Type Theories and Lexical Networks: Using Serious Games as the Basis for Multi-Sorted Typed Systems.

Stergios Chatzikyriakidis, Mathieu Lafourcade, Lionel Ramadier and Manel Zarrouk[⋆]

LIRMM
University of Montpellier 2
stergios.chatzikyriakidis@lirmm.fr
mathieu.lafourcade@lirmm.fr
lionel.ramadier@lirmm.fr
manel.zarrouk@lirmm.fr

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 as well as different sorts of information in doing formal semantics using modern type theories (type theories within the tradition of Martin Löf). We discuss the domain of base types, adjectival and verbal types, hyperonymy/hyponymy relations as well as more advanced issues like homophony and polysemy.

1 Introduction

Modern Type Theories, i.e. Type Theories within the tradition of Martin Löf [13,14], have become a major alternative to Montague Semantics (MS) in the last twenty years. A number of influential approaches using MTTs have been proposed by the years [15,11,16,7] and have shown that the rich typing system offered by these MTTs (type many-sortedness, dependent types, type universes) has considerable advantages over simple typed theories predominantly used in mainstream formal semantics. One 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 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 [5,4]. However, this expresiveness 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

⋆ This work is partially supported by the ANR ContInt Polymnie project in France.

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

entities found in MS, but on the other hand, constructing such a type ontology is not at all straightforward and easy task. Different approaches and assumptions have been put forward. For example [11,12] proposed to treat CNs as types, in effect every CN is a different type. Approaches like [16] on the other hand, take a more moderate view by building type ontologies according to classifier systems (in effect type ontology is built using intuitions from classifier systems).

On the other hand, work with lexical-semantic networks have provided us with structured lexicons that indicate lexical and semantic relations like for example WordNet [8]. A very promising line of research in lexico-semantic network construction concerns networks which are built collaboratively by using GWAPs, games with a purpose i.e. serious games. This is the case of JeuxDeMots Lexical Network (JDM, [9]), which is constructed through many GWAPs along with a contributive tool (Diko) which allows players/users to contribute directly and to browse the knowledge base.

In this paper, we propose to ground our semantic ontologies as well as any other information needed in order to perform reasoning tasks using MTT semantics in the JDM lexical network. In order to do this, we present some first thoughts on how such an endeavour can be made 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.

2 From JDM to MTTs

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 [11,12], every CN is associated with a base type. Note the first major difference with MS: CNs are not predicates but rather Types.[2] For base types, an easy way to unify the two approaches is to assume that CNs are basically base types. In JDM, POS tagging will provide this sort of information. We further have to exclude instances of terms (for example $John$ as an instance of Man) in order to distinguish between instances of terms and terms themselves (CNs).[3] This can be by excluding named entities:

(1) $\forall A.POS(N,A) \wedge \neg SEM\_LABEL(NE,A) \Rightarrow A{:}\textsc{cn}.$

[2] See [12] for a number of arguments as regards the advantages of this move.

[3] This does not mean that we are not interested in instances. To 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 which we do not show here. The type CNis technically a universe, a collection of the names of types into a single type. See [11,3] for more information on the use of the type theoretic notion of universe as this is employed in MTT semantics.

Hyponym and hypernym relations are then defined as subtypes:

(2) $\forall A, B.Hyp(A, B) \Rightarrow A < B{:}\textsc{cn}$.

(3) $\forall A, B.HyR(A, B) \Rightarrow B < A{:}\textsc{cn}$.

Synonyms can be defined using equality:[4]

(4) $\forall A, B.Syn(A, B) \Rightarrow A = B{:}\textsc{cn}$.

Synonymicity is not only relevant for CNs but for other linguistic categories. We can do that as well by changing to:

(5) $\forall A, B.Syn(A, B) \Rightarrow A = B{:}C(C{:}LType)$

LType is a universe of linguistic types, it includes the types instantiated in linguistic semantics (CN, adjectival and verbal types, types for quantifiers etc. See [2] for a discussion).

For instances of terms, like for example proper names, we define the following:

(6) $\forall A, B.Ins(A, B) \Rightarrow A{:}B$

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 $Einstein{:}Person$ with $Person{:}\textsc{cn}$.

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 in general thematic relations are defined. These can guide us in assigning types for predicates. In JDM, one can look for semantic relations like $action > agent$, $action > patient$ and various other such relations. For example *man* appears as the agent of a number of verbs that express actions like *question*. There is a further relation, the inverse agent 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, *question* will involve among others *teacher, mother, child, daugther, person, human.* How can we make sense in order to provide typings in MTTs? Well, there is a straightforward to do this. What we need is to find the most general term, i.e. the term that all the other terms are hyponyms of. Instances of terms are not needed in this process.[5]

[4] Of course, this will treat A and B as perfect synonyms which as we know do not really exist in natural languages. We do not discuss this issue here.

[5] The formula reads as follows: forall 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

(7) $\forall A, B.Agent(A, B).\exists C.(Hyp(A, C) \vee (A = C)) \Rightarrow C \rightarrow Prop$

Similar processes can be defined for the patient arguments as well as for adjectives. Adjectives like *grande* can be defined as regards typing by looking at the terms that can have this characteristic. Again, JDM, gives us a list of terms and instances of terms for this reason. For adjectives, we might (depending) on the adjective, give either a typing in the same style as we have proposed above or a polymorphic type extending over a universe which includes the most general type found along with its subtypes:

(8) $\forall A, B.Agent(A, B).\exists C.Hyp(A, C) \Rightarrow C \rightarrow Prop$

(9) $\forall A, B.Char(A, B).\exists C.Hyp(A, C) \Rightarrow \Pi C{:}\mathrm{CN}_C.C \rightarrow Prop$

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 $part\_of\colon [\![Object]\!] \rightarrow [\![Object]\!] \rightarrow Prop$.

