Collecting Crowd-Sourced Lexical Coercions for Compositional Semantic Analysis

Mathieu Lafourcade, Bruno Mery, Mehdi Mirzapour, Richard Moot, Christian Retoré

Collecting Crowd-Sourced Lexical Coercions for Compositional Semantic Analysis

Mathieu Lafourcade[1], Bruno Mery[2,3], Mehdi Mirzapour[1], Richard Moot[1], and Christian Retoré[1]

[1] LIRMM – UMR 5506, CNRS & Université de Montpellier
[2] LaBRI – UMR 5800, CNRS & Université de Bordeaux
[3] IUT de Bordeaux, Université de Bordeaux

Abstract. Type-theoretic frameworks for compositional semantics are aimed at producing structured meaning representations of natural language utterances. Using elements of lexical semantics, these frameworks are able to represent many difficult phenomena related to the polysemy of words and their context-dependent meanings. However, they are just as powerful as their built-in lexical resources. This paper explores ways to create and enrich wide-coverage, weighted lexical resources from crowd-sourced data, specifically, investigating how existing rich lexical networks – created and validated by serious games – can be used to infer linguistic coercions with ranking.

1 Type-Theoretic Semantic Frameworks with Rich Lexical Information

The **semantic analysis** of natural language aims to produce a complete and structural meaning representation of a given text (for example, a logical formula or a Discourse Representation Structure) that makes explicit the entities referenced in the text as well as their relationships; this is used for word sense disambiguation, coreference resolution, natural language inference and other complex tasks. Rich lexical information is required to compute the meaning of utterances such as *I am going to the bank*, as *bank* is ambiguous between a geographic feature and a service building. Frameworks based on theories such as [6] and [17], including [1], [2], [5], [10] and [18] are able to obtain a representation using a Montague-like compositional process that correctly interprets sentences such as *I am going to the bank; they blocked my account*. They not only interpret the types of the lexemes involved as indicating that *bank* is a service building, but also that *they* is a reference to a financial institution that is introduced in the first part of the sentence, and then coerced to a relevant human agent in the second. Such frameworks require a rich corpus of lexical resources incorporating some degree of world knowledge encoded as complex types and lexical coercions; several of these frameworks benefit from software implementations, such as [4] and [11].

In these approaches, the natural language utterance is analysed in its syntax and semantics layers as in classical Montague grammar, producing a **logical**

form; typing the terms with a **rich system of sorts** intended to capture **restrictions of selection**, using a **semantic lexicon**, will produce some **typing mismatches** whenever polysemous terms are **linguistically coerced** to one of their facets. In our approach, the *Montagovian Generative Lexicon* (MGL), detailed in [18], this is characterised by a **mismatched application**, such as a functional predicate P requiring an argument of type B being applied to an argument a of type A: $(P^{B\rightarrow \mathbf{t}}\ a^{A})$. This is resolved using a **lexical transformation** that serves as the representation of the **linguistic coercion** taking place, with two possibilities: *adapting the argument* with a transformation $f_1^{A\rightarrow B}$, the application becoming $(P\ (f_1\ a))$, and *adapting the predicate* with a transformation $f_2^{(B\rightarrow \mathbf{t})\rightarrow (A\rightarrow \mathbf{t})}$, the application becoming $((f_2\ P)\ a)$. Depending on the transformations that are provided by the functional terms P and a, one or the other adaptation occurs.

The phrase *the dinner was delicious but took a long time* (adapted from a canonical example) can schematically be represented as
$(\text{and}\ (\lambda x.\text{delicious}\ x)\ (\lambda x.\text{long}\ x))\ \text{dinner})$. Within a many-sorted system, *take a long time* is restricted to events (entities of type $\mathbf{evt}$), *delicious* is restricted to food (entities of type F), and at least a transformation is needed in order to predicate on the two facets of *dinner*; MGL resolves this by having *dinner* of type $\mathbf{evt}$ and possessing a transformation $f_{content}^{\mathbf{evt}\rightarrow F}$, a function mapping the dinner event to the food that was served at the dinner in question.

A crucial issue for these systems is then to have sufficient lexical resources to function. In order to be successful, MGL and other type-theoretic logical frameworks for lexical semantics require:

Syntax-Semantics Analysis: A suitable syntax-semantics analyser which is able to provide a sufficiently structured output for a Montague-style analysis. MGL can make use of Grail ([15]) easily, yielding λ-DRT-based outputs.

