Type Theories and Lexical Networks: using Serious Games as the basis for Multi-Sorted Typed Systems*

Stergios Chatzikyriakidis[1], *Mathieu Lafourcade*[2], *Lionel Ramadier*[3], *and Manel Zarrouk*[4]

[1] Centre for Linguistic Theory and Studies in Probability (CLASP), Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg; Open University of Cyprus

[2] LIRMM, University of Montpellier

[3] Radiology Dept. CHU Montpellier

[4] National University of Ireland, Galway

ABSTRACT

In this paper, we show how a rich lexico-semantic network which has been built using serious games, JeuxDeMots, can help us in grounding our semantic ontologies in doing formal semantics using rich or modern type theories (type theories within the tradition of Martin Löf). We discuss the issue of base types, adjectival and verbal types, hyperonymy/hyponymy relations as well as more advanced issues like homophony and polysemy. We show how one can take advantage of this wealth of lexical semantics in a formal compositional semantics framework. We argue that this is a way to sidestep the problem of deciding what the type ontology should look like once a move to a many sorted type system has been made. Furthermore, we show how this kind of information can be extracted from a lexico-semantic network like JeuxDeMots and inserted into a proof-assistant like Coq in order to perform reasoning tasks.

Keywords: Lexical Networks, JeuxDeMots, Type Theory, Type Ontologies, Formal Semantics, Natural Language Inference

*The first author supported by a grant from the Swedish Research Council for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg.

1 INTRODUCTION

Modern Type Theories (MTTs), i.e. Type Theories within the tradition of Martin-Löf (1975); Martin-Löf (1984), have become a major alternative to Montague Semantics (MS) in the last twenty years. A number of influential approaches using MTTs have been proposed throughout this period (Ranta 1994; Luo 2011; Retoré 2014; Cooper *et al.* 2014), showing that the rich typing system offered by these approaches (type many-sortedness, dependent types, type universes among other things) has considerable advantages over simple typed systems predominantly used in mainstream formal semantics. A further important aspect for considering the use of MTTs over traditional Montagovian frameworks concerns the proof-theoretic nature of the former but not of the latter.[1] This latter fact makes MTTs a suited formal semantics language to perform reasoning tasks, as these are exemplified for example in work on inference using proof-assistant technology (Chatzikyriakidis and Luo 2014b,a; Bernardy and Chatzikyriakidis 2017). However, this expressiveness of typing comes with a cost. For example, how does one decide on the base types to be represented? On the one hand, we do have a way to get a more fine-grained type system unlike the monolithic domain of entities found in MS, but on the other hand, constructing such a type ontology is not at all a straightforward and easy task. Different approaches and assumptions have been put forward w.r.t this issue. For example Luo (2011, 2012); Chatzikyriakidis and Luo (2017b) proposed to treat CNs as types, in effect arguing that every CN is a type (roughly a one to one correspondence between common nouns and types). Approaches like Retoré (2014) on the other hand, take a more moderate view and build their typing ontology according to classifier systems, i.e. the intuitions for deciding which types are to be represented or not are taken from classifier systems found in a number of natural languages. On the other hand, work in lexical-semantic networks have provided us with structured lexicons specifying elaborate

[1] At least in the way it is employed in the Montagovian setting, simple type theory can be viewed as model theoretic. However, there is interesting work on the proof theory of simple type theory. The higher order theorem prover LEO-II Benzmüller *et al.* (2007) is an example of such work. We are grateful to an anonymous reviewer for pointing this out to us.

lexical and semantic relations. A classic such case is e.g. WordNet Fellbaum (1998). A very promising line of research in lexico-semantic network construction concerns networks which are built collaboratively by using Games with a Purpose (GWAPs). This is the case of the Lexical Network JeuxDeMots (JDM) (Lafourcade 2007b). JDM is constructed through many GWAPs along with a contributive tool (Diko) which allows players/users to contribute directly and to browse the knowledge base.

Given this background, what we want to propose in this paper is the grounding of our semantic ontologies, as well as any other information needed in order to perform reasoning tasks using MTT semantics, in JDM. In order to do this, we present some first thoughts on how such an endeavour can be accomplished by looking at the way a translation procedure from JDM to MTTs can be performed. Issues to be discussed include the domain of base types, instances of these types, adjectival and verbal types, hyponymy/hypernomy relations, as well as more advanced issues like homophony and polysemy. We then show how one can exploit this translation procedure by extracting this information from JDM in order to feed a reasoning device that implements an MTT. We show some easy cases of inference that are taken care of via a combination of the lexical semantics information extracted from JDM and the proof theoretic power of MTTs (performed by the proof-assistant Coq) and further show how JDM can actually help us in order to reason with cases where reasoning with implicit premises is at play. The structure of the paper is as follows: in Section 2, the JDM project is described as well as the produced lexical network. In Section 3, we describe two main endogenous inference mechanisms (deductive and inductive scheme), followed by a discussion on the annotation of relations between terms. Then, in Section 4, we discuss the building of type ontologies using information from JDM and propose a number of translation procedures between JDM and an MTT. The section also includes a brief intro to MTT semantics, highlighting aspects of the theory that will play a role in this paper the most. Lastly, in Section 5 we look at the possibility of performing natural language inference tasks using MTT semantics powered by information drawn from JDM. We present a number of inference cases that rely mostly on lexical-semantic information taken by JDM and the proof-theoretic power of MTT semantics using the proof-assistant Coq.

2 BUILDING A LEXICAL NETWORK

JeuxDeMots[2], a project launched in September 2007, aims to build a large lexico-semantic network (Lafourcade 2007a). The network is composed of terms (nodes or vertices) and typed relations (links between nodes). It contains terms and possible refinements in the same spirit as WordNet synsets (Miller 1995), although being organized as decision trees. There are more than 80 different relation types which occurrences are directed, weighted, and possibly annotated (Lafourcade *et al.* 2015).

2.1 *GWAPs*

The game JeuxDeMots is a two player GWAP (Game With A Purpose, see von Ahn and Dabbish 2008), where people are supposed to *earn and collect* words. The main mechanism whereby this goal is achieved is the provision of lexical and semantic associations to terms proposed by the system.

When a Player (let's call him/her A) starts a game, a term T, along with some instructions concerning the type of lexical relation (e.g. synonym, antonym, domain, etc.), is displayed. The term T could have been chosen from the database by the system or offered to be played by other players. Player A has a limited amount of time (around 60 seconds) to enter terms which, to his/her mind, are relevant w.r.t. both the term T and the lexical relation. The maximum number of terms a player can enter is limited, thus encouraging the player to think carefully about his/her choices. A screenshot of the user interface is shown in Figure 1.

The very same term T, along with the same set of instructions, will be later given to another player, Player B, for whom the process is identical. In order to make the game more entertaining, the two players score points for words they both choose. Score calculation is explained in Lafourcade (2007a) and was designed to increase both precision and recall in the construction of the database. The more 'original' a proposition given by both players is, the more it is rewarded. Figure 2 shows an end of game with collected rewards. Answers given by both players are displayed and those common to both players, as well as their scores, are highlighted.

[2] http://www.jeuxdemots.org

Figure 1: Screenshot of an ongoing game with the target verb *fromage* (cheese). Several propositions have been given by the user and are listed on the right hand side

Figure 2: Screenshot of the game result with the target noun *fromage*. Proposals of both players are displayed, along with points won by both

For a target term T, common answers from both players are inserted into the database. Answers given by only one of the two players are not, thus reducing noise and the chance of database corruption. The semantic network is, therefore, constructed by connecting terms by typed and weighted relations, validated by pairs of players. These relations are labeled according to the instructions given to the players and weighted according to the number of pairs of players who choose them. Initially, prior to putting the game online, the database was populated with nodes. However if a pair of players suggests a non-existing term, the new node is added to the database.

In the interest of quality and consistency, it was decided that the validation process would involve anonymous players playing together. A relation is considered valid if and only if it is given by at least one pair of players. This validation process is similar to that presented by von Ahn and Dabbish (2004) for the indexing of images, by Lieberman *et al.* (2007) and von Ahn *et al.* (2006) to collect common sense knowledge, and Siorpaes and Hepp (2008) for knowledge extraction. As far as we know, this technique has never been used for building semantic networks. Similar Web-based systems already exist in NLP, such as Open Mind Word Expert (Mihalcea and Chklovski 2003), which aims to create large sense-tagged corpora with the help of Web users, and SemKey Marchetti *et al.* (2007), which makes use of WordNet and Wikipedia to disambiguate lexical forms referring to concepts, thus identifying semantic keywords.

For the design of JeuxDeMots, we could have chosen to take into account all of the players' answers according to their frequency from the very outset. The database would have grown much quicker this way, but to the detriment of quality. The rationale behind this choice was to limit the impact of fanciful answers or errors due to misinterpreted instructions or terms. The integration of rarer terms and expressions is slower; nevertheless, these terms are added to the database eventually, once the more common solutions have been exhausted, thanks to the process of creating *taboo* terms. Once a relation with term T has been proposed by a large number of pairs of players, it becomes taboo. During a game, taboo terms are displayed along with term T, discouraging (but not forbidding) players from entering them. In this way, players are encouraged to make other, more original choices. Therefore, more infrequent terms eventually find

their way into the database, and the chances of error are reduced to a minimum.

Even if a relation becomes taboo, its weight can, and does, evolve. However, this tends to be done slowly as the relation is proposed to the players less often. It is important to allow relation weights to continue to evolve, as we can hardly consider such a relation as complete. Eventually, a given term can become taboo when involved in several different relation types. The fact that taboo relations continue to evolve is essential, otherwise the weights of two given relations could become equal and then information about the relative strength relations would be lost.

The approach presented here complements that developed by Zock and Bilac (2004) and Zock and Schwab (2008) who tried to create an index based on the notion of association to assist users in navigating the Web or elsewhere, or to help a person find a word on the tip of their tongue. Their approach is bottom-up, i.e. the terms are known (based on word proximity in corpora), but the nature of the link isn't. This has to be inferred, which is far from an easy task. In our case, we provide one of the two terms, term T as well as the relation type. It is the target terms which interest us. Our approach is top-down.

