f^{\tau\multimap Dog}.(\text{bark}^{Dog\multimap \mathbf{t}}\ (f\ x))\quad \text{the\_sergeant}^{Human}$
  To reduce this, there has to be a way to consider humans as dogs. In this specific case, the lexical entry for *sergeant* has such a term available, the term $f_{Order}{}^{Human\multimap Dog}$ that implies that the delivery of a specific kind of speech has a certain dog-like quality. We thus have $(\text{bark}\ (f_{Order}\text{the\_sergeant}))$ as a felicitous result.

Thus, simple predicates, transformations and coercions all share a type being a single direct linear implication.

In fact, non-linear composition only occurs in cases of co-predication and, in general, of conjunction.

Our polymorphic conjunction has to duplicate the argument. A first step is simply this:

$$\Lambda\alpha\beta\tau\lambda P^{\alpha\multimap\mathbf{t}}Q^{\beta\multimap\mathbf{t}}x^{!\tau}f^{\tau\multimap\alpha}g^{\tau\multimap\beta}.(\texttt{copy}\ x\ \texttt{as}\ y,z\ \texttt{in}\ (\wedge^{\mathbf{t}\multimap\mathbf{t}\to\mathbf{t}}\ (P\ (f\ y))\ (Q\ (g\ z))))$$

This is the original notation proposed by Abramsky. For convenience's sake, this can be abbreviated as:

$$\Lambda\alpha\beta\tau\lambda^{0}P^{\alpha\multimap\mathbf{t}}Q^{\beta\multimap\mathbf{t}}f^{\tau\multimap\alpha}g^{\tau\multimap\beta}\lambda x^{\tau}.(\wedge^{\mathbf{t}\multimap\mathbf{t}\to\mathbf{t}}\ (P\ (f\ x))\ (Q\ (g\ x)))$$

In this notation, λ^{0} introduces linear terms and λ Intuitionist non-linear terms that can be duplicated in the formula.

We therefore use non-linear terms (implemented by !-exponentiation) for the specific purpose of duplicating the argument in conjuncts.

4.4 A Proper Treatment of Co-Predications and Constraints

Recall that our goal in proposing an implementation with linear types was to give an account of the compatibilities, or lack thereof, of transformations used in co-predications. For this, we refine the typing of transformations as follows:

$$f:(\tau\multimap\alpha)\&((\tau\multimap\delta)\multimap\mathbf{t})$$

The linear operator &, read "with", is the additive conjunction. This means that a transformation f can be viewed as a pair $<\pi_1\ f,\pi_2\ f>$. $(\pi_1\ f)$, of type $\tau\multimap\alpha$, is the transformation we have been using up to now, turning an argument of a type τ into another of type α which is needed by some predicate. The second member of the pair, $(\pi_2\ f)$ of type $((\tau\multimap\delta)\multimap\mathbf{t})$, is a *compatibility test* for other transformations. In a nutshell, this will be applied to other transformations to check type compatibilities (there have to be transformations available with τ as their source and δ, or a subtype thereof, as their target) and to accept or reject specific transformations (the term itself returns a value of type **t** that might be true or false).

We modify our polymorphic conjunction as follows:

$$\Lambda\alpha\beta\delta\gamma\tau\lambda^{0}P^{\alpha\multimap\mathbf{t}}Q^{\beta\multimap\mathbf{t}}\lambda x^{\tau}f^{(\tau\multimap\alpha)\&((\tau\multimap\delta)\multimap\mathbf{t})}g^{(\tau\multimap\beta\&((\tau\multimap\gamma)\multimap\mathbf{t})}.(\wedge^{\mathbf{t}\multimap\mathbf{t}\multimap\mathbf{t}\multimap\mathbf{t}}\ (P\ (f\ x))\ (Q\ (g\ x))\ (\texttt{comp}\ f\ g))$$

(Note that, in order to be tested for compatibility, f and g have to be duplicated and are no longer labelled as "linear".)

The comp operator is simply the following:

$$\texttt{comp} =_{def} \Lambda\alpha\beta\gamma\delta\tau\lambda f^{\tau\multimap\alpha\&((\tau\multimap\delta)\multimap\mathbf{t})}g^{\tau\multimap\beta\&((\tau\multimap\gamma)\multimap\mathbf{t})}.(\wedge^{t\multimap t\multimap t}\ ((\pi_2\ f)(\pi_1\ g))\ ((\pi_2\ g)(\pi_1\ f)))$$

Or, abbreviating a bit and omitting typing and projections, $(\wedge\ (f\ g)\ (g\ f))$. Recall that the linear & reflects a pair of objects, only one of which will be ultimately used; in transformations, the first part (the transformative term) is used for adapting the argument and as argument for compatibility tests, the second (the compatibility test) is used only when it has a transformation as its argument. The typing ensures that there are no other possibilities.