A System of Base Types: Each word (lexeme) should be associated to a type in a defined system of **lexical sorts** (base types). While the exact definition of these sorts is debated, this can be decided arbitrarily; the principle is to capture the *restrictions of selection* by the predicate; the system of sorts can be refined if necessary.

A Set of Lexical Coercions: The core of MGL-based lexical semantics is the set of lexical transformations (corresponding to the *linguistic coercions* available for each *lexeme*), that allows co-composition to occur. Transformations might also be *constrained* in their use, and be dependent on a specific *context*. The difficulties in building a wide-coverage lexical database for MGL lies in the acquisition of all such relevant transformations for all lexemes.

To sum up, MGL needs lexical entries in a format such as the following:

Lexeme	Logical Term	Type	Comments
dinner	$(\lambda x.\text{dinner } x)$	$\mathbf{evt} \to \mathbf{t}$	As in Montague semantics, nouns are predicates ; $\mathbf{evt}$ is for *events*, $\mathbf{t}$ for *propositions*
associated with dinner	$f_{content}$	$\mathbf{evt} \to F$	Transformation to "food that was served at dinner" (type F)
the dinner	the_dinner	$\mathbf{evt}$	MGL mechanisms yield an entity-typed term for DPs
to take a long time	$(\lambda x.\text{long } x)$	$\mathbf{evt} \to \mathbf{t}$	Predicate of events
to be delicious	$(\lambda x.\text{delicious } x)$	$F \to \mathbf{t}$	Predicate of food

(MGL encompasses higher-order composition mechanisms that will allow operators such as *the* and *and* to compute the correct predications.)

The main goal of this paper is to show how we can obtain the *lexical transformations* (such as $f_{content}$ above) for our lexical entries.

2 Lexical Data Crowd-Sourced from Serious Games

While lexical networks have been developed for a long time (*WordNet*, defined in [14], being the reference), there have been several recent efforts to build collaborative, crowd-sourced resources that can reflect the current uses and relations of words by language speakers. One approach, given in [8], is to engage as many people as possible in a "serious" game (or, more accurately, a "game with a purpose") in order to identify and co-validate lexical and relational information by having different competent speakers of the language competing to identify lexical meaning and relations between words. These games include *JeuxDeMots*, described in [9], which has provided a lexical network that comprises more than 150 million relations between words (as strings of characters) and has proven remarkably robust. One advantage of this network over expert-produced and corpus-based resources is that a large amount of world knowledge has been added by the players: facts such as restrictions of selection (as in *cats can meow*) or ontological inclusion (as in *armchairs are chairs*) are explicitly produced by human players, while they are hard to get from other sources because of their "trivial" nature. This network can be used as a relevant source of lexical data for type-theoretic frameworks, as discussed in [3] for MTT. While that publication has demonstrated the capacity in which the *types* for each lexeme can be extracted and deducted (an MGL-compatible lexicon with a different set of core sorts can easily be produced), we want to focus on the extraction of the *lexical transformations* from such lexical networks.

We will be using the lexical network created by *JeuxDeMots* (amended by several related open, contributive resources), Grail as a syntax-semantics analyser and MGL for lexical semantics in order to produce a coherent treatment process for French text.

2.1 Lexemes, Sorts and Sub-Types

JeuxDeMots and many other lexical networks operate upon character strings, while MGL differentiate between *contrastively ambiguous* homonyms. Words such as *bank* are considered has having (at least) two different entries in the Generative Lexicon tradition: one of the type *Financial Institution* and the other of type *Geographical Feature*, that have the same string representation. In *JeuxDeMots* where the string is the basic unit, there are two ways to detect contrastive ambiguity: *Semantic Features* and *Refinements*. A *Semantic Feature* is similar to a lexical sort (or base type), as it reflects a broad category of things that the character string can denote; there might be several of such features associated to each string. *Refinements* are single-meaning facets for this string that have been crowd-sourced for the express purpose of resolving the contrastive ambiguity.

For example, in French, *un bar* is ambiguous and can denote at least three different things:

- a place where drinks are served (as in the English "bar", roughly synonymous with "public house" and "café", polysemous via metonymy with the furniture item sharing this name and function);
- a type of fish (in English, "sea bass");
- and a pressure unit (1 bar $=$ 100kPa).