Some other games[3] complement the main game of JDM. Their purpose is to cross validate the information collected in the main game, or to accelerate the relation harvesting for some specific types of relations. For instance, there are games for collecting word polarity (positive, negative, and neutral), for sentiments associated with words, guessing games, sorting games, location preposition games, and so on.

Since September 2007, around 1.5 million matches have been played for JDM, a total of 25 000 hours of cumulative playing. More than 250 million matches have been played for the other games of the JDM platforms.[4]

2.2 *Direct crowdsourcing*

Playing games in order to fill the lexical network is a kind of indirect crowdsourcing, where people (players) do not negotiate their contri-

[3] http://imaginat.name/JDM/Page_Liens_JDMv1.html

[4] http://www.jeuxdemots.org/jdm-about.php

bution beforehand. In some cases, direct crowdsourcing (with negotiation between contributors) is desirable. Indeed, some lexical relation might be complicated enough to be playable without some linguistic knowledge. This is for example the case for TELIC ROLE, which is the goal/purpose of an object (or action). For instance, a *butcher knife* has the telic role of *cutting meat*. It is to be differentiated from the INSTRUMENT of a predicate, which indicates what can be done with the object. A butcher knife could be used to *stab someone*, but this is not its telic role.

In some other cases (depending on each term), a given relation might not be productive enough to be playable. For example, the CAN PRODUCE relation for *cow* could reasonably be *milk*, but there are not many other answers.

All theses considerations lead to the need of a more direct crowdsourcing interface. The Diko[5] service allows to visualize and contribute to the JDM lexical network. A voting mechanism is at the core of the validation (or invalidation) of proposed relations between terms.

2.3 *Inside the JDM Lexical Network*

As mentioned above, the structure of the lexical network we are building relies on the notions of nodes and relations between nodes, as it was initially introduced in the end of 1960s by Collins and Quillian (1969), developed in Sowa and Zachman (1992), used in the small worlds by Gaume *et al.* (2007), and more recently clarified by Polguère (2014). Every node of the network is composed of a label (which is a term or an expression, or potentially any kind of string) grouping together all of its possible meanings.

The relations between nodes are typed. Each type corresponding to specific semantics that could be more or less precise. Some of these relations correspond to lexical functions, some of which have been made explicit by Mel'cuk and Zholkovsky (1988) and Polguère (2003). We would have liked our network to contain all the lexical functions defined by Mel'cuk, but, considering the principle of our software, JDM, this is not viable. Indeed, some of these lexical functions are too specialized and typically aim at some generative procedure (instead of

[5] http://www.jeuxdemots.org/diko.php

automatic text analysis and understanding), as in our case. For example, we can consider the distinction between the Conversive, Antonym, and Contrastive functions, a distinction that could be made through annotations for a quite generic antonym relation. Mel'cuk also considers function refinements, with lexical functions characterized as "wider" or "narrower". Given that JDM is intended for users who are "simple Internet users" and not necessarily experts in linguistics, such functions could be wrongly interpreted. Furthermore, some of these functions are clearly too poorly lexicalized, that is, very few terms feature occurrences of such relations. This is, for example, the case of the functions of 'Metaphor' or 'Functioning with difficulty'.

JDM has a predefined list of around 80 relation types, and players cannot define new types by themselves. These types of relations fall into several categories:

• Lexical relations: synonymy, antonymy, expression, lexical family. These types of relations are about vocabulary and lexicalization.

• Ontological relations: generic (hyperonymy), specific (hyponymy), part of (meronymy), whole of (holonymy), mater/substance, instances (named entities), typical location, characteristics and relevant properties.

• Associative relations: free associations, associated feelings, meanings, similar objects, more and less intense (Magn and anti-Magn). These relations are rather about subjective and global knowledge; some of them can be considered phrasal associations.

• Predicative relations: typical agent, typical patient, typical instrument, location where the action takes place, typical manner, typical cause, typical consequence etc. These relations are about types of relations associated with a verb (or action noun) as well as the values of its arguments (in a very wide sense).

Some relation types are specific to some noun classes. For example, for a noun referring to an intellectual piece of work (book, novel, movie, piece of art, etc.), the relation of *author* is defined. In case of a medical entity, targets and symptoms are defined.

Some outgoing relations for the French word *fromage* are shown below:

fromage → r_associated 800 → lait
fromage → r_associated 692 → camembert
fromage → r_associated 671 → chèvre

fromage → r_associated 580 → vache
fromage → r_associated 571 → gruyère
fromage → r_associated 460 → brebis
fromage → r_associated 419 → roquefort
fromage → r_isa 310 → produit laitier
fromage → r_associated 257 → produit laitier
fromage → r_associated 221 → brie
fromage → r_hypo 214 → gruyère
fromage → r_meaning 205 → produit laitier
fromage → r_hypo 204 → brie
fromage → r_associated 201 → dessert
fromage → r_associated 201 → fromage blanc
fromage → r_locution 199 → fromage de brebis
fromage → r_patient-1 199 → manger
fromage → r_locution 195 → fromage de tête
fromage → r_hypo 189 → fromage blanc
fromage → r_isa 189 → aliment
fromage → r_raff_sem 183 → fromage > produit laitier
fromage → r_isa 182 → ingrédient
fromage → r_lieu 182 → pizza
fromage → r_carac 180 → puant
fromage → r_sentiment 177 → envie
fromage → r_consequence 173 → puer du bec
fromage → r_holo 171 → pizza
fromage → r_associated 168 → laitage
fromage → r_hypo 167 → fromage de vache
fromage → r_hypo 163 → fromage double crème
fromage → r_hypo 163 → fromage à pâte pressée cuite
fromage → r_part_of 163 → lipide
fromage → r_part_of 161 → croûte
fromage → r_lieu :160 → plateau à fromage
fromage → r_carac 160 → odorant
fromage → r_associated#0:154 → raclette
fromage → r_locution :154 → dommage fromage
fromage → r_associated 149 → cancoillotte
fromage → r_locution 148 → faire tout un fromage
fromage → r_locution :148 → fromage analogue
fromage → r_locution :148 → fromage de synthèse

fromage → r_hypo 148 → fromage à pâte dure
fromage → r_similar 148 → substitut de fromage
fromage → r_hypo#8:147 → emmental
…

2.4 *Refinements*

Word senses (or usages) of a given term T are represented as standard nodes T>glose$_1$, T>glose$_2$, ..., T>glose$_n$ which are linked with REFINE(ment) relations. Glosses are terms that help the reader to identify the proper meanings of the term T. For example, consider the French term *frégate (Eng. frigate)*:

- *frégate*→REFINE→*frégate>navire*
  - *frégate>navire* →REFINE→ *frégate>navire>ancient*
  - *frégate>navire* →REFINE→ *frégate>navire>modern*
- *frégate*→REFINE→*frégate>bird*

A frigate can be a ship or a bird (both English and French have the same ambiguity for this word), and as a ship it can either be an ancient ship (with sails) or a modern one (with missiles and such). As can be seen in the above example, word refinements are organized as a decision tree, which can have some advantages over a flat list of word meanings for lexical disambiguation.

A given word sense is treated as any standard term; it can be played regularly. The general polysemous term contains (in principle) the union set of all possible relations given by the senses. In practice, we proceed the other way around, trying to distribute relations from the appropriate term to the proper senses.

2.5 *Negative relations*

A given relation is weighted, and the weight could be negative. A negative weight is only the result of some contributive process (i.e. it is never an outcome of the games) where volunteers add information to the lexical network. The purpose of negative weights is to give some foundation to the inhibitory process that allows us to reject (instead of select) some given meaning during a Word Sense Disambiguation task.

- *frégate>navire* →REFINE→ *coque (Eng. hull)*
- *frégate>bird* →REFINE$_{<0}$→ *coque*

Consider the sentence (in English): *The frigate had her hull breached.* Obviously, the negative relations immediately forbid the frigate from being a bird in this sentence. Hence, negative relations are of primary interest for representing contrastive phenomena among the various senses of a given term.

2.6 *Aggregate nodes*

The JDM lexical network also contains aggregate nodes that are inferred from the set of relations produced by players and contributors. An aggregate (node) is a node that encompasses either:

- a predicate (a verb) + one argument, like for example:
  lion [AGENT] *eat*,
  eat [PATIENT] *salad*.

- a noun + one feature, like for example:
  cat [CARAC] *black*,
  cat [LOCATION] *sofa*,
  rabbit [MADE-OF] *chocolate*.

Aggregates can be combined recursively, for example (parentheses are given for for the purpose of readability):

A :: (*cat [CARAC] black*) [AGENT] *eat*
B :: (*cat [CARAC] black*) [AGENT] (*eat [PATIENT] mouse*)

The motive of such aggregate nodes is to associate information (through relations) with some contextualized items:

- *A* →PATIENT→ *bird*
- *B* →LOCATION→ *garden*

The choice of aggregate node depends on the weight of the relations in the lexical network. An automated process will randomly select some relations and propose them as the aggregate to the players. Those which are selected for playing are dubbed as interesting and reified (instantiated as node) in the lexical network. For example, the relation:

soldier →AGENT→ *kill*
it could lead to the aggregated node:
soldier [AGENT] *kill*

can be proposed to player with various relation types to fill, as PATIENT, LOCATION, MANNER, INSTRUMENT, etc.

2.7 *Some figures*

By February 2017, the JDM lexical network contained roughly 67 million relations between more than 1 million nodes. Around 24 000 terms are refined into 65 000 word senses (word usages). More than 800 000 relations are negative and can be used as inhibitory items. The generic 'associated ideas' relations represent around 25% of the relation total. Annotations (see below) represent around 4.5% of the total. Informational relations (like part-of-speech, some conceptual values like HUMAN, ALIVE, PLACE, SUBSTANCE, ARTIFACT, etc.) stand for 20%.

3 INFERRING AND ANNOTATING RELATION

Inference is the process of proposing new relations on the basis of the actual contents of the network. Simple procedures tend to provide correct but mostly irrelevant results. In Sajous *et al.* (2013) an endogenous enrichment of Wiktionary is done with the use of a crowdsourcing tool. A similar approach of using crowdsourcing has been considering by (Zeichner *et al.* (2012)) for evaluating inference rules that are discovered from texts.

In what follows, we describe two endogenous inference mechanisms which assist the annotation spreading, although other schemas are running in the inference engine, producing new relations and deriving benefit from the produced annotations (Zarrouk 2015).