Using linear mechanisms for compositional semantics, as straightforward as it is from a logical and formal point of view, requires a change of perspective in the interpretation of semantical compositionality. As it is apparent that the entities involved in linear applications do not disappear from the mind of the speaker as meaning is constructed, the "meaning assembly" phase presented here should be integrated with additional mechanisms. We always suppose that the context of each utterance is provided in a mechanism similar to Luo's Signatures, Asher's presuppositions, or as Cooper storage or in continuation-style semantics. We have not elaborated on this aspect here, but providing an account in the style of, e. g. Dynamic Semantics is straightforward and require the manipulation and output of a provided complex object recording the context. With linear application, this becomes necessary in order to keep track of the consumed terms.

There are thus two different mechanisms at work in the "meaning assembly" phase:

1. A representation of the entities, relations and facts inferred by the discourse, updated incrementally;

2. A mechanism that derive the predicated facts, from instances of the entities concerned, that are consumed as the composition proceeds.

In each step of this composition, fresh copies of the entities involved are created and combined to provide a set of new entities and facts that are integrated to the context. Simple predication asserts a fact, the use of transformations produce different entities that can be referred to later in the discourse, and co-predications duplicate the entity involved in order to assert a number of facts on potentially different facets of the entity.

4.5 Reasons of the Upgrade

Linear ΛTY_n is a step forward from ΛTY_n in several ways. It allows for greater flexibility in the definition of constraints on co-predicative utterances, covering different possible phenomena with a word-level distinction. It is based on a more constrained logic, linear Intuitionist logic second-order logic, which is an interesting fragment in itself (with many operations that can be resolved by lazy evaluation), with constraints on the higher-order quantification; we are much closer to the probable complexity of human compositional semantics than with full System-F. More importantly, this implementation dispenses with external constraints that are stapled to System-F with an *ad hoc* verification mechanism. The constraints of compatibility between transformations is now integrated to the system, and each transformation now has these constraints as a part of its implied meaning. The result is much more satisfying than ΛTY_n, even if it is still far from complete.

5 Summary

The computation of semantics is a long, segmented process, starting from a syntactic analysis to infer the rough form of a Chomskyan deep structure, integrating terms derived from the lexicon and composing them in the meaning assembly phase, writing the resulting term as a logical representation and interpreting it as a truth-valued statement, a model-checker, a set of possible worlds or a sequence of judgements on entities and facts, before combining any interpretation with others to form a representation of the discourse. The study of lexical semantics tends to add separate mechanisms that allow discourse-level and semantic-level data to influence the lexicon (and reciprocally), resulting in an even more complex picture.

The strength of this proposal is thus to keep things in a single framework for meaning assembly, an usual λ-calculus-based process using terms typed with formulae of second-order Intuitionist linear logic in a many-sorted type set, and to express constraints on the use of terms internally through the typing of the terms.

Moreover, our system makes use of a restricted fragment of the wider second-order linear logic, and is a straightforward extension of existing Montagovian frameworks.

Linear ΛTY_n should be completed with resolution rules for type mismatches that better embed coercive sub-typing and its formal properties (publication pending, by Frederic Lang). In addition, there are several additional structural constructs to be specified: Determiner Phrases and quantification using Russell/Hilbert operators (work in progress), as well as adjuncts such as adjectives and adverbs. A proper characterisation of sorts, lexical transformation and wide-coverage lexica should also be undertaken (via semi-automated acquisition from corpora).

A capable system would also need extended linguistic investigations. While automated or semi-automated acquisition is possible to build a starting lexicon, specific studies would allow us to define more precisely some aspects of the system, specifically the constraints on transformations used in co-predicative utterances. It would also allow us to refine term-specific constraints and transformations according to language- and dialect-specific idiosyncrasies.

For now, we propose this simple implementation of lexical semantics, including the treatment of constraints on co-predications, using the λ-calculus with second-order Intuitionist linear types. We are confident in its ability to cover target phenomena and prove flexible enough to account for idiosyncrasies, dialects and creative uses.