This is denoted in the data from *JeuxDeMots* as having different refinements, as well as having several semantic features, including some that are incompatible with each other (here, "bar" has "location", "artefact" and "living being" as features). Extracting the initial information from *JeuxDeMots* is an easy task. *Common nouns* are logically predicates of a type $\tau \rightarrow \mathbf{t}$, with τ the *sort* of entity involved, initially one of the salient semantic features in *JeuxDeMots*. These initial sorts can be refined later if necessary for selection restrictions, which are also included in the lexical network (for example, the networks detail possible patients and agents of predicates). If the crowd-sourced data indicates that several incompatible semantic features are available for a single word, it simply means that we will have several distinct entries of several different lexemes that have the same string representation.

The typing for word taking nouns as arguments (such as adjectives, verbs...) are derived from a similar process and has been thoroughly explored in [3]. Specific sorts are added as has been proposed for MGL in [12] : specific sorts can be introduced for nouns denoting *groups of a given sort* (such as *committee* being a predicate that denotes a group of people, $\mathbf{g}_P$) and *massive entities* such as *water* being a mass physical noun, of sort $\mathbf{m}_\varphi$). Several "operational" terms, such as the polymorphic conjunction *and* and determiners derived from Hilbert operators, are added by hand.

We can also derive a sub-typing mechanism. However, as discussed in [13], MGL restricts this mechanism (as well as the strict notion of "coercion among types") to *ontological inclusions*. In the core system of sorts, this ontological hierarchy can be detected in data given from *JeuxDeMots* as denoting *hyperonymy* quite easily (such relations are pervasive in lexical networks). No other type-driven coercions are included in the system.

2.2 Lexical Transformations

From the definitions of the MGL system that have been presented before, the way to determine the lexical transformations never has been explicitly mentioned. By playing *JeuxDeMots*, players effectively create a database of **coercions between words** that we need to process according to the lexical entries and their typings.

As explained before, there are two ways to resolve a **type mismatch** in an application such as $(P^{B\to\mathbf{t}}\ a^{A})$: adapting either **the functional argument** or **the predicate**.

Adapting the Argument

The most common adaptation is done on the argument, using a relevant transformation $f^{A\to B}$, yielding the correctly-typed $(P\ (f\ a))$. There are two **origins** possible for the transformation f: it can either come from the argument term a itself, or from the predicate term P.

Argument-driven transformations are extracted directly from entries revealing several meanings, that will be constrained by predication. For example, the expression *a book* is itself polysemous; an applied predicate such as *read*, *finish*, or *pack* will have strong typing constraints that will select one or several meanings. In MGL, the lexeme *book* (understood as the common noun associated to a literary concept versus the calendar-related verb) will have *a single sort*, R for readable object, and several transformations: $f^{R\to P}_{author}$, $f^{R\to\varphi}_{object}$, $f^{R\to\mathbf{evt}}_{read}$, $f^{R\to\mathbf{evt}}_{write}$, etc.

In all of these cases, the transformation is *part of the lexical entry for the argument* and *constrained by the typing of the predicate*: *pro-monarchy* applies to people and is of type $P\to\mathbf{t}$, *to read* is of type $R\to\mathbf{t}$, *to finish* of type $\mathbf{evt}\to\mathbf{t}$, and *to pack* of type $\varphi\to\mathbf{t}$.

In order to derive these transformations from lexical networks such as *JeuxDeMots*, we list the various possible *target types* for the predicates that are listed as *common patients* for this word. Then, for each target type, we select the compatible *associations* listed for the word, and generate a lexical transformation associated to this word, of a given typing, labeled with the association that is given in the lexical network. We retain the weight (frequency) of the association in order to filter out the more dubious transformation at a latter stage (this can also be used to rank the preferred interpretations, if several are available).

For this example, listing all strings associated with *livre* (book; there are more than 3000 associations in the network, weighted and labelled), we will scan for *nouns denoting humans* and obtain *auteur* (author) and *écrivain* (writer), both with very strong relative weights; for *action verbs denoting non-instantaneous events* (that can said to *begin* or *end*) and obtain *lire* (to read) and *écrire* (to write), with a much stronger relative weight for the former. This process is detailed in Section 4. Even if some associations provided by the players in the network are dubious, filtering by type and syntactic properties yields exactly what is needed for the lexicon.

Predicate-driven transformations are not (all) to be found in lexical networks or crowd-sourced data, as they characterise a predication made "in the spur of the moment". For example, the predicate *to read* is associated to a transformation $f^{\varphi \to R}_{written}$ that allows sentence such as *I read the wall* to be felicitous (and simply supposes that there is something written on said wall). Of predicate-driven transformations, some will be derived from lexical data (*grinding* is a common example, whenever *food* is associated to a word denoting a *living being*); others will need to be generated whenever the predication occurs, and be invalidated or not. The dynamic aspect of the crowd-sourced lexical network will integrate additional transformations as they are deemed pertinent, or fashionable, by its community; this is a strong argument in favor of such resources.