3.1 *Inference*

In order to increase the number of relations inside the JDM network, an inference engine proposes relations to be validated by other human contributors (or experts in the case of specialized knowledge). The core ideas about inferences in our system are the following:

- as far as the engine is concerned, inferring is deriving candidate conclusions (in the form of relations between terms) from previously known ones (existing relations);
- candidate inferences may be logically blocked regarding the presence or absence of some other relations;

- candidate inferences can be filtered out on the basis of a strength evaluation.

3.1.1 Deductive scheme

The first type of inference we are describing is the deduction or top-down scheme, which is based on the transitivity of the ontological relation *is-a* (hypernym). If a term A is a kind of B and B holds a relation R with C, then we can expect that A holds the same relation with C. The scheme can be formally written as follows:

$$(1) \qquad A \xrightarrow{is-a} B \quad \wedge \quad B \xrightarrow{R} C \quad \Rightarrow \quad A \xrightarrow{R} C$$

If we consider a term T with a set of weighted hypernyms, for each hypernym, the inference engine deduces a set of inferences. Those inference sets are not disjoint in the general case, and the weight of a proposed inference in several sets is the incremental geometric mean of each occurrence.

Of course, the scheme above is far too naive, especially considering the resource we are using. Indeed, B may be, possibly, a polysemous term and ways to block inferences that are certainly wrong can be devised. If there are two different meanings of term B that hold between the first and the second relation (Figure 3), then the inference is most likely wrong.

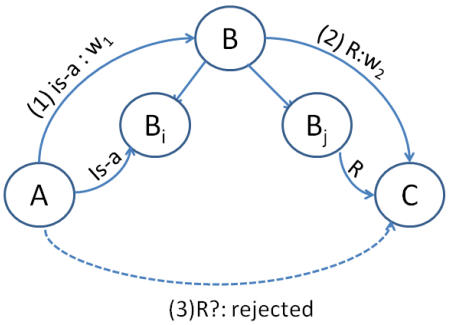

Figure 3: Triangular inference scheme with logical blocking based on the polysemy of B

Moreover, if one of the premises is tagged as *true but irrelevant*, then the inference is blocked. It is possible to assess a confidence level for each produced inference in a way that dubious inferences can be filtered out. The weight *w* of an inferred relation is the geometric mean of the weight of premises. If the second premise has a negative value,

the weight is not a number and the proposal is discarded. As the geometric mean is less tolerant of small values than the arithmetic mean, inferences which are not based on two valid relations (premises) are unlikely to go through.

3.1.2 Induction scheme

As for the deductive inference, induction exploits the transitivity of the relation *is-a*. If a term B is a hypernym of A and A holds a relation R with C, then we might expect that B could hold the same type of relation with C.

(2) $$A \xrightarrow{is-a} B \quad \wedge \quad A \xrightarrow{R} C \quad \Rightarrow \quad B \xrightarrow{R} C$$

This schema is a generalization inference. The global processing is similar to the one applied to the deduction scheme and similarly some logical and statistical filtering may be undertaken. The term joining the two premises is possibly polysemous. If the term A presents two distinct meanings which hold respectively of the premises (Figure 4), then the inference done from that term may be probably wrong.

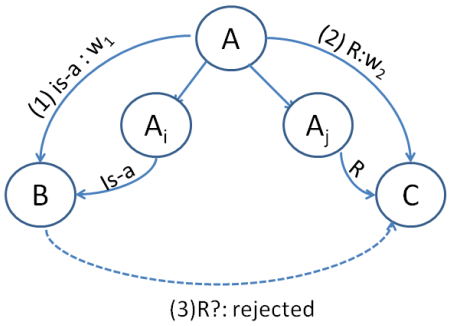

Figure 4:
Induction scheme. Central Term A may be polysemous with meanings holding premises, thus inducing a probably wrong relation

3.2 *Relation annotations*

JDM is a combined lexical semantic network (i.e one containing both general knowledge but also specialist domain knowledge). Besides being typed, relations are weighted and directed. In general, and especially in cases of specialized knowledge, the correlation between the weight of the relation and its importance is not strict. This is why it seems interesting to introduce annotations for some relations as these can be of great help in such areas as medicine, for instance.

In information retrieval, this annotation can be helpful to the users. For instance, in the field of medicine, practitioners may want to know if the characteristic of a given pathology is rare or frequent. For example, the relation between *measles* and *children* is *frequent* and as such will be available in the network.

3.3 *Annotation values*

These annotations will have a filter function in the inference scheme. The types of annotations are of various kinds (mostly frequency and relevance information). The different main annotation labels are:

- frequency annotations: very rare, rare, possible, frequent, always true;
- usage annotations: often believed true, language misuse;
- quantifier: any number like 1, 2, 4, etc. or many, few;
- qualitative: pertinent, irrelevant, inferable, potential, preferred.

Concerning language misuse, a doctor can use the term flu (illness) instead of virus of influenza: it is a misuse of language as the doctor makes use of a "language shortcut". The annotation often believed true is applied to a wrong relation. This is very often considered true, for instance, *spider* (is-a/often believed true) *insect*. This kind of annotation could be used to block the inference scheme. Qualitative annotation relates to the inferable status of a relation, especially concerning inference. The pertinent annotation refers to a proper ontological level for a given relation. For instance: *living being* (charac/pertinent) *alive* or *living being* (can/pertinent) *die*. Another case concerns synonyms: in this case, it may be relevant to choose a preferred synonym, as in the case of hepatocellular *carcinoma* (preferred), *HCC*, *malignant hepatoma*.

The annotation **inferable** is used when a relation is inferable (or has been inferred) from an already existing relation. For example: *cat* (charac/inferable) alive because *cat* (is-a) *living being*.

The annotation **potential** may be used for terms above the pertinent ones in the ontological hierarchy, for example: *bird* (has-part/always true) *wings* and *animal* (has-part/potential) *wings*.

Finally, the annotation **irrelevant** is used for a valid relation that is considered as too far below the pertinent level, for example, *animal* (has-part/irrelevant) *atoms*.

The annotation **quantifier** represents the number of parts of an object. Each human has two lungs so the quantifier relation there is 2. This kind of annotation is not necessarily a numeral, but can be more or less a subjective value, e.g. *few, many*, etc.

The annotation **frequency** can be of five different types: *always true, frequent, possible, rare and exceptional.* There are also two qualitative types (*pertinent and irrelevant*).

The first annotations have been introduced manually, but with the help of the inference scheme, they will spread through the network. We assign empirical values to each annotation's label: 4 to always true, 3 to frequent, 2 to possible, 1 to rare and 0 to the rest of the annotations. These allow us to select annotations to facilitate or block an inference scheme.

The annotation **possible** is a special case. Depending of the configuration of the system, it may block (stricter approach) or not block (lenient approach) the inference mechanism. If a system is lenient, we may obtain many inference proposals that might be wrong (high recall, low precision). On the other hand, if a system is strict, we reduce the risk of wrong proposals, but at the cost of missing adequate ones (low recall, high precision).

4 FROM JDM TO MTTS

In this section, we show how we can exploit the richness of the lexico-semantic information found in JDM, in order to decide on the typing ontology and assign types to objects in a compositional semantics framework that is richly typed. But before we get into this discussion, a very brief intro to MTT semantics.

4.1 *A gentle and brief intro to MTT semantics*

We use the term Modern Type Theory (MTT) to refer to a variant of a class of type theories as studied by Martin-Löf (1975); Martin-Löf (1984) and others, which have dependent types, inductive types and other powerful and expressive typing constructions. In this paper, we are going to employ one of these variants, namely the Unified Theory of dependent Types (UTT) complemented with the coercive subtyping mechanism (Luo 1994, 1999; Luo *et al.* 2012). Given the different typing constructions found in MTTs, various interpretations of linguistic

semantics might be different than what we usually find in traditional Montagovian formal semantics based on simple type theory.

4.1.1 Common nouns as types and subtyping

A key difference between MTT-semantics and Montague semantics (MS) lies in the interpretation of common nouns (CNs). In Montague (1974), the underlying logic, i.e. Church's simple type theory (Church 1940), is 'single-sorted' in the sense that there is only one type, e, of all entities. The other types such as the type of truth values, i.e. t, and the function types generated from types e and t do not stand for types of entities. Thus, no fine-grained distinctions between the elements of type e exist, and as such all individuals are interpreted using the same type. For example, *John* and *Mary* have the same type in simple type theory, i.e. the type e of individuals. An MTT, on the other hand, can be regarded as a 'many-sorted' logical system in that it contains many types. In this respect, MTTs can make fine-grained distinctions between individuals and use those different types to interpret subclasses of individuals. For example, one can have *John* : *man* and *Mary* : *woman*, where *man* and *woman* are different types. Another very basic difference between MS and MTTs is that common nouns in MTTs (CNs) are interpreted as *types* (Ranta 1994) rather than sets or predicates (i.e., objects of type $e \rightarrow t$) as in MS. The CNs *man, human, table* and *book* are interpreted as types *man, human, table* and *book*, respectively. Then, individuals are interpreted as being of one of the types used to interpret CNs.

This many-sortedness has the welcome result that a number of semantically infelicitous sentences, which are however syntactically well-formed, like e.g. *the ham sandwich walks* can be explained easily. This is because a verb like *walks* will be specified as being of type *Animal* → *Prop* while the type for *ham sandwich* will be *food* or *sandwich*:[6]

[6] This is of course based on the assumption that the definite NP is of a lower type and not a Generalized Quantifier. Furthermore, the idea that common nouns should be interpreted as types rather than predicates has been argued in Luo (2012) on philosophical grounds as well. There, Luo argues that the observation found in Geach (1962) according to which common nouns, in contrast to other linguistic categories, have criteria of identity that enable them to be compared, counted or quantified, has an interesting link with the constructive notion of

(3) *the ham sandwich* : *food*

(4) *walk* : *human* → *Prop*

Interpreting CNs as types rather than predicates has also a significant methodological implication: compatibility with subtyping. For instance, one may introduce various subtyping relations by postulating a collection of subtypes (physical objects, informational objects, eventualities, etc.) of the type *Entity* (Asher 2012). It is a well-known fact that if CNs are interpreted as predicates as in the traditional Montagovian setting, introducing such subtyping relations would cause problems given that the contravariance of function types would predict that if $A < B$, then $B \rightarrow Prop < A \rightarrow Prop$ would be the case. Substituting A with type *man* and B with type *human*, we come to understand why interpreting CNs as predicates is not a good idea if we want to add a coercive subtyping mechanism.

