Natural Language Reasoning using Coq: Interaction and Automation

Stergios Chatzikyriakidis
LIRMM, University of Montpellier 2
stergios.chatzikyriakidis@lirmm.fr

Résumé. Dans cet article, nous présentons une utilisation des assistants des preuves pour traiter l'inférence en Language Naturel (NLI). D' abord, nous proposons d'utiliser les theories des types modernes comme langue dans laquelle traduire la sémantique du langage naturel. Ensuite, nous implémentons cette sémantique dans l'assistant de preuve Coq pour raisonner sur ceux-ci. En particulier, nous évaluons notre proposition sur un sous-ensemble de la suite de tests FraCas, et nous montrons que 95.2% des exemples peuvent être correctement prédits. Nous discutons ensuite la question de l'automatisation et il est démontré que le langage de tactiques de Coq permet de construire des tactiques qui peuvent automatiser entièrement les preuves, au moins pour les cas qui nous intéressent.

Abstract. In this paper, we present the use of proof-assistant technology in order to deal with Natural Language Inference. We first propose the use of modern type theories as the language in which we translate natural language semantics to. Then, we implement these semantics in the proof-assistant Coq in order to reason about them. In particular we evaluate against a subset of the FraCas test suite and show a 95.2% accuracy and also precision levels that outperform existing approaches at least for the comparable parts. We then discuss the issue of automation, showing that Coq's tactical language allows one to build tactics that can fully automate proofs, at least for the cases we have looked at.

Mots-clés : Inference en Langage Naturel, Théorie des Types, Sémantique Formelle, FraCas, Coq, Automatisation des Preuves.

Keywords: Natural Language Inference, Type Theory, Formal Semantics, FraCas, Coq, Proof automation.

1 Introduction

1.1 Natural Language Inference

Central within a theory of formal semantics for Natural Language (NL) is the study of Natural Language Inference (NLI). Roughly put, NLI is the task of determining whether an NL hypothesis can be inferred from an NL premise. Human beings do not only have the ability to understand infinite many NL sentences but can further reason about these. In effect, understanding a NL sentence amounts (among others) to knowing what can be inferred or not from such a sentence.

Natural Language Inference has been also central in the field of computational semantics. As Cooper et al. aptly put it 'inferential ability is not only a central manifestation of semantic competence but is in fact centrally constitutive of it' (Cooper & Ginzburg, 1996). Inferential ability according to Cooper et al. (Cooper & Ginzburg, 1996) is the best way to test the semantic adequacy of NLP systems. [1]

A number of NLI platforms have been proposed over the years in order to evaluate NLI systems. The two most important ones are : a) the FraCas test suite, and b) the Recognizing Textual Entailment (RTE) challenges. For the needs of this paper, we concentrate on the FraCas test suite (Cooper & Ginzburg, 1996)

1. At this point, it is necessary to distinguish between deep and shallow approaches to inference. In a nutchell, shallow approaches refers to approaches where no translation to an intermediate language is done (Romano *et al.*, 2006; Glickmann *et al.*, 2005; Hickl *et al.*, 2005; MacCartney *et al.*, 2008) among many others, where deep approaches concern approaches that perform a translation to a logical language prior to inference (Bos & Markert, 2005; Pulman, 2013). There are also hybrid approaches like (MacCartney, 2009). Obviously, the approach pursued here is a deep approach.

1.2 The FraCas test suite

The FraCas Test Suite (Cooper & Ginzburg, 1996) arose out of the FraCas Consortium, a huge collaboration with the aim to develop a range of resources related to computational semantics. The FraCas test suite is specifically designed to reflect what an adequate theory of NLI should be able to capture. It comprises NLI examples formulated in the form of a premise (or premises) followed by a question and an answer. It involves 345 examples classified according to the semantic phenomenon involved, e.g. quantifiers, adjectives, tense etc. Here are some illustrative examples from the suite :

(1) Some irish delegates finished the survey on time.
Did any delegate finish the survey on time ? [Yes, FraCas 3.55]

(2) All mice are small animals.
Mickey is a large mouse.
Is Mickey a large animal ? [No, FraCas 3.210]

In this paper, we discuss inference against a subset of the FraCas test suite, approximately 1/4 of the suite.