Adapting the Predicate
Transformations that change the type of a predicate are less common in MGL mechanisms. They are either provided by adverbs or other modifiers to a predicate, or intrinsic to the predicate. The former will be changing the target typing of a predicate. Examples of the latter include the polysemy of readings between collective and distributive among plural predicates; these should be added manually for the lexicon.

2.3 Constraints and Relaxation

Transformations are associated to compatibility constraints (either as logical operators, or as arbitrary functions); these are intended to suppress or signal hazardous co-predications. Our interpretation of such phenomena is the following: *predicate-modifying* transformations are cumulative, *argument-driven* transformations are compatible with each other and not constrained, and *predicate-driven* transformations are exclusive: only one of them can be used on a given entity, exclusive of any other transformations and of the original term. As suggested by several studies including [18], the latter constraint can be relaxed. Lexical predicate-driven transformations such as *grinding* (that we can get from crowd-sourced data) are compatible with other meanings, as long as there is a syntactic break between the two predications. There are ways to relax the constraints of application of non-lexicalised predicate-driven transformations as well, but these are syntax-, discourse- or pragmatic- dependent.

Detecting and validating constraints on co-predication, beyond the simple claim above, can also be a crowd-sourced task. *JeuxDeMots* itself is not suitable for this; however, a recent effort, *Ambiguss* (available at `https://ambiguss.calyxe.fr/`) from the same team should become a good resource for this in the near future. *Ambiguss* is a database of sentences containing ambiguous (polysemous) terms that asks players to compete in determining the possible readings for those terms in context, and thus will be able to detect whenever co-predications are accepted or rejected when enough data will have been crowd-sourced; *Ambiguss* should also be useful for the evaluation of the performance of MGL on word-sense disambiguation tasks.

3 Integrating and Ranking Transformations

3.1 Adding Collected Transformations to the Lexicon

The semantic lexicon in MGL associates a set of *lexical transformations* (as optional λ-terms) to each lexeme. When the pertinent data has been collected from *JeuxDeMots*, the coercion is transcribed as a transformation with a functional type $A \to \tau$ where A is the sort of the source lexeme and τ is the expected typing, predicted during the collection process. The target concept serves as the name of the transformation. This data can be directly input and used in our prototype implementation of MGL given in [11] (as the only necessary data are a source type, a target type and a name), and is added to the list of transformations of the current lexeme.

3.2 Scoring Interpretations

Compositional Lexical Semantics can produce the precise meaning of a polysemous term in context. However, there are cases where the immediate context is not sufficient to totally determine the sense used for a word, and a composed phrase can still be ambiguous: in the example above, *finir un livre*, the entity *livre* (book) is coerced to an event with a duration, but there are many suitable lexicalised events that can be used, and thus many lexical transformations that have a correct typing that can be used to resolve the type mismatch. MGL usually resolves this by producing **several** interpretations: one interpretation (a well-typed logical formula with a suitable transformation placed whenever necessary) per available coercion. The production of MGL is thus not a single logical representation, but a **collection of representations**.

Associating a preference score to different possible interpretations given by a compositional formalism is not a common area of research (while this practice is pervasive in statistical or machine learning approaches to word-sense disambiguation); the more convincing works on this matter are mostly derived from Preference Semantics described, e. g., in [7]. An interesting extension is [16], giving an implementation of the constraint-based preference scoring.

In order to provide a preference ranking, we exploit a strong advantage of extracting coercions from crowd-sourced data: the most common preferred coercions associated to a lexeme will be clearly represented in the lexical network (because *JeuxDeMots* counts the number of times each fact has been contributed or validated). Using the relative weights of different coercions, MGL will then be able to associate a *relative score* (similar to a probability measure) to each logical representation produced, allowing to rank the interpretations (while still being able to generate all possible meanings).