The subtyping mechanism used in the MTT endorsed in this paper is that of coercive subtyping (Luo 1999; Luo *et al.* 2012). Coercive subtyping can be seen as an abbreviation mechanism: A is a (proper) subtype of B ($A < B$) if there is a unique implicit coercion c from type A to type B and, if so, an object a of type A can be used in any context $\mathfrak{C}_B[\_]$ that expects an object of type B: $\mathfrak{C}_B[a]$ to be legal (well-typed) and equal to $\mathfrak{C}_B[c(a)]$.

To give an example: assume that both *man* and *human* are base types. One may then introduce the following as a basic subtyping relation:

(5) *man* < *human*

4.1.2 Σ-types, Π-types and universes

In this subsection, the dependent types Σ and Π. as well as universes are briefly introduced.

Dependent Σ-types. One of the basic features of MTTs is the use of Dependent Types. A dependent type is a family of types that depend

set/type: in constructive mathematics, sets (types) are not constructed only by specifying their objects but they additionally involve an equality relation. The argument is then that the interpretation of CNs as types in MTTs is explained and justified to a certain extent. Extensions and further theoretical advances using the CNs as types approach can be found in Chatzikyriakidis and Luo (2017b).

on some values. The constructor/operator Σ is a generalization of the Cartesian product of two sets that allows the second set to depend on the values of the first. For instance, if *human* is a type and *male* : *human* → *Prop*, then the Σ-type Σh : *human*. *male*(h) is intuitively the type of humans who are male.

More formally, if A is a type and B is an A-indexed family of types, then $\Sigma(A,B)$, or sometimes written as $\Sigma x{:}A.B(x)$, is a type, consisting of pairs (a,b) such that a is of type A and b is of type $B(a)$. When $B(x)$ is a constant type (i.e., always the same type no matter what x is), the Σ-type degenerates into the product type $A \times B$ of non-dependent pairs. Σ-types (and product types) are associated projection operations π_1 and π_2 so that $\pi_1(a,b) = a$ and $\pi_2(a,b) = b$, for every (a,b) of type $\Sigma(A,B)$ or $A \times B$.

The linguistic relevance of Σ-types can be directly appreciated once we understand that in their dependent case Σ-types can be used to interpret linguistic phenomena of central importance, like adjectival modification (see for example Ranta 1994). To give an example, *handsome man* is interpreted as Σ-type (6), the type of handsome men (or more precisely, of those men together with proofs that they are handsome):

(6) Σm : *man*. *handsome*(m)

where *handsome*(m) is a family of propositions/types that depends on the man m.

Dependent Π*-types.* The other basic constructor for dependent types is Π. Π-types can be seen as a generalization of the normal function space where the second type is a family of types that might be dependent on the values of the first. A Π-type degenerates to the function type $A \rightarrow B$ in the non-dependent case. In more detail, when A is a type and P is a predicate over A, $\Pi x{:}A.P(x)$ is the dependent function type that, in the embedded logic, stands for the universally quantified proposition $\forall x{:}A.P(x)$. For example, the following sentence (7) is interpreted as (8):

(7) Every man walks.

(8) Πx : *man*.*walk*(x)

Π-types are very useful in formulating the typings for a number of linguistic categories like VP adverbs or quantifiers. The idea is that

adverbs and quantifiers range over the universe of (the interpretations of) CNs and as such we need a way to represent this fact. In this case, Π-types can be used, universally quantifying over the universe CN. Example (9) is the type for VP adverbs[7] while (10) is the type for quantifiers:

(9) $\Pi A : \mathrm{CN}.\ (A \to \mathit{Prop}) \to (A \to \mathit{Prop})$

(10) $\Pi A : \mathrm{CN}.\ (A \to \mathit{Prop}) \to \mathit{Prop}$

Further explanations of the above types are given after we have introduced the concept of type universe below.

Type Universes. An advanced feature of MTTs, which will be shown to be very relevant in interpreting NL semantics in general, is that of universes. Informally, a universe is a collection of (the names of) types put into a type (Martin-Löf 1984).[8] For example, one may want to collect all the names of the types that interpret common nouns into a universe CN : **Type**. The idea is that for each type A that interprets a common noun, there is a name $\overline{A}$ in CN. For example,

$$\overline{[\![man]\!]} : \mathrm{CN} \quad \text{and} \quad T_{\mathrm{CN}}(\overline{[\![man]\!]}) = [\![man]\!].$$

In practice, we do not distinguish a type in CN and its name by omitting the overlines and the operator T_{CN} by simply writing, for instance, $[\![man]\!] : \mathrm{CN}$.

Having introduced the universe CN, it is now possible to explain (9) and (10). The type in (10) says that for all elements A of type CN, we get a function type $(A \to \mathit{Prop}) \to \mathit{Prop}$. The idea is that the element A is now the type used. To illustrate how this works let us imagine the case of the quantifier *some* which has the typing in (10). The first argument we need has to be of type CN. Thus *some human* is

[7] This was proposed for the first time in Luo (2011).

[8] There is quite a long discussion on what properties these universes should have. In particular, the debate is largely concentrated on whether a universe should be predicative or impredicative. A strongly impredicative universe U of all types (with $U : U$ and Π-types) has been shown by Girard (1971) to be paradoxical, and as such logically inconsistent. The theory UTT we use here has only one impredicative universe Prop (representing the world of logical formulas) together with infinitely many predicative universes which as such avoids Girard's paradox (Luo 1994).

of type (*human* → *Prop*) → *Prop* given that the *A* here is *human* : CN (*A* becomes the type *human* in (*human* → *Prop*) → *Prop*). Then given a predicate like *walk* : *human* → *Prop*, we can apply *some human* to get *some human* : *Prop*.

4.2 *Getting MTT typings from JDM*

In this section, we will show how we can define a translation procedure between JDM and MTTs, in order to base our typing judgments and other related lexico-semantic information in JDM. We show some basic examples in which this can be done.

4.2.1 Base types and instances of base types

MTTs, as already said, are many-sorted systems in that they involve a multitude of types rather than just one monolithic type *e* domain of entities. In the accounts proposed by Luo (2011, 2012), every common noun is associated with a base type. What this idea amounts to, among other things, is that in this approach, CNs are base types and as such, are clearly separated in terms of their formal status with either adjectives or intransitive verbs. The type of CNs, like *Man*, *Human* and *Animal* is CN, the universe of common nouns.

The idea is then to extract these base types from common nouns in JDM (terms in JDM). POS tagging of JDM will provide information about which words are the common nouns. What we further have to do in getting the base types, is to exclude instances of terms (for example *John* as an instance of *Man*) in order to distinguish between instances of terms and the terms themselves (CNs).[9] This can be done by excluding named entities (NEs). The second part of the conjunction takes care of that by not allowing *A* to be an instance, i.e. an NE:[10]

(11) $\forall A.POS(N,A) \wedge \neg(Ins(A)) \Rightarrow A{:}\mathrm{CN}$.

[9] This does not mean that we are not interested in instances. On the contrary. What we are saying here is that this rule distinguishes between CNs and instances of these CNs (the difference between a type like Man and an instance of this type, e.g. $John$). There will be a separate rule to derive instances.

[10] Note that modified CNs are also going to be of type CN. To give an example, consider the analysis of adjectival modification. In MTTs, this would be a Σ type, where the first component would be an element *A* of type CN and the second projection a predicate over *A*. The first projection is defined as a coercion, and thus the modified CN can be used as being of type CN. For more information on this, please refer to (Chatzikyriakidis and Luo 2013, 2017a) for more information.

Hyponym and hypernym (noted as *isa* in JDM) relations naturally correspond to subtypes and supertypes. We only use the subtype relation in order to provide a translation procedure:

(12) $\forall A, B.\mathit{Hyp}(A, B) \Rightarrow A < B{:}\mathrm{CN}$.

(13) $\forall A, B.\mathit{Hyper}(A, B) \Rightarrow B < A{:}\mathrm{CN}$.

This basically means that as soon as you have, let us say, a hyponym relation, e.g. $\mathit{Hyp}(A, B)$, this will be translated into a type-theoretic judgment of the following form:

(14) $A < B{:}\mathrm{CN}$

If we want to be more meticulous, we first have to judge A and B as being of type CN and then we can further add the subtype relation.

Moving on to synonyms, these can be defined using equality:[11]

(15) $\forall A, B.\mathit{Syn}(A, B) \Rightarrow A = B{:}\mathrm{CN}$.

Synonymicity is not only relevant for CNs but for other linguistic categories. We can encode this intuition as follows:

(16) $\forall A, B.\mathit{Syn}(A, B) \Rightarrow A = B{:}C(\mathit{CLType})$

The above rule can declare synonymous words that have the same type via the equality relation. The type itself belongs in the universe *LType*. *LType* can be seen as a universe of linguistic types. The main intuition is that it includes the types instantiated in linguistic semantics (CN, adjectival and verbal types, types for quantifiers etc.). The interested reader is directed to Chatzikyriakidis and Luo (2012) for more details as well as some of the introduction rules for *LType*).

For instances of terms, such as proper names, we define the following:[12]

(17) $\forall A.\exists B.\mathit{Ins}(A, B) \Rightarrow A{:}B$

This means that if A is an instance of B, then A is of type B. For example, if *Einstein* is an instance of *person*, then what we get is

[11] Of course, this will treat A and B as perfect synonyms. We make this simple-minded assumption in this paper, even though perfect synonyms do not really exist in natural language.

[12] Note that here we overload the notation and sometimes treat *Ins* as an one place predicate and sometimes like a two place predicate.