1.3 Modern Type Theories

The term Modern Type Theories (MTTs), refers to type theories within the tradition of Martin Löf (Martin-Löf, 1971; Martin-Löf, 1984).[2] In linguistics, this tradition has been initiated with the pioneering work of Ranta (Ranta, 1994).[3]. In this paper, we use one such TT, specifically Luo's Type Theory with Coercive Subtyping (Luo, 2011, 2012a) among many others. One of the advantages of MTTs compared to traditional Montagovian approaches is that MTTs can be seen as being both model-theoretic and proof-theoretic. NL semantics can first be represented in an MTT in a model-theoretic way and then these semantic representations can be understood inferentially in a proof-theoretic way. (Luo, 2014). Besides this advantage, MTTs offer a number of features that allow for semantic fine-grainedness and expresiveness. Some of the most important ones are briefly mentioned below :

1. Type structures in MTTs are very rich. Types in MTTs can be used to represent collections of objects (or constructive sets, informally) in a model-theoretic sense, although they are syntactic entities in MTTs.
2. The notion of signature in an MTT, as introduced in (Luo, 2014; Chatzikyriakidis & Luo, 2014b), can be used to represent situations or (incomplete) possible worlds.

As regards the second point, see (Luo, 2014) for more examples. Now, elaborating on the expressiveness of typing structures of MTTs, we briefly mention the following type structures :

– Dependent sum types (Σ-types $\Sigma(A, B)$ which have product types $A \times B$ as a special case). Σ-types have been used to interpret intersective and subsective adjectives without the need of resorting to meaning postulates. The inferences follow directly from typing (Ranta, 1994; Chatzikyriakidis & Luo, 2013). Note that subtyping is essential for the Σ-type to work (Luo, 2012b).
– Dependent product types (Π-types $\Pi(A, B)$, which have arrow-types $A \rightarrow B$ as a special case). These are basic dependent types that, together with universes (see below), provide polymorphism among other things. To give an example, verb modifying adverbs are typed by means of dependent Π-types (together with the universe CN of common nouns) (Luo, 2012b; Chatzikyriakidis, 2014).
– Disjoint union types ($A + B$). Disjoint union types have been proposed to give interpretations of privative adjectives (Chatzikyriakidis & Luo, 2013).
– Universes. These are types of types, basically collections of types. Typical examples of universes in MTT-semantics include, among others, the universe $Prop$ of logical propositions as found in impredicative type theories and the universe CN of (the interpretations of) common nouns (Luo, 2012b) Further uses of universes can be seen in (Chatzikyriakidis & Luo, 2012) where the universe $LType$ of all linguistic types is used in order to deal with coordination.

2. Note that this is a term introduced by (Luo, 2011, 2012a) in order to distinguish type theories with the tradition of Martin-Löfand simple type theories as these are generally used within the Montagovian tradition. A similar term, 'rich' type theories has been used by other people like (Cooper *et al.*, 2014).

3. Potentially, even further back, with the work of Sundholm (Sundholm, 1986, 1989)

– Dot-types ($A \bullet B$). These are special types introduced to study co-predication (Luo, 2012b). It is worth mentioning that coercive subtyping is essentially employed in the formulation of dot-types. [4]

Besides the above, we should also emphasise that *subtyping* is crucial for an MTT to be a viable language for formal semantics. Furthermore, also very importantly, subtyping is needed when considering many linguistic features such as copredication (Asher, 2012; Pustejovsky, 1995).

2 Using Coq as a Natural Language Reasoner

Given MTT's proof-theoretic aspect, it is not surprising that many proof assistants implement MTTs. Starting from the early AUTOMATH system and all the way to the state of the art proof-assistants like Coq (Coq, 2004) or Agda (Agda 2008, 2008), MTTs have been shown to be a good language for interactive theorem proving. Perhaps, the most advanced of these provers is the Coq proof-assistant (Coq, 2004), a powerful proof assistant that has been used successfully to derive a number of impressive results. Some of these include a complete mechanized proof of the four colour theorem (Gonthier, 2005), the odd order theorem (Gonthier *et al.*, 2013) as well as CompCert , a formally verified compiler for C (Leroy, 2013).

Now, given : a) Coq's powerful reasoning ability and b) that it implements a MTT, a new avenue of research is opened up. To use Coq as a NL reasoner. This has been attempted in (Chatzikyriakidis & Luo, 2014a) with a number of promising results. In this paper, I present an extension of this approach that improves on accuracy and precision over previous deep approaches to NLI. We then discuss, how we can use proof automation in order to fully automate NL reasoning.