Our proposal for producing this **score** consists in a simple procedure. In order to analyse and rank interpretations for *finir un livre* (finish a book) :

1. Starting from textual data syntactically analysed using Categorial Grammars, we obtain a full logical representation, the terms of which are

typed using a **many-sorted semantic lexicon** (in which event types $DurableEvent$ and $AtomicEvent$ are differentiated), we first identify the **adaptation** taking place in applications with type mismatches such as (finish^{DurableEvent} a_book^{Readable}).
2. The lexicon then uses data available from *JeuxDeMots* in order to build the **set of possible coercions**: taking all **relations to the argument first** *livre* (book), we **select the ones with compatible types** with the typing of the predicate: verbs denoting a non-instant action. In this example, there are only *lire* (680) and *écrire* (210) in the first two hundred relations.
3. Finally, normalising to a probability-like measure, this yields a **score between 1 and 0** for several interpretations ranked by order of probability. In this example, this means a score of 0.764 for *lire* and 0.236 for *écrire*.
4. This is validated by the crowd-sourced glosses for the complete phrasal expression that appear in that same order above.

In the case where **no suitable coercion is given by the argument**, a predicate-driven transformation can be used, as discussed in Section 2.2. Lexical transformations provided by the (functional) argument are always more specific and preferred to generic, predicate-driven transformations; thus the latter only occur when there is no other choice, and there is only one interpretation (that will not be associated with a preference score).

Specific contexts can provide coercions that are not scored by this procedure, or that have different preferences. For instance, different contexts of enunciation will change the order of preferences in *finish a book*, placing *writing* before *reading* in an academic universe, or adding new possible coercions in contexts such as a bookbinding workshop. In order to capture these differences, JDM allows to filter entries by the background of their contributors, but this feature is still experimental, though, it greatly reduces the available data.

3.3 Correcting the Lexicon using Different Sources

JeuxDeMots is a strong and rich resource for French as it is, encompassing at the moment more than 150 million relations between more than 2 million terms (words, names, and phrases). However, one of its more interesting characteristics for our purpose is that it is a living resource, constantly updated by crowdsourcing (there are several players logged in and actively contributing at any given time) and experts (selected members of the team constantly correct, update and add linguistic data and world knowledge). This means that, after acquiring the initial lexicon used for a MGL system by converting relevant data from *JeuxDeMots* as outlined above, the same source can be used in order to continuously update and correct our lexicon. The presence of *phrases* in data from *JeuxDeMots* can also be used to evaluate and validate our compositional system. We can, for instance, derive the type and transformations for the lexical entry of *finir* (*to finish*), do the same thing for *livre* (*book*), and conclude from a MGL composition the meaning of *finir un livre* (*finish a book*: what is *finished* is

the event of *reading* or *writing* a book); this is validated by the associated meanings of *finir un livre* that appears as a phrasal expression in *JeuxDeMots*.

This process can be systematised and automated, as *JeuxDeMots* can be enriched by asking the players the meaning of an expression that has not yet been analysed (within reasonable limits). The meaning of words in context, as predicted by our compositional system, can also be checked by the players of the serious game *Ambiguss*. Thus, the evaluation of our system, as well as its correction and continuous improvement can be crowd-sourced.

4 A Short Case Study

We will concentrate on some examples in order to demonstrate how the ideas that are described in previous sections can actually be achieved using *JeuxDeMots*, illustrating the translation procedure from *JeuxDeMots* to proper meaning representations for MGL.

4.1 The Straightforward Case

Let us consider the phrase *To finish a book* in a situation where we have Claire, a researcher, reading a book by Villani:

(1) Claire a fini le livre de Villani.
'Claire finished Villani's book.'

In order to analyse such a sentence, we search for a more generic phrase in *JeuxDeMots*: **(a)** *la femme a fini un livre* (the woman finished a book) for (1) ; lexical coercions to actions such as *reading* or *writing* are excepted from *JeuxDeMots*. Moreover, *JeuxDeMots* also gives us the built-in weights relation that results in the expected ranking on the obtained lexical coercions.

4.2 Lexicon Organization in MGL and Meaning Representation

A detailed explanation on lexicon organization in MGL is available in the literature [18, Sec 2.4]. In summary, the lexicon in MGL associates to each word w a principal λ-term which is Montague term with a much richer typed system and optional λ-terms known as *modifiers* or *transformations* modeling *lexical coercions*. The following sample lexicon is designed for the example (a):

Lexeme	Main λ-term	Optional λ-terms
livre (obj)	book$^{R\to\mathbf{t}}$	$f_{read}^{R\to evt}$ $f_{write}^{R\to evt}$
femme	woman$^{P\to\mathbf{t}}$	
fini	finish$^{\alpha\to(\mathbf{evt}\to\mathbf{t})}$	