References

[Abramsky, 1993] Abramsky, S. (1993). Computational interpretations of linear logic. *Theor. Comput. Sci.*, 111(1&2):3–57.

[Asher, 2011] Asher, N. (2011). *Lexical Meaning in Context: a Web of Words*. Cambridge University Press.

[Bassac et al., 2010] Bassac, C., Mery, B., and Retoré, C. (2010). Towards a Type-Theoretical Account of Lexical Semantics. *Journal of Language, Logic, and Information*, 19(2).

[Bekki and Asher, 2012] Bekki, D. and Asher, N. (2012). Logical polysemy and subtyping. In Motomura, Y., Butler, A., and Bekki, D., editors, *JSAI-isAI Workshops*, volume 7856 of *Lecture Notes in Computer Science*, pages 17–24. Springer.

[Benton et al., 1993] Benton, P. N., Bierman, G. M., Paiva, V. d., and Hyland, M. (1993). A term calculus for intuitionistic linear logic. In *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, TLCA '93, pages 75–90, London, UK, UK. Springer-Verlag.

[Braüner, 1996] Braüner, T. (1996). Introduction to Linear Logic. In *BRICS Lecture Series*.

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

[Girard, 1972] Girard, J. Y. (1972). Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur. Thèse de Doctorat d'État, Université Paris VII.

[Girard, 1987] Girard, J.-Y. (1987). Linear Logic. *Theoritical Computer Science*, pages 1–102.

[Girard, 1995] Girard, J.-Y. (1995). Linear logic : its syntax and semantics. In *Advances in linear logic*.

[Gupta and Aha, 2005] Gupta, K. M. and Aha, D. W. (2005). Interpreting Events Using Generative Sublanguage Ontologies. In *Third International Workshop on Generative Approaches to the Lexicon*.

[Luo et al., 2013] Luo, Z., Soloviev, S., and Xue, T. (2013). Coercive subtyping: Theory and implementation. *Inf. Comput.*, 223:18–42.

[Mery, 2011] Mery, B. (2011). *Modélisation de la Sémantique Lexicale dans le cadre de la Théorie des Types*. PhD thesis, Université de Bordeaux.

[Mery et al., 2015a] Mery, B., Moot, R., and Retoré, C. (2015a). Computing the Semantics of Plurals and Massive Entities using Many-Sorted Types. In *To appear in the selected proceedings of LENLS 11*, Lecture Notes in Computer Science. Springer.

[Mery et al., 2015b] Mery, B., Moot, R., and Retoré, C. (2015b). Typed Hilbert Operators for the Lexical Semantics of Singular and Plural Determiner Phrases. In *Epsilon 2015 – Hilbert's Epsilon and Tau in Logic, Informatics and Linguistics*, Montpellier, France.

[Mery and Retoré, 2013] Mery, B. and Retoré, C. (2013). Recent advances in the logical representation of lexical semantics. In *NCLS – Workshop on Natural Language and Computer Science, LiCS 2013*, Tulane University, New Orleans.

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

[Muskens, 1996] Muskens, R. (1996). Meaning and Partiality. In Cooper, R. and de Rijke, M., editors, *Studies in Logic, Langage and Information*. CSLI.

[Nunberg, 1993] Nunberg, G. (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.

[Pinkal and Kohlhase, 2000] Pinkal, M. and Kohlhase, M. (2000). Feature Logic for Dotted Types: A Formalism for Complex Word Meanings. In *ACL 2000*.

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

[Pustejovsky, 1995] Pustejovsky, J. (1995). *The Generative Lexicon*. MIT Press.

[Pustejovsky and Asher, 2000] Pustejovsky, J. and Asher, N. (2000). The Metaphysics of Words in Context. *Objectual attitudes, Linguistics and Philosophy*, 23:141–183.

[Retoré, 2014] Retoré, C. (2014). The Montagovian Generative Lexicon Lambda Ty_n: a Type Theoretical Framework for Natural Language Semantics. In Matthes, R. and Schubert, A., editors, *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 202–229, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Retoré, 2014] Retoré, C. (2014). Typed Hilbert Epsilon Operators and the Semantics of Determiner Phrases. In *Formal Grammar 2014*, Tübingen, Germany.

[Soloviev and Luo, 2001] Soloviev, S. and Luo, Z. (2001). Coercion completion and conservativity in coercive subtyping. *Annals of Pure and Applied Logic*, 113(1-3):297–322.