2.1 How the system works

As already said, Coq implements a MTT, more specifically the Calculus of Inductive Constructions, a type theory which is very close to the MTT we are using for representing NL semantics. It is thus straightforward to provide MTT NL semantics in Coq since Coq 'speaks' so to say the language already. Now, given that Coq is a powerful theorem prover, it can further reason about the implemented semantics. In fact, we can use Coq's proof mechanisms to prove valid NL inferences, in the same sense we use Coq to prove valid mathematical or logical theorems. We now move to exemplify how this idea can be evaluated against the FraCas test suite (Cooper *et al.*, 1996). The FraCas test suite involves three categories of NLIs : positive (YES), negative (NO) and unknown (UNK). For positive NLIs, we construct the example as a declarative hypothesis in the form of a conditional and try to prove it as a theorem. A correct account should be able to prove all YES NLIs as theorems. For negative NLIs, we formulate the example but instead we try to prove the negation of the consequent. For UNK NLIs, we should not be able to find a proof for either case of the consequent (both positive and negative). We use the modified Grammatical Framework parser (GF, (Ljunglof & Siverbo, 2011)) in order to parse the FraCas examples and then we translate the examples to the syntax of Coq. [5] But let us see how this works by looking at example (1) repeated below as (3) :

(3) Some irish delegates finished the survey on time.
Did any delegate finish the survey on time ? [Yes]

We assume a Σ type approach to modification, where the first projection is a coercion ($\Sigma(delegate, Irish) < delegate$ with $delegate$:CN). In Coq this is done by using dependent record types :

(4) Record Irishdelegate : CN := mkIrishdelegate [d : > delegate ; _ : Irish d]

Adverbs and quantifiers are given the following types respectively :

(5) $\Pi A{:}\ \text{CN}.\ (A \to Prop) \to (A \to Prop)$

4. See (Bassac *et al.*, 2010) for another proposal of using coercions to deal with co-predication. See also (Chatzikyriakidis & Luo, 2015) for further elaboration on the existing dot-type account.

5. Note that for the moment, we do not have an automatic translation procedure between the two. This is something that we are currently working on.Given GF's ability to translate accurately between languages (natural or formal), this task seems feasible.

(6) $\Pi A\colon \text{CN}.\ (A \to Prop) \to Prop$

These assumptions put together are enough to prove the example. The start of the proof is shown below :

```
Theorem IRISH:(some Irishdelegate(on_time(finish(the survey)))->
(some delegate)(on_time (finish(the survey))).
```

The command $theorem$ puts Coq into proof mode. We unfold the definitions using $cbv\ delta$ which replaces the occurrences of a defined notion by the definition itself in the current goal (or in any of the hypotheses). We then apply *intro*, which introduces the antecedent as an assumption :

```
H:exists x:Irishdelegate,(let (a, _) := ADV (finish (the survey)) in a)
 ============================
exists x : delegate, (let (a, _):=ADV (finish (the survey))
```

Then we apply $induction$ which performs $elim\ H$ (it applies the correct destructor to an inductive type, in our case to the hypothesis H) and $intro$:

```
x:Irishdelegate
H:(let (a, _):= ADV (finish(the survey))in a) (let (c, _):=x in c)
============================
exists x0:delegate,(let (a, _):=ADV(finish(the survey))in a)x0
```

At this point we can substitute $x0$ with x. The treatment of adjectival modification allows this substitution, and thus a proof can be found :

```
IRISH2 < exists x.
1 subgoal
x : Irishdelegate
H : (let (a, _):= ADV(finish (thesurvey))in a)(let (c, _)
============================
(let (a, _):=ADV(finish(the survey))in a)x
IRISH2 < assumption.
Proof completed.
```

We have tested against almost 30% of the FraCas test suite (a total of 102 plus 3 examples outside the suite). The evaluation involved examples from the following sections :

– Quantifiers and monotonicity (41 examples).
– Conjoined noun phrases (15 examples).
– Adjectives (18 examples).
– Dependent plurals (2 examples)
– Comparatives (10).
– Epistemic, intensional and reportive attitudes (11 examples).
– Collective predication (6 examples).
– Quantificational adverbs (2 examples)