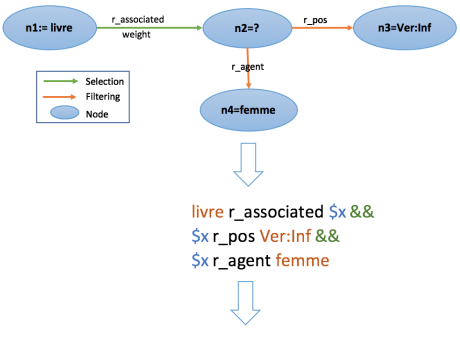
The sorts used here are as follows: P for $Person$, R for $Readable$, and $\mathbf{evt}$ for $Event$. Skipping unnecessary details on quantifiers and syntax, we can represent two possible meanings for (a) as
$((finish^{\alpha\rightarrow(\mathbf{evt}\rightarrow\mathbf{t})}$ (the woman)$^{P})$ $(f_{read}^{R\rightarrow\mathbf{evt}}$ (a book)$^{R}))$ and
$((finish^{\alpha\rightarrow(\mathbf{evt}\rightarrow\mathbf{t})}$ (the woman)$^{P})$ $(f_{write}^{R\rightarrow\mathbf{evt}}$ (a book)$^{R}))$.

4.3 Collecting Coercions

The relations gained form the game players in *JeuxDeMots* can technically be represented in different structures. The two kinds of data representation that are implemented and available are **relational** and **graph-based databases**. A simple SQL query on the existing RDBMS or *Cypher*, the graph query language, on the existing graph-based database can give us our targets. We can also use *Datalog* which is a declarative database query language which has deep grounds in logic programming. Although all of the options are technically available, we illustrate the process using a PHP-like high-level query syntax in our diagrams which should be easy to understand and has actually been experimented.

For example (a), what we want to do is basically to find the coercions *lire* (read) and *écrire* (write) for a sentence with the object *livre* (book) and the subject of type *Person*. As illustrated in Fig. 1, we can find all the nodes with relation *r_associated* to the node *livre*. Two filters are then applied. The first one rules out the candidates that do not have the syntactic property of being a verb; to do so in *JeuxDeMots* we use *r_pos* relation that targets the node *Ver:Inf*. The second rules out the verbs that cannot have an agent of type *Human*; to do so we use *r_agent* relation that targets the node *femme* (woman). We sort in descending order the final table with the built-in weight property of *r_associated* relation that exists in *JeuxDeMots*.

Fig. 1. Obtaining Coercions: General Scheme, Query Code and Outcome Table for Example (a)



n1	weight	relation	n2	relation	n3	relation	n4
livre	680	r_associated	lire	r_pos	Ver:Inf	r_agent	femme
livre	210	r_associated	écrire	r_pos	Ver:Inf	r_agent	femme

4.4 The Non-Human Case

Regarding the extraction of the coercions from JeuxDeMots, we can see that the input of a query involves more than two words. For instance, considering the meaning of "dévorer un livre" (idiomatic French, *to devour a book* that can be used to denote binge-reading); in JeuxDeMots a relation between one of the meanings of "dévorer" and one of the meanings of "livre" depends on a third ingredient, namely the agent of "dévorer". Assuming that Blanche is Claire's pet goat, contrast the following:

(2) a. Claire a dévoré le livre de Villani.
'Claire devoured Villani's book.'
b. Blanche a dévoré le livre de Villani.
'Blanche devoured Villani's book.'
c. Claire a permis à Blanche de finir le livre de Villani.
'Claire allowed Blanche to finish Villani's book.'
d. Claire a promis à Blanche de finir le livre de Villani.
'Claire promised Blanche to finish Villani's book.'

Observe that the last two examples with control verbs require a syntactic computation to determine the agent/subject of the action that is performed on the book.

A further limit is that there are no sorts, types or sets in JeuxDeMots. If one is asked what a goat can eat, it is unlikely that a player answers"a book". The answer: bedsides, grass, bushes, flowers, branches, leaves etc. would be "any object that is small and not too hard", but there is no single word corresponding to this class.

For this particular case, the fact that a goat may eat a book **is**, actually, included in JeuxDeMots. Some players included and validated the fact that a goat may eat "paper" and books being made of paper (this is indicated in the entry for "book" as a constitutive coercion), they can be eaten by goats. In this case, there even is also a direct fact that goats can eat books, but with a much weaker confidence.