2.3 Polysemy

The next issue we want to look at is polysemy, most specifically what to do with respect to 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 they 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 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 a word with more than one meaning (irrespective of whether the meanings are related or not) are dubbed as polysemous, and then additional levels of refinement where relations between the different meanings can arise (or not). 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 [11], homophony is treated via using 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 (see [10] for the formal details). It is a difficult task to be able to translate from a polysemous term identified in JDM

or are equal to C, the predicate $C \rightarrow Prop$ is returned. In case there is no supertype in the refinements, then there are two 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, $[\![Human]\!] \rightarrow Prop$ and $[\![Vehicle]\!] \rightarrow Prop$.

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. JDM distinguishes different meanings of *individual* with respect to its domain of appearance, i.e. the different notion of individual in statistics, biology or administration. This level of fine-grainedness is not found in formal semantics. However, there is no reason why we should not go into this level of detail in MTTs. This can be captured in a translation procedure by presenting the different types according to the different domains. In order to encode domains, we use type theoretic contexts [15,6]. The following translation can be defined for these cases:

(10) $\forall A, B_1, ..., B_n.POS(N, A) \wedge \neg(Instance(PN, A)) \wedge Domain(B_1, ..., B_n) \wedge A(\text{in } B_1, ..., B_n) \Rightarrow A{:}\text{CN in } \mathit{\Gamma}_B 1, A{:}\text{CN in } \mathit{\Gamma}_{B...}, A{:}\text{CN in } \mathit{\Gamma}_B n$

What about other cases of polysemy like for example *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:

(11) $\forall A, B_1, ..., B_n.POS(N, A) \wedge \neg(POS(PN, A)) \wedge Ref(A, (B_1, ..., B_n)) \Rightarrow A < B_1, ..., B_n{:}\text{CN}$

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: 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:

(12) $\forall A, B_1...B_n.POS(N, A) \wedge Ref(A(B_1, ..., B_n))A, B \in DR \Rightarrow A{:}CN < B_1 \bullet B... \bullet B_n$

More information on ways to approach the translation procedure with respect to polysemy will be given in the full paper. There, other cases will be discussed: a) cases where the two meanings are associated with different types (e.g. a case where one meaning is verbal and the other adjectival), b) Cases where a combination of approaches might be needed (e.g. a polysemous noun that further has adjectival or verbal meanings). Further issues of how to use world-knowledge information drawn from JDM and in which way will also be discussed. Last but not least, we are going to discuss how can such an idea be implemented in a system that will take as input information from JDM and will output MTT representations (to be used for example in the proof-assistant Coq).

References

1. C. Benzmüller, Theiss F., and Fietzke A. The LEO-II project. In *Automated Reasoning Workshop*, 2007.
2. S. Chatzikyriakidis and Z. Luo. An account of natural language coordination in type theory with coercive subtyping. In Y. Parmentier and D. Duchier, editors, *Proc. of Constraint Solving and Language Processing (CSLP12). LNCS 8114*, pages 31–51, Orleans, 2012.
3. S. Chatzikyriakidis and Z. Luo. Adjectives in a modern type-theoretical setting. In G. Morrill and J.M Nederhof, editors, *Proceedings of Formal Grammar 2013. LNCS 8036*, pages 159–174, 2013.
4. S. Chatzikyriakidis and Z. Luo. Natural language inference in Coq. *J. of Logic, Language and Information.*, 23(4):441–480, 2014.
5. S. Chatzikyriakidis and Z. Luo. Natural language reasoning using proof-assistant technology: Rich typing and beyond. In *Proceedings of EACL2014*, 2014.
6. S. Chatzikyriakidis and Z. Luo. Using signatures in type theory to represent situations. *Logic and Engineering of Natural Language Semantics 11. Tokyo*, 2014.
7. Robin Cooper, Simon Dobnik, Shalom Lappin, and Staffan Larsson. A probabilistic rich type theory for semantic interpretation. In *Proceedings of the European Associaton of Computational Linguistics*, 2014.
8. C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT press, 1998.
9. M. Lafourcade. Making people play for lexical acquisition with the jeuxdemots prototype. In *SNLP'07: 7th international symposium on natural language processing*, page 7, 2007.
10. Z. Luo. Type-theoretical semantics with coercive subtyping. *Semantics and Linguistic Theory 20 (SALT20), Vancouver*, 2010.
11. Z. Luo. Contextual analysis of word meanings in type-theoretical semantics. *Logical Aspects of Computational Linguistics (LACL'2011). LNAI 6736*, 2011.
12. Z. Luo. Common nouns as types. In D. Bechet and A. Dikovsky, editors, *Logical Aspects of Computational Linguistics (LACL'2012). LNCS 7351*, 2012.
13. P. Martin-Löf. An intuitionistic theory of types: predicative part. In H.Rose and J.C.Shepherdson, editors, *Logic Colloquium'73*, 1975.
14. P. Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, 1984.
15. A. Ranta. *Type-Theoretical Grammar.* Oxford University Press, 1994.
16. C. Retoré. The montagovian generative lexicon Tyn: a type theoretical framework for natural language semantics. In R. Matthes and A. Schubert, editors, *Proc of TYPES2013*, 2013.