Einstein:*person* with *person*:CN. In more detail, the procedure is as follows: given an instance *A* of a term *B*, first you declare *B*:CN and then judge the instance *A* to be of that type, i.e. *B*. This is the easy straightforward case and assumes that every instance will be an instance of one term. However, things are more complicated in practice. Given that JDM is a very elaborate lexical network, proper names will be instances of many terms (and thus, in MTT terms, types). To give an example: in the case of a proper name like *Einstein*, what we get is a number of terms from JDM that Einstein is an instance of: *physicist*, *scientist*, *human individual*. The question is which one do we choose. This is not an easy question to answer. One option would be to go for the term that is the most specific. But how do we define this? One way to do this, and given the discussion on relations of hyponymy, is to define it by saying that the term chosen should not have any subtypes in the given entry. For example *individual* will have subtypes (*scientist*, and also *physicist*) *scientist* (*physicist*). In this case, we are left with *physicist*. This is one way to do it. Note that given subtyping, we do get that *Einstein* is also an instance of the supertypes. This is a viable solution provided that all the terms are somehow connected in terms of subtyping. But there might be discontinuous relations. For example, imagine the case of the term *man*. Let us assume that *Einstein* is an instance of this term (surprisingly the term does not arise in JDM). Now, *physicist* and *scientist* are not subtypes of *man*. In this case, it seems that one has to make a decision about the type of *Einstein* based on those two types. It seems to us that in principle one should be able to make use of both types depending on the context. How one disambiguates is another issue however. Another way to do this is to assume that such instances are complex types, and treat them as disjoint union types in type-theoretical terms. Doing so will mean that *Einstein* will be of the following complex type *physicistman*:

(18) $physicistman = physicist + man$

(19) $Einstein{:}physicist + man$

Now, in this situation one can have such a complex case without actually resorting to context. The correct type will be disambiguated according to what is needed. In case a term of type *man* is needed like in *Einstein was a bachelor*, then the type *man* is going to be used. In cases like *Einstein was a well-known physicist*, the type *physicist* is to be

used. Note that this relies on the assumption that a subtyping mechanism is at play which will provide us with the following subtyping relations:

(20) *physicistman* < *man*

(21) *physicistman* < *physicist*

In this context, a more general way of translating these cases into MTTs would be as follows:

$$\forall A, D.\exists B, C.Ins(A,B) \Rightarrow \begin{cases} \text{A:B} & \text{iff } Ins(A,D) \wedge \text{B} < \text{D} \\ \text{A:B} + \text{C}, & \text{iff } Ins(A,C) \wedge Ins(A,D) \rightarrow \neg\ (\text{B} < \text{D} \vee \text{C} < \text{D}) \end{cases}$$

The first case is trivial. The second case says that each A that is an instance of type B has the type $B + C$, in case it is also an instance of C and for every other type D that A is an instance of, it is not the case that either B or C are subtypes of D.

4.2.2 Predicates and world knowledge information

The next question is, how can one extract information on the type of predicates, like for example verbs. JDM provides loads of information with every word, for example characteristics, synonyms, antonyms, collocations. For verbs, *agent*, *patient* and thematic relations in general are defined. This is particularly helpful for a rich type theory like the one used here, since predicates also make use of type many-sortedness. Thus, *walk* will be defined as being a function from type *animal* to propositions, *black* from type *object* to propositions and so on:

(22) *walk*:*animal* → *Prop*

(23) *black*:*object* → *Prop*

The information in JDM is enough to provide MTT typings for predicates as well. In JDM, as already said, one can look at semantic relations like *action*,*patient*, dubbed as predicative relations in the classification given in the previous section, and various other such relations. For example, *man* appears as the agent of a number of verbs that express actions, e.g. *question*. But, the most helpful relation is the inverse agent/theme/patient relation, $agent^{-1}$. This relation returns a list of terms (and instances of terms) that can function as the agent for the action denoted by the verb. For example, the verb *question* will

involve among others *teacher, mother, child, daugther, person, human*. How can we make sense in order to provide typings in MTTs? There is a straightforward way to do this. What we need is to find the most general term, i.e. the term of which all the other terms are hyponyms. Instances of terms are not needed in this process.[13]

(24) $\forall A, B. \exists C. Ag(A,B) \wedge (Hyp(A,C) \vee (A = C)) \Rightarrow B{:}C \rightarrow Prop$

However, things may be less straightforward than that simply because there exists no term in the $agent^{-1}$ relation that is a supertype of all the others. In this case, we see two plausible options: a) introduce a supertype or b) split the refinements into different classes. For example in case we have refinements *human*, *man*, *pilot*, *vehicle*, *car*, *bike*, we can split this into class A = $pilot, man <$ *human* and class $B = bike, car <$ *vehicle* and propose an overloaded polysemous type for the verb in question, with two different typings, *human* → *Prop* and *vehicle* → *Prop*. As far as the supertype is concerned, the suggestion is that we go for a default supertype, which will be the supertype of all types. For example in the work of Chatzikyriakidis and Luo (2014a), this type is *object*. We can think of such a type, no matter whether we agree that this should be type *object* or not (denoted as *Toptype* below). Then, with these considerations in mind, we may want to update the previous correspondence:

$$\forall A, B. Ag(A,B) \Rightarrow \begin{cases} \text{B:C} \rightarrow Prop \text{ iff } \exists \text{ C.}((Hyp(\text{A,C}) \wedge Ag(\text{C,B})) \vee (\text{A} = \text{C})) \\ \text{B:}D \rightarrow Prop \text{ iff } \neg \exists C. (Hyp(A,C) \wedge D = TopType) \end{cases}$$

The first condition says that if A is the agent of predicate B (expressed by an adjective here), then in case there is a type C that is also the agent of B, it is a supertype of A as well as a supertype of all other types that are agents of B,[14] or if C is actually A, then the type for the adjective will just be a predicate over C. In case there is no hypernym of A, we choose as our type a predicate over the Toptype (the type of which all other types are subtypes).

[13] The formula reads as follows: for all A and B, where A is an agent of B (so B is a predicate), if there exists a C such than all A are either hyponyms of C or are equal to C, the predicate $C \rightarrow Prop$ is returned.

[14] This last bit is not actually encoded in the rule for formatting reasons. The following condition is implicit: ∀ E.*Ag*(E,B)→*Hyp*(E,C).

Moving on to adjectives, similar processes can be defined. However, this time we look at another relation, called *carac* (characteristic) that denotes a characteristic of a term. For example, for $grand$, we find the characteristics *chose* and *homme*, 'object' and 'man' respectively among others. There are two ways to assign types for adjectives here: a) propose a type using the same reasoning for predicates above or b) propose a polymorphic type extending over a universe which includes the most general type found satisfying the characteristic in question (e.g. blackness, bigness, etc.), along with its subtypes:[15]

(25) $\forall A,B.\exists C.Car(A,B) \wedge (Hyp(A,C) \vee (A = C)) \Rightarrow C \rightarrow Prop$

(26) $\forall A,B.\exists C.Car(A,B) \wedge (Hyp(A,C) \vee (A = C)) \Rightarrow \Pi U{:}\mathrm{CN}_C.U \rightarrow Prop$

Using a polymorphic universe in terms of inference will suffice in order to take care of the class of adjectives known as subsective (e.g. *skilful*), while for intersective adjectives (e.g. *black*) a non-polymorphic type is needed. This, along with the use of Σ types for modified, by adjectives, CNs (e.g. *black man*, *skillful surgeon* etc.) will suffice to take care of the basic inferential properties of the two classes of adjectives Chatzikyriakidis and Luo (2013, 2017a) for more details on this analysis). However, as in the case of verbs, more should be said in order to take care of the complications already discussed for verbs previously. Taking these issues into consideration, the updated rule is as follows:

$$\forall A,B.Car(A,B) \Rightarrow \begin{cases} \mathrm{B{:}C} \rightarrow Prop & \text{iff } \exists\ \mathrm{C.((}Hyp\mathrm{(A,C)} \wedge Ag\mathrm{(C,B))} \vee \mathrm{(A = C))} \\ \mathrm{B{:}}D \rightarrow Prop & \text{iff } \neg\exists C.(Hyp(A,C) \wedge D = TopType) \end{cases}$$

$$\forall A,B.Car(A,B) \Rightarrow \begin{cases} \mathrm{B{:}}\Pi U{:}CN_C\ U \rightarrow Prop & \text{iff } \exists\ \mathrm{C.((}Hyp\mathrm{(A,C)} \wedge Ag\mathrm{(C,B))} \vee \mathrm{(A = C))} \\ \mathrm{B{:}}\Pi U{:}CN_D\ U \rightarrow \mathrm{Prop} & \text{iff } \neg\exists C.(Hyp(A,C) \wedge D = TopType) \end{cases}$$

The above two rules are for intersective and subsective adjectives. Hyponymy relations between adjectives can be encoded as meaning postulates.

(27) $\forall A,B.POS(adj,A) \wedge POS(Adj,B) \wedge Hyp(A,B) \Rightarrow \forall x{:}C.A(x) \rightarrow B(x)$
where $A,B{:}C \rightarrow Prop$

[15] For example the universe $U{:}\mathrm{CN}_C$ will contain the type C along with its subtypes.

Due to the abundance of information that JDM has to offer, one can further encode different sorts of information in the form of axioms or definitions. For example the *has_part* relation, in effect a mereological relation, can be translated as a *part of* relation with *part_of*:*object* → *object* → *Prop* and follow a translation procedure along standard assumptions for mereology for formal semantics[16]. There are many more interesting relations in JDM, like for example the collocation relation, (*locution* in JDM) or the magnifying and its inverse anti-magnifying relation, *magn* and *anti-magn* respectively. Now, there is no clear way of what we can do with these relations. One can of course just encode a similar relation in the type-theoretic language used, but the question is what do we gain in terms of reasoning for example, by doing so. For instance, looking at the entry for *homme* 'man', we see a number of collocations like *homme grande* and *homme libre*. The collocations that involve adjectival modification most of the times give rise to subsective inferences. For example a great man is a man and a free man is also a man. It would be tempting in this respect to treat these cases as subtypes of the term. In this case, we allow some non-compositionality and treat collocations as involving one word. Of course, this will not give us the correct results in all cases. For example, think of the term *objet* 'object'. Among the collocations in the category discussed, e.g. *object du désir* or *objet de curiosité*, there are also collocations like *programmation orientée objet* 'object oriented programming' that will give us the wrong results if we go the subtyping route. One can however decide on whether to take such a stance based on the amount of collocations that can be correctly captured in going the subtype route in relation to those that are not. This is a complex issue with which we will not deal in this paper.