4.5 Limits of MGL

In MGL as it stands now, coercions are attached to one word – except ontological inclusions which are encoded via sub-typing – but the actual coercion used to fix a type mismatch could be the combination of several coercions provided by all the words in the expressions. As observed above a coercion may be triggered by several words, and may be the result of a sequence of relations. How can this be stored in MGL, in a way that the general compositional mechanism of MGL produces the wanted readings and discards the unwanted ones, while not becoming of unreasonable computational complexity?

A solution is to split the coercions, one part going attached to each word and the combination giving the needed coercion. A type mismatch $P^{A\to\tau}(u^B)$ may be solved by first using a coercion $f^{B\to X}$ attached to u and a coercion $g^{X\to A}$ attached to P.

In all the above examples, there is a coercion from *books* to their physical facet. Having a *goat* as an agent will provide the verb *to eat* (which normally has an agent of type $Animal$ and a patient of type $Food$) with a coercion from $Food$ to physical objects, representing the *ingestion* of these objects and the world knowledge "fact" that "goats will eat (mostly) anything". That way, the goat Blanche may well eat Villani's book.

4.6 A Direct Solution

We would expand our sample lexicon to this:

Lexeme	Syntax Constraint	Main λ-term	Optional λ-terms
livre (obj)	*subj*: P	book$^{R\to\mathbf{t}}$	$f_{read}^{R\to evt}$ $f_{write}^{R\to evt}$
livre (obj)	*subj*: A	book$^{R\to\mathbf{t}}$	$f_{eat}^{R\to evt}$
chèvre femme fini		goat$^{A\to\mathbf{t}}$ woman$^{P\to\mathbf{t}}$ finish$^{\alpha\to(\mathbf{evt}\to\mathbf{t})}$	

The syntactic constraint column extends the standard MGL lexicon in order to capture meanings that depend on the object and subject in a given sentence. In our case, having a subject of type either *Animal* or *Person* can significantly change the meaning and it obviously needs different kind of coercions in the meaning representation layer.

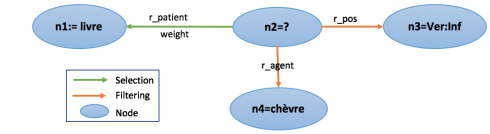
Considering the example:

(3) Blanche a fini le livre de Villani.
'Blanche finished Villani's book.'

We adapt this to a more generic phrase, **(b)** *la chèvre a fini un livre* (the goat finished a book) for (3), and the single reading for (b) should be $((finish^{\alpha\to(\mathbf{evt}\to\mathbf{t})}$ (the goat)$^A)$ $(f_{eat}^{R\to\mathbf{evt}}$ (a book)$^R))$.

We want to find the coercion *manger* (eat) for a sentence with the object *livre* (book), the subject being the word *chèvre* (goat). As illustrated in Fig. 2, we can find all the nodes with relation *r_patient* to the node *livre*. As before, two filters are applied, selecting for verbs that can have *chèvre* as object; we could then sort by weight if there were more than a single result.

Fig. 2. Obtaining Coercions: General Scheme, Query Code and Outcome Table for Example (b)




```
$x r_patient livre &&
$x r_pos Ver:Inf &&
$x r_agent chèvre
```

n1	weight	relation	n2	relation	n3	relation	n4
livre	30	r_patient	manger	r_pos	Ver:Inf	r_agent	chèvre

This is the low-confidence direct relation extracted from *JeuxDeMots*. In future work, we want to acquire the coercion using "paper" as an intermediate step by searching for possible sequences of words (of limited length).

Thus, we can use *JeuxDeMots* to derive the coercions that we need incorporate into the MGL lexicon.

Conclusion

We claim that the Grail analyser, the MGL account of lexical polysemy based on compositional semantics and ΛTY_n logic, together with lexical and semantic data crowd-sourced from *JeuxDeMots*, can form a complete chain of analysis that can tackle complex linguistic phenomena for the French language. Grail has been in use for long in different versions, *JeuxDeMots* is a mature database, continuously updated, which data is publicly available, and we have presented a process and experimental data that shows that this treatment chain works and can be automated. Moreover, this system can be evaluated, corrected and updated in a semi-automated fashion using similar crowd-sourced data.

What remains to be done is mostly a work of integration of these various components; we should also examine modifications to the MGL framework that allows multi-part coercions to be added as the composition of transformations licensed from different lexemes.