100/105 examples were correctly captured, giving an overall accuracy of 95.2%. The state of the art in precision on the FraCas test suite is Maccartrney (MacCartney, 2009) with an overall accuracy of 70.5%, evaluated however against 53.3% of the suite (183 examples) and against semantic phenomena that we have not tested against (e.g. ellipsis). In order to look at more direct comparisons, MacCartney offers an accuracy of 97.7% in the quantifier section, evaluating against 44 examples, while we offer an accuracy of 100% on an evaluation against 41 examples. We offer an accuracy of 87.5% while (MacCartney, 2009) 80% for the same number of examples, while for plurals we offer an accuracy of 82.5% for 17 examples while (MacCartney, 2009) offers 75% for 25 examples. The system also achieves higher accuracy than the earlier similar system proposed by (Chatzikyriakidis & Luo, 2014a), raising accuracy from 93.5% to 95.2%. However, the

current system lacks an automatic translation between the parser and the syntax of Coq and this work is done manually in this respect, as we have already pointed out. [6] The nature of the automatic translation can significantly reduce these results if it is not efficient (basically because of low recall problems). The system as it stands can however guarantee great precision. To put things in perspective, it offers a precision of 100% for the YES section in the quantifier section while (MacCartney, 2009) offers 95.5% precision, while for the adjectival case it offers a precision of 81.8%, compared to 71.4% of (MacCartney, 2009). The following table summarizes the results from four sections of the FraCas test suite and a comparison between the approach presented here, (MacCartney, 2009) as well as (Angeli & Manning, 2014) is shown :

	Category	Count	Precision			Recall			Accuracy		
			AM	MC	N	AM	MC	N	AM	MC	N
1	Quantifiers	AM-MC :44 N :41	91	95	100	100	100	100	95	97	100
2	Plurals	AM-MC :25 N :17	80	90	100	29	64	69.2	38	75	82.5
3	Adjectives	AM-MC :15 N :15	80	71	81.8	66	83	100	73	80	87.5
4	Attitudes	AM-MC :9 N :11	-	100	100	0	83	100	22	89	100

We conclude that the approach as presented here offers a number of welcome results compared to previous approaches, most importantly a very high precision and accuracy. It is however worth pointing out that in order to have a more realistic evaluation of the system, this should be developed into an automatic system, where a) translation into Coq is not done manually but automatically and b) the system is further evaluated against a bigger fragment of the FraCas test suite as well as against inferences using natural text like the RTE challenges.

2.2 Automation

Given that Coq is an interactive rather than an automated theorem prover, in order to prove a given theorem the user has to guide the prover to the proof. The way to do this is via Coq's predefined tactics or any other tactic libraries that have been defined for different purposes. Coq itself involves a number of tactics that are designed in order to automate part of proofs. For example, the tactic $intuition$ solves all first-order intuitionistic tautologies. For a number of examples and once the definitions have been unfolded, these can be solved with $intuition$ only. Given that Coq allows the construction of new tactics, one can define a new tactic that will just unfold definitions followed by the application of $intuition$. This tactic, can then fully automate a number of the examples we have dealt in this paper. Another variant of this auto tactic, much more effective can be achieved by further adding $jauto$, a tactic like Coq's pre-defined $auto$ tactic but much more powerful since it can break up existentials as well, and congruence which can reason with equalities and inequalities. This new tactic, let us call it $AUTO$ is shown below :

(7) Ltac AUTO :=cbv delta ;intuition ;try repeat congruence ;jauto ;intuition.

Most of the examples we have looked at, can be solved with this tactic (for example 3 is such a case). Other examples need more powerful tactics. For example, comparatives and collective predication examples need more powerful tactics to be solved. However, these tactics can be provided and then they can be gathered into one general auto tactic that can effectively solve all the examples. This is indeed what we have done, we defined 3 different auto tactics and then combined these three tactics (let us call them a, b and c) into a general auto tactic (d) :

(8) Ltac d := solve[a|b|c].

With this tactic full automation can be achieved. The tactic succeeds if one of the tactics can solve the theorem. Otherwise, it fails.

3 Conclusions

In this paper, we have presented the use of Coq as an NL reasoner. Given Coq's 'understanding' of MTT semantics, we straightforwardly implemented MTT semantics for NL. We evaluate the approach against almost 30% of the FraCas test

6. The details of such a translation are currently under development and we hope that an efficient translation that will maintain the impressive results as regards accuracy will be available.

suite, where an accuracy of 95.2% was achieved and a better overall performance in the three comparable sections of the FraCas with earlier approaches was shown. We then discussed ways of using Coq's tactical language in order to fully automate the proof process. It was shown that at least for the examples we have looked at, this is feasible.