4.2.3 Polysemy

The next issue we want to look at is polysemy, more specifically the translation process in case of polysemous terms. First of all, we have to note here that JDM does not distinguish between homophony and polysemy in the sense these are usualy understood in the literature on formal semantics (e.g. *bank* as homophonous and *book* as polysemous). For JDM, there is only one term to refer to both homophony

[16] For an overview see Champollion and Krifka (2016).

and polysemy, and this is polysemy. This is what we are going to use here as well, a single notion for all cases where different meanings associated with a given word are found. For JDM, there is this first level where words with more than one meaning (irrespective of whether the meanings are related or not) are dubbed as polysemous, and then additional levels of refinement are provided. In MTTs, as in formal semantics in general, there are different treatments with respect to cases of homophony and cases of polysemy. For example, in Luo (2011), homophony is treated via local coercions (local subtyping relations), while logical polysemy (cases like *book*) via introducing dot-types, types that encode two senses that do not share any components (Luo 2010). It is a difficult task to be able to translate from a polysemous term identified in JDM to the correct mapping in MTTs. However, there are some preliminary thoughts on how this can be achieved. First of all, let us look at some cases of polysemy identified in JDM that would not be considered such cases in mainstream formal semantics. For example the term *individual* is marked as polysemous in JDM. The reason for this is that JDM goes into more detail than what most formal semantics theories do. For example, JDM distinguishes different meanings of *individual* with respect to its domain of appearance, e.g. a different notion of individual is found in the domain of statistics, a different one in the domain of biology, and so on. This level of fine-grainedness is usually not found in formal semantics. However, there is no reason why we should not go into this level of detail in MTTs. In order to encode domains, we use type theoretic contexts as these have been used by Ranta (1994); Chatzikyriakidis and Luo (2014c), among others. The idea is that a relation can appear in different domains. If this is the case, then different relations might be at play depending on the domain. For example, different refinements of a term might be possible in a domain A than in a domain B.[17]

(28) $\forall A, C.\mathit{Domain}(C) \wedge \mathit{POS}(N, A)\ \mathit{in}\ C \wedge \neg(\mathit{Ins}(A)) \Rightarrow A{:}\mathrm{CN}\ \text{in}\ \Gamma_C$

[17] An anonymous reviewer asks how does the equation help us in using this information. The idea is that as soon as the conditions are satisfied, i.e. there is a relevant domain for a given type declaration, then this declaration is made inside the relevant type theoretic context, e.g. the context of zoology, philosophy, etc. In the case of Coq, this can be done by introducing local sections.

The above example identifies a noun, which is not an instance, in a domain C and declares this to be of type CN in context Γ_C. All this information such as POS, domain and instance status is part of the JDM network. To give an example, take the term French term *fracture* (fracture) in JDM. This is associated with a number of different domains, let us mention two here, *géologie* and *médicine*. This will basically add the term *fracture* into two different contexts where the relations between *fracture* and other terms in the given context might differ in the different contexts. For example, one might have a term B being a subtype of *fracture* in one domain but not in the other:

What about other cases of polysemy like *book* or *bank*? One way to look at the translation process in these cases is the following: in case a term is dubbed polysemous in JDM, we look at the semantic refinements and introduce all these refinements as subtypes of the initial term:

(29) $\forall A, C.\mathit{POS}(N, A) \wedge \neg(\mathit{Ins}(A)) \wedge \mathit{Ref}(A, C) \Rightarrow A < C{:}\mathrm{CN}$

Now in order to decide whether we are going to use local coercions or dot-types we proceed as follows: the types that participate in dot-types are limited and enumerable:[18] some of these include *phy*, *info*, *event*, *inst* among others. We can thus create such a set of refinements that can be senses of a dot-type. Call this set *dot refinements*, DR. Now, in case the refinements happen to be members of this set then we can form a dot-type out of the individual refinements:

(30) $\forall A, B, C.POS(N, A) \wedge \mathit{Ref}(A(B, C)) \in DR \Rightarrow A{:}CN < B \bullet C$

Other cases of polysemy that should be taken into consideration involve cases where the two meanings are associated with different types (e.g. cases like *run*). In this case, we have at least two verbal meanings with a different verbal arity as well as a different CN argument. An easy way to do this is to just overload the types to take care of situations like these. For example, in Luo (2011), the polysemy of

[18] An anonymous reviewer asks how these types are chosen. This is not an easy question. For the needs of this paper, and given that dot-types have specific properties compared to other polysemous terms, the types comprising the dot-types are limited. We enumerate these types based on existing theoretical work on co-predication by Pustejovsky (1995) and Asher (2008), among others.

run is assumed to be captured using a Unit type which allows us to overload the type with the two different typings:

(31) $run_1 : human \rightarrow Prop$

(32) $run_2 : human \rightarrow institution \rightarrow Prop$

With this last note, we will move on to look how information from JDM can be used in order to perform reasoning tasks. What we are going to do is to look at simple cases of lexical semantics information extraction from JeuxdeMots, their direct translation to MTT semantics feeding the proof-assistant Coq. Reasoning is then performed using the assistant.

5 JDM, MTTS AND REASONING USING PROOF-ASSISTANTS

Coq is an interactive theorem prover (proof-assistant). The idea behind it and proof-assistants in general is simple and can be roughly summarized as follows: one uses Coq in order to check whether propositions based on statements previously pre-defined or user defined (definitions, parameters, variables) can be proven or not. Coq is a dependently typed proof-assistant implementing the calculus of Inductive Constructions (CiC, see Coq 2007). This means that the language used for expressing these various propositions is an MTT. To give a very short example of how Coq operates, let us say we want to prove the following propositional tautology in Coq:

(33) $((P \vee Q) \wedge (P \rightarrow R) \wedge (Q \rightarrow R)) \rightarrow R$

Given Coq's typed nature we have to introduce the variables P, Q, R as being of type *Prop* (P, Q, R:*Prop*). To get Coq into proof mode, we have to use the command *Theorem*, followed by the name we give to this theorem, followed by the theorem we want to prove:

(34) *Theorem A*: $((P \vee Q) \wedge (P \rightarrow R) \wedge (Q \rightarrow R)) \rightarrow R$

This will put Coq into proof mode:

```
Theorem A:((P\/Q)/\(P -> R)/\(Q->R))->R.
1 subgoal
============================
(P\/Q)/\(P -> R)/\(Q -> R)->R
```

Now, we have to guide the prover to a proof using its pre-defined proof tactics (or we can define our own). For the case under consideration, we first introduce the antecedent as an assumption using *intro*:[19]

```
A < intro.
1 subgoal
H :(P \/ Q)/\(P -> R)/\(Q -> R)
============================
R
```

We split the hypothesis into individual hypothesis using *destruct*:[20]

```
destruct H. destruct H0.
1 subgoal
H : P \/ Q
H0 : P -> R
H1 : Q -> R
============================
R
```

Now, we can apply the elimination rule for disjunction which will basically result in two subgoals:

```
elim H.
2 subgoals
H : P \/ Q
H0 : P -> R
H1 : Q -> R
============================
P -> R
subgoal 2 is:
Q -> R
```

The two subgoals are already in the hypotheses. We can use the *assumption* tactic that matches the goal in case an identical premise exists, and the proof is completed:

[19] This tactic moves the antecedent of the goal into the proof context as a hypothesis.

[20] After destructing H, we get $H0$ as $H0{:}(P \rightarrow R) \wedge (Q \wedge R)$.

```
assumption. assumption.
1 subgoal
H : P \/ Q
H0 : P -> R
H1 : Q -> R
============================
Q -> R
Proof completed.
```

Now, as we have already said, Coq implements an MTT. In this respect, Coq 'speaks' an MTT so to say. It is also a powerful reasoner, i.e. it can perform elaborate reasoning tasks. These two facts open up the possibility of using Coq for reasoning with NL using MTT semantics. Indeed, earlier work has shown that Coq can be used to perform very elaborate reasoning tasks with very high precision (Mineshima *et al.*; Bernardy and Chatzikyriakidis 2017). To give an example, consider the case of the existential quantifier *some*. Quantifiers in MTTs are given the following type, where A extends over the CN (this is reminiscent of the type used for VP adverbs):

(35) $\Pi A : \text{CN}.\ (A \rightarrow \textit{Prop}) \rightarrow (A \rightarrow \textit{Prop})$

We provide a definition based on this type, giving rather standard semantics for the existential quantifier (in Coq notation):

```
Definition some:=fun(A:CN)(P:A->Prop)=>exists x:A,P x.
```

This says that given an A of type CN and a predicate over A, there is an $x{:}A$ such that P holds of x. Imagine, now, that we want to see the consequences of this definition. For example we may want to check whether *John walks* implies that *some man walks* or that *some man walks* implies that *some human walks*. We define, following our theoretical assumptions about CNs, man and $human$ to be of type CN and declare the subtyping relation $man < human$. The subtyping relations in Coq are declared by first introducing them as axioms and then coercing them:

```
Parameter man human: CN
Axiom mh: man -> human. Coercion mh: man >-> human.
```

This is all we need to get the above inferences. These assumptions suffice to prove these inferences in Coq. We formulate the theorem and put Coq into proof mode:

```
Theorem EX: walk John-> (some man) walk.
```

Unfold the definition for *some* and use *intro*

```
EX < intro.
1 subgoal
H : walk John
============================
exists x : man, walk x
```

Using the *exists* tactic to substitute x for $John$. Using *assumption* the theorem is proven. Now, what we want to show is that we can actually use JDM to extract lexical and typing information, translate this information into MTT semantics in the form of Coq code and then perform reasoning tasks. Let us look at the following example:

(36) John Fitzgerald Kennedy ate some gruyère

Suppose, now, that we further want to check whether the following is true:

(37) John Fitzgerald Kennedy ate some gruyère ⇒ John Fitzgerald Kennedy ate some cheese