References

1. Asher, N.: Lexical Meaning in Context: A Web of Words. Cambridge University Press (Mar 2011)
2. Bekki, D.: Dependent Type Semantics: An Introduction. In: Christoff, Z., Galeazzi, P., Gierasimczuk, N., Marcoci, A., Smet, S. (eds.) Logic and Interactive RAtionality (LIRa) Yearbook 2012. vol. I, pp. 277–300. University of Amsterdam (2014)

3. Chatzikyriakidis, S., Lafourcade, M., Ramadier, L., Zarrouk, M.: Modern Type Theories and Lexical Networks: Using Serious Games as the Basis for Multi-Sorted Typed Systems. Journal of Language Modelling (2017)
4. Chatzikyriakidis, S., Luo, Z.: Natural language inference in coq. J. of Logic, Lang. and Inf. 23(4), 441–480 (Dec 2014), http://dx.doi.org/10.1007/s10849-014-9208-x
5. Cooper, R.: Copredication, dynamic generalized quantification and lexical innovation by coercion. In: Fourth International Workshop on Generative Approaches to the Lexicon (2007)
6. Cruse, D.A.: Lexical Semantics. Cambridge, New York (1986)
7. Fass, D., Wilks, Y.: Preference semantics, ill-formedness, and metaphor. Comput. Linguist. 9(3-4), 178–187 (Jul 1983), http://dl.acm.org/citation.cfm?id=1334.980082
8. Lafourcade, M.: Making people play for Lexical Acquisition with the JeuxDeMots prototype. In: SNLP'07: 7th International Symposium on Natural Language Processing. p. 7. Pattaya, Chonburi, Thailand (Dec 2007), https://hal-lirmm.ccsd.cnrs.fr/lirmm-00200883
9. Lafourcade, M., Joubert, A.: JeuxDeMots : un prototype ludique pour l'émergence de relations entre termes. In: JADT'08 : Journées internationales d'Analyse statistiques des Données Textuelles. pp. 657–666. France (Mar 2008), https://hal-lirmm.ccsd.cnrs.fr/lirmm-00358848
10. Luo, Z.: Contextual analysis of word meanings in type-theoretical semantics. In: LACL'11 Proceedings of the 6th international conference on Logical aspects of computational linguistics. Springer-Verlag Berlin (2011)
11. Mery, B.: Challenges in the Computational Implementation of Montagovian Lexical Semantics. In: Kurahashi, S., Ohta, Y., Arai, S., Satoh, K., Bekki, D. (eds.) New Frontiers in Artificial Intelligence: JSAI-isAI 2016 Workshops, LENLS, HAT-MASH, AI-Biz, JURISIN and SKL, Kanagawa, Japan, November 14-16, 2016, Revised Selected Papers, pp. 90–107. Springer International Publishing, Cham (2017)
12. Mery, B., Moot, R., Retoré, C.: Computing the Semantics of Plurals and Massive Entities using Many-Sorted Types. In: Murata, T., Mineshima, K., Bekki, D. (eds.) New Frontiers in Artificial Intelligence JSAI-isAI 2014 Workshops, LENLS, JURISIN, and GABA, Kanagawa, Japan, October 27-28, 2014, Revised Selected Papers, Lecture Notes in Artificial Intelligence, vol. 9067, p. 357. Springer-Verlag Berlin Heidelberg (2015), https://hal.inria.fr/hal-01214435
13. Mery, B., Retoré, C.: Are books events ? Ontological Inclusions as Coercive Sub-Typing, Lexical Transfers as Entailment. In: LENLS '12, in jSAI 2015. Kanagawa, Japan (Nov 2015)
14. Miller, G.A.: Wordnet: A lexical database for english. Commun. ACM 38(11), 39–41 (Nov 1995), http://doi.acm.org/10.1145/219717.219748
15. Moot, R.: The grail theorem prover: Type theory for syntax and semantics. In: Chatzikyriakidis, S., Luo, Z. (eds.) Modern Perspectives in Type-Theoretical Semantics, pp. 247–277. Springer International Publishing, Cham (2017), https://doi.org/10.1007/978-3-319-50422-3_10
16. Nagao, K.: A preferential constraint satisfaction technique for natural language analysis. IEICE TRANSACTIONS on Information and Systems 77(2), 161–170 (1994)
17. Pustejovsky, J.: The Generative Lexicon. MIT Press (1995)
18. Retoré, C.: The Montagovian Generative Lexicon Lambda Ty_n: a Type Theoretical Framework for Natural Language Semantics. In: 19th International Conference on Types for Proofs and Programs (TYPES 2013). Leibniz International Proceedings in Informatics (LIPIcs), vol. 26, pp. 202–229. Schloss Dagstuhl, Germany (2014)