Let us see whether we can extract this information from JDM. We use the JDM XML version, and further use simple Python code to extract the relevant information and turn it into Coq code. We first extract all the synonyms and subtypes of cheese and translate them to MTT semantics (in Coq code). The result is more than 200 subtypes for cheese (*fromage* in French), among them the type for *gruyère*. What the code does is that it first declares all subtypes to be of type CN and then further declares them to be subtypes of the CN in question (cheese in our case. The result is something like this (we use the first 5 subtypes to illustrate this):

```
Parameter gruyere:CN.
Parameter brie:CN.
Parameter kiri:CN.
```

```
Parameter camembert:CN.
Axiom  Gruyere:gruyere -> fromage.
Coercion Gruyere:gruyere>-> fromage.
Axiom Brie:brie->fromage.Coercion Brie:brie>->fromage.
Axiom Kiri:kiri->fromage.Coercion Kiri:kiri>->fromage.
Axiom Camembert:camembert->fromage.
Coercion Camembert:camembert>-> fromage.
```

The next step is to extract information about *John Fitzgerald Kennedy*. The only thing needed here is to extract the information for the instances of the type *man* (*homme* in French). Simple coding in Python can extract all the subtypes for *man* as well as its instances, declaring them as being of type *man*. What we get in doing so is the following (we only show the information relevant to our example):

```
Parameter man: CN.
Parameter John Fitzgerald Kennedy: man.
```

The next step is extracting the information for the verb *eat* (*manger* in French). Here we use a more simple and less elegant way of extracting the function types than we have described in the previous section. We first chose 6 very basic types, *woman, man, human, animal, food, object*. If any of these types is present as an agent argument (starting hierarchically from type *object* and all the way down to the other types), it is added as an argument to the function type. Thus, in case of a predicate which has an object agent, the type *object* → *Prop* is returned. The other types, even if present, are neglected. If *object* is not present, the next type is checked and so on. Doing so, we end up with the type *Animal* → *food* → *Prop* for *eat*. *Cheese* is of course a subtype of *food* (we get this from the hyponyms of *food*), and *human* of *animal*. So, the only thing left is a definition of the quantifier. Quantifiers and related elements can perhaps be assumed to belong to a closed set of words that can be given their semantics manually. This is what we do here by manually providing a definition for *some*. With this in place, what we get is the following information for Coq (only the relevant code to the example is shown):

```
Definition CN:= Set.
Definition some:= fun(A:CN)(P:A->Prop)=>exists x:A,P x.
Parameters man woman human animal food object: CN.
```

```
Axiom  Man:man->human. Coercion Man:man>->human.
Axiom  Human: human->animal.Coercion Human:human>->animal.
Axiom  Animal: animal->object.
Coercion Animal: animal>->object.
Axiom  Food: food->object.Coercion food:food>->object.
Axiom  Woman: woman->human. Coercion Woman:woman>->human.
Parameter gruyere: CN.
Axiom  Gruyere: gruyere->fromage.
Coercion Gruyere:gruyere>-> fromage.
Parameter John_Fitzgerald_Kennedy: human.
```

This is in fact enough to work through the inference we are interested in. Of course, this is not a very elaborate example, but it is a nice way to exemplify how information from a lexical network can be used in a compositional semantics framework to perform reasoning tasks. Note, that a number of other inferences also follow from the previous example:

(38) John Kennedy ate some gruyère ⇒ some man ate some gruyère.

(39) John Kennedy ate some gruyère ⇒ John Kennedy ate some food.

Let us look at another example:

(40) The frigate had its hull breached.

In this example, what we need to predict is that the *bird* sense cannot be used. On the contrary, we should predict that the *ship* sense is required. First of all, the way this is achieved in JDM is via using negative weights as we have mentioned in chapter 2. We will now see that compositional semantics can further help us in this task. We start with the assumption that frigate is not yet refined or can be either a *bird* or a *ship*. The next thing we have is an NP with a possessive pronoun. Following Ranta (1994), we assume a pronominalization and a genitive rule. The two rules are shown below (adapted from Ranta (1994):[21]

$$\frac{A{:}\text{CN} \quad a{:}A}{\begin{cases}\text{PRON(A,a):A} \\ \text{PRON(A,a) = a:A}\end{cases}} \qquad \frac{A{:}\text{CN} \quad B{:}\text{CN} \quad C(x{:}A, y{:}B) \quad a{:}Ab{:}B \quad c{:}C(a,b)}{\begin{cases}\text{Gen(A,B(x,y),C(x,y),a,b,c):B} \\ \text{Gen(A,B(x,y),C(x,y),a,b,c = b):B}\end{cases}}$$

[21] Ranta (1994) uses type Set instead of CN that we are using.

The result of sugaring in English will be *A's B*. The pronominalization rule will depend on the type that A will take. For example $Pron(man, a)$ will return *he*, $Pron(woman, a)$ *she* and $Pron(object, a)$ *it*. Returning to our example, the possessive *its* is a combination of the two rules we have presented, i.e. *Pron* and *Gen*. As we have said, the semantics for pronouns will depend on the value for CN. This is also the case obviously for *its*. Let us assume that A takes the value *frigate*. This will give us:

(41) $Gen(frigate, hull(x, y), C(x, y), Pron(frigate, a), b\ hull, c)$:*hull*.

The C relation is an underspecified relation, since it can take different values, given the semantic polysemy of the genitive. Assuming that the relation involved in our example is one of meronymy, what we get is an elaboration of $C(a, b)$ to $has\_part(a, b)$. Now, notice that JDM provides meronymy relation refinement between two objects, of which one is a ship and the other a hull, but not between a bird and a hull. Specifically, supplies us with the following information (translated into MTT semantics):

(42) $\forall a$:*ship*.$\exists b$:*hull*.$has\_part(a)(b)$

But not:

(43) $\forall a$:*bird*.$\exists b$:*hull*. $has\_part(a)(b)$

Parsing *The frigate had its hull breached*, what we get is the following (simplified):

(44) $breached(the(ship, a))(Gen(ship, hull(x, y), has\_part(x, y),$
$Pron(ship, a), b$:$hull, c))$

Now, we can assume that the negative weight amounts to the negation of the has_part relation:

(45) $\forall a$:*bird*$\neg(\exists b$:*hull*.$(has\_part(a)(b)))$

If now, we substitute with bird and given the information associated with hull as a refinement of bird, what we will get is a contradiction:

(46) $breached(the(ship, a))(Gen(bird, hull(x, y), has\_part(x, y),$
$Pron(ship, a), b$:$hull, c)) \wedge \forall a$:*bird*.$\neg(\exists b$:*hull*.$(has\_part(a)(b)))$

If we have a system that can spot contradictions between information derived from lexical semantics (in our case the negative weight translating into a meaning postulate) and information derived for semantic compositionality, we might use this in order to disambiguate word senses as well. For example, one can define a ranking algorithm that will rank the senses of a given word in a sentence depending on whether they give rise to contradictions between lexical semantics information and information derived for semantic compositionality. In this manner, one can seek to define a combined strategy to disambiguate using insights from both the lexical network itself as well as the formal system in which this information is encoded.

5.1 *Reasoning with missing premises: enthymematic reasoning*

It is a well-known fact that natural language inference (NLI) is not only about logical inference. Better put, logical inference is only part of NLI. Other kinds of non-logical inferencing is going in NLI, e.g. implicatures, presuppositions, or enthymematic reasoning to name a few. The latter form of reasoning is particularly important for the scope of this article, since enthymematic reasoning is basically deriving conclusions from missing premises or implicit premises. Consider the following classic case of an enthymeme:

(47) Socrates is human, therefore he is mortal

In this example, there is an implicit premise at play, namely that all humans are mortal. This is not given however. It is somehow presupposed as part of the general world knowledge. What would be interesting to see is to check whether such implicit arguments can be retrieved via the richness of a network like JDM. Indeed, this can be done. In particular, the entry for *mortal* in JDM, specifies *human* as one of its hyponyms. So, extracting lexical relations for *human* will also extract the synonym relation. Thus, it is easy to get the inference we are interested in. The same kind of information that can lead to retrieving the implicit argument in further examples like the ones shown below can be found using the richness of a network like JDM:[22]

(48) He coughs. He is ill.

[22] We are rather simplifying here, given that in JDM there is more than one relation between *human* and *mortal*. One finds the hyponym relation, the synonym relation as well as the characteristic relation (mortality as a characteristic

(49) She has a child, thus she has given birth.

Of course, it would be naive to think that enthymematic inference can be dealt with in full via using only information present in a lexical network, no matter how rich that network is. We are not suggesting anything like this. The interested reader that wants to have a deeper look at the issue of enthymemes and reasoning with them in a type-theoretic framework is directed to Breitholtz (2014). For the needs of this paper, it is enough to mention that at least some cases of enthymemes can be captured via a combination of lexical semantics information taken from a rich lexical network like JDM and their feeding to a richly typed logical system like the one we are endorsing in this paper.

6 CONCLUSION

In this paper we have looked at the way one can use information from a rich GWAP lexical network in order to construct typing ontologies for NL in rich type theories. Rich or modern type theories offer us elaborate typing structures and type many-sortedness, where the monolithic domain of individuals is substituted by a multitude of types. The problem that is created however, given this context, concerns which types need to be represented and which not, as well as the criteria that one uses in order to reach to such a decision. In this paper, we have proposed that one does not have to take such a decision but rather leave this information flow from a lexical network, in our case a rich GWAP network, JDM. We have proposed an initial way of doing this, namely extracting information from JDM w.r.t. to base types for common nouns as well as the types for other categories like verbs and adjectives. We have also proposed to use MTTs for several other types of information obtained from such a rich network. Lastly, we have initiated a discussion on how one can further use this

of humans). From a logical point of view, it cannot be the case that two terms are both synonyms and hyponyms. Furthermore, as one of the reviewer's notes, the synonym relation in JDM seems to be assymetrical, otherwise one would expect things like *pandas are human* to be inferred. This raises a more general issue, i.e. handling potentially contradictory information in the network. This is something that we will definitely explore in the future.

richness of information, especially common knowledge information, in order to deal with aspects of inference. On the one hand you have a wealth of lexical-semantic relations and on the other, a very rich and expressive compositional framework with powerful reasoning mechanisms. The result one would aim at, given this situation, is a combination of these two aspects in order to perform reasoning tasks with NLI. We have discussed some simple reasoning examples using the Coq proof-assistant, a proof-assistant that implements an MTT. Information is extracted from JDM and then translated into Coq code (thus into an MTT variant). The results are promising, showing a potential to deal with important aspects of NL reasoning. Furthermore, some easy cases of reasoning under implicit premises, i.e. enthymematic inference, were also shown to be captured via retrieving the implicit premises as lexical information associated with words appearing in the explicit premises. It is our hope that this work will initiate a more active discussion on the need for more fine grained frameworks for formal semantics as well as an active dialogue between people working on lexical networks and type theoretical (or logical in general) semantics from both a theoretical and an implementational point of view.

REFERENCES

Nicholas Asher (2008), A type driven theory of predication with complex types, *Fundamenta Informaticae*, 84(2):151–183.

Nicholas Asher (2012), *Lexical Meaning in Context: a Web of Words*, Cambridge University Press.

Christoph Benzmüller, Frank Theiss, and Arnaud Fietzke (2007), The LEO-II Project, in *Automated Reasoning Workshop*.

Jean-Philippe Bernardy and Stergios Chatzikyriakidis (2017), A Type-Theoretical system for the FraCaS test suite: Grammatical Framework meets Coq, ms, University of Gothenburg. `http://www.stergioschatzikyriakidis.com/uploads/1/0/3/6/10363759/iwcs_bercha.pdf`.

Ellen Breitholtz (2014), *Enthymemes in Dialogue: A micro-rhetorical approach*, Ph.D. thesis, University of Gothenburg.

Lucas Champollion and Manfred Krifka (2016), Mereology, in Paul Dekker and Maria Aloni, editors, *Cambridge Handbook of Semantics*, pp. 369–388, Cambridge University Press.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2012), An Account of Natural Language Coordination in Type Theory with Coercive Subtyping, in Y. PARMENTIER and D. DUCHIER, editors, *proceedings of Constraint Solving and Language Processing (CSLP12). LNCS 8114*, pp. 31–51, Orleans.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2013), Adjectives in a modern type-theoretical setting, in G. MORRILL and J.M NEDERHOF, editors, *Proceedings of Formal Grammar 2013. LNCS 8036*, pp. 159–174.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2014a), Natural Language Inference in Coq, *Journal of Logic, Language and Information.*, 23(4):441–480.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2014b), Natural Language Reasoning Using proof-assistant technology: Rich Typing and beyond, in *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, pp. 37–45.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2014c), Using Signatures in Type Theory to Represent Situations, *Logic and Engineering of Natural Language Semantics 11. Tokyo*.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2017a), Adjectival and Adverbial Modification: The View from Modern Type Theories, *Journal of Logic, Language and Information*, 26(1):45–88.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2017b), *On the Interpretation of Common Nouns: Types Versus Predicates*, pp. 43–70, Springer International Publishing.

Alonzo CHURCH (1940), A Formulation of the Simple Theory of Types, *J. Symbolic Logic*, 5(1).

Allan M COLLINS and M Ross QUILLIAN (1969), Retrieval time from semantic memory, *Journal of verbal learning and verbal behavior*, 8(2):240–247.

Robin COOPER, Simon DOBNIK, Shalom LAPPIN, and Staffan LARSSON (2014), A probabilistic rich type theory for semantic interpretation, in *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, pp. 72–79.

Coq 2007 (2007), *The Coq Proof Assistant Reference Manual (Version 8.1), INRIA*, The Coq Development Team.

Christiane FELLBAUM (1998), *WordNet: An Electronic Lexical Database*, MIT press.

Bruno GAUME, Karine DUVIGNAU, and Martine VANHOVE (2007), Semantic associations and confluences in paradigmatic networks, in Martine VANHOVE, editor, *Typologie des rapprochements sémantiques*, p. (on line), John Benjamins Publishing Company.

Peter GEACH (1962), *Reference and Generality: An examination of some Medieval and Modern Theories*, Cornell University Press.

Jean-Yves GIRARD (1971), Une extension de l'interpretation fonctionelle de Gödel à l'analyse et son application à l'élimination des coupures dans et la thèorie des types, *in proceedings of the 2nd Scandinavian Logic Symposium*. North-Holland, Amsterdam, pp. 63–92.

Mathieu LAFOURCADE (2007a), Making people play for Lexical Acquisition., in *SNLP 2007, 7th Symposium on Natural Language Processing. Pattaya, Thailande, 13-15 December 2007*.

Mathieu LAFOURCADE (2007b), Making people play for Lexical Acquisition with the JeuxDeMots prototype, in *SNLP'07: 7th international symposium on natural language processing*, p. 7.

Mathieu LAFOURCADE, Alain JOUBERT, and Nathalie. LE BRUN (2015), *Games with a Purpose (GWAPS)*, Focus Series in Cognitive Science and Knowledge Management, Wiley, ISBN 9781848218031.

Henry LIEBERMAN, Dustin SMITH, and Alea TEETERS (2007), Common Consensus: a web-based game for collecting commonsense goals., in *Workshop on Common Sense for Intelligent Interfaces, ACM Conferences for Intelligent User Interfaces (IUI 2007), Honolulu*.

Zhaohui LUO (1994), *Computation and Reasoning: A Type Theory for Computer Science*, Oxford University Press.

Zhaohui LUO (1999), Coercive subtyping, *Journal of Logic and Computation*, 9(1):105–130.

Zhaohui LUO (2010), Type-Theoretical Semantics with Coercive Subtyping, *Semantics and Linguistic Theory 20 (SALT20), Vancouver*.

Zhaohui LUO (2011), Contextual analysis of word meanings in type-theoretical semantics, *Logical Aspects of Computational Linguistics (LACL'2011). LNAI 6736*.

Zhaohui LUO (2012), Common Nouns as Types, in *LACL'2012, LNCS 7351*.

Zhaohui LUO, Sergei SOLOVIEV, and Tao XUE (2012), Coercive subtyping: theory and implementation, *Information and Computation*, 223:18–42.

Andrea MARCHETTI, Maurizio TESCONI, Francesco RONZANO, Marco ROSELLA, and Salvatore MINUTOLI (2007), SemKey: A Semantic Collaborative Tagging System, in *Tagging and Metadata for Social Information Organization Workshop, WWW07*.

Per MARTIN-LÖF (1975), An Intuitionistic Theory of Types: predicative part, in H.ROSE and J.C.SHEPHERDSON, editors, *Logic Colloquium'73*.

Per MARTIN-LÖF (1984), *Intuitionistic Type Theory*, Bibliopolis.

Igor MEL'CUK and Andrei ZHOLKOVSKY (1988), The Explanatory Combinatorial Dictionary, in Martha Walton EVENS, editor, *Relational Models of the Lexicon: Representing Knowledge in Semantic Networks*, pp. 41–74, Cambridge University Press, Cambridge.

Rada MIHALCEA and Timothy CHKLOVSKI (2003), Building sense tagged corpora with volunteer contributions over the Web, in *RANLP*, volume 260 of *Current Issues in Linguistic Theory (CILT)*, pp. 357–366, John Benjamins, Amsterdam/Philadelphia.

George A. MILLER (1995), WordNet: A Lexical Database for English, *Commun. ACM*, 38(11):39–41.

Koji MINESHIMA, Yusuke MIYAO, and Daisuke BEKKI (), Higher-order logical inference with compositional semantics, in *Proceedings of EMNLP15*, pp. 2055–2061.

Richard MONTAGUE (1974), *Formal Philosophy*, Yale University Press, collected papers edited by R. Thomason.

Alain POLGUÈRE (2003), Collocations et fonctions lexicales : pour un modèle d'apprentissage., *Revue Française de Linguistique Appliquée*, E(1):117—133.

Alain POLGUÈRE (2014), From Writing Dictionaries to Weaving Lexical Networks, *International Journal of Lexicography*, 27(4):396—418.

James PUSTEJOVSKY (1995), *The Generative Lexicon*, MIT.

Aarne RANTA (1994), *Type-Theoretical Grammar*, Oxford University Press.

Christian RETORÉ (2014), The Montagovian Generative Lexicon Lambda Ty: a Type Theoretical Framework for Natural Language Semantics, in Ralph MATTHES and Aleksy SCHUBERT, editors, *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 202–229, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, ISBN 978-3-939897-72-9, ISSN 1868-8969, doi:http://dx.doi.org/10.4230/LIPIcs.TYPES.2013.202, `http://drops.dagstuhl.de/opus/volltexte/2014/4633`.

Franck SAJOUS, Emmanuel NAVARRO, Bruno GAUME, Laurent PRÉVOT, and Yannick CHUDY (2013), Semi-automatic enrichment of crowdsourced synonymy networks: the WISIGOTH system applied to Wiktionary, *Language Resources and Evaluation*, 47(1):63–96.

Katharina SIORPAES and Martin HEPP (2008), Games with a Purpose for the Semantic Web, 23:50–60, ISSN 1541-1672, doi:10.1109/MIS.2008.45, `http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4525142`.

John SOWA and John ZACHMAN (1992), Extending and Formalizing the Framework for Information Systems Architecture, *IBM Systems Journal*, 31(3):590–616.

Luis VON AHN and Laura DABBISH (2004), Labeling Images with a Computer Game, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pp. 319–326, ACM, New York, NY, USA.

Luis VON AHN and Laura DABBISH (2008), Designing games with a purpose, *Commun. ACM*, 51(8):58–67.

Luis VON AHN, Mihir KEDIA, and Manuel BLUM (2006), Verbosity: a game for collecting common-sense facts, in *CHI*, pp. 75–78, ACM.

Manel ZARROUK (2015), *Endogeneous Consolidation of Lexical Semantic Networks*, Theses, Université de Montpellier.

Naomi ZEICHNER, Jonathan BERANT, and Ido DAGAN (2012), Crowdsourcing inference-rule evaluation, in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pp. 156–160, Association for Computational Linguistics.

Michael ZOCK and Slaven BILAC (2004), Word lookup on the basis of associations: from an idea to a roadmap., in *Proceedings of the Workshop on Enhancing and Using Electronic Dictionaries, Association for Computational Linguistics.*, pp. 29–35.

Michael ZOCK and Didier SCHWAB (2008), Lexical Access Based on Underspecified Input, in *Proceedings of the Workshop on Cognitive Aspects of the Lexicon*, COGALEX '08, pp. 9–17, Association for Computational Linguistics, Stroudsburg, PA, USA.