Références

Agda 2008 (2008). Agda proof assistant. `http://appserv.cs.chalmers.se/users/ulfn/wiki/agda.php`

ANGELI G. & MANNING C. D. (2014). Naturalli : Natural logic inference for common sense reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

ASHER N. (2012). *Lexical Meaning in Context : a Web of Words.* Cambridge University Press.

BASSAC C., MERY B. & RETORÉ C. (2010). Towards a type-theoretical account of lexical semantics. *Journal of Logic, Language and Information*, **19**(2).

BOS J. & MARKERT K. (2005). Recognising textual entailment with logical inference. In *Proc. of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP),*, p. 98–103.

CHATZIKYRIAKIDIS S. (2014). Adverbs in a modern type theory. In N. ASHER & S. SOLOVIEV, Eds., *Proceedings of LACL2014. LNCS 8535*, p. 44–56.

CHATZIKYRIAKIDIS S. & LUO Z. (2012). An account of natural language coordination in type theory with coercive subtyping. In Y. PARMENTIER & D. DUCHIER, Eds., *Proc. of Constraint Solving and Language Processing (CSLP12). LNCS 8114*, p. 31–51, Orleans.

CHATZIKYRIAKIDIS S. & LUO Z. (2013). Adjectives in a modern type-theoretical setting. In G. MORRILL & J. NEDERHOF, Eds., *Proceedings of Formal Grammar 2013. LNCS 8036*, p. 159–174.

CHATZIKYRIAKIDIS S. & LUO Z. (2014a). Natural language inference in Coq. *J. of Logic, Language and Information.*, **23(4)**, 441–480.

CHATZIKYRIAKIDIS S. & LUO Z. (2014b). Using signatures in type theory to represent situations. *Logic and Engineering of Natural Language Semantics 11. Tokyo*.

CHATZIKYRIAKIDIS S. & LUO Z. (2015). Individuation criteria, dot-types and copredication : A view from modern type theories. In *Under Review*.

COOPER R., CROUCH D., VAN EIJCK J., FOX C., VAN GENABITH J., JASPARS J., KAMP H., MILWARD D., PINKAL M., POESIO M. & PULMAN S. (1996). Using the framework. *Technical Report LRE 62-051r*. http ://www.cogsci.ed.ac.uk/ fracas/.

COOPER R., DOBNIK S., LAPPIN S. & LARSSON S. (2014). A probabilistic rich type theory for semantic interpretation. In *Proceedings of the European Associaton of Computational Linguistics*.

COOPER R. & GINZBURG J. (1996). A compositional situation semantics for attitude reports. In J. SELIGNMANN & D. WESTERSTAHL, Eds., *Logic, language and computation,*. CSLI.

Coq (2004). *The Coq Proof Assistant Reference Manual (Version 8.0), INRIA.* The Coq Development Team.

GLICKMANN O., DAGAN I. & KOPPEL M. (2005). Web based probabilistic textual entailment. In *Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment.*

GONTHIER G. (2005). A computer checked proof of the four colour theorem.

GONTHIER G., ASPERTI A., AVIGAD J., BERTOT Y., COHEN C., GARILLOT F., LE ROUX S., MAHBOUBI A., O'CONNOR R., BIHA S. O. *et al.* (2013). A machine-checked proof of the odd order theorem. In *Interactive Theorem Proving*, p. 163–179. Springer.

HICKL A., WILLIAMS J., BENSLEY J., ROBERTS K., RINK B. & SHI Y. (2005). Recognizing textual entailment with lCC's groundhog system. In *Proc. of Second PASCAL Challenges Workshop on Recognizing Textual Entailment*, p. 80–85.

LEROY X. (2013). The compcert c verified compiler : Documentation and user's manual. http ://compcert.inria.fr/man/manual.pdf.

LJUNGLOF P. & SIVERBO M. (2011). *A bilingual treebank for the FraCaS test suite.* Clt project report, University of Gothenburg.

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

LUO Z. (2012a). Common nouns as types. In D. BECHET & A. DIKOVSKY, Eds., *Logical Aspects of Computational Linguistics (LACL'2012). LNCS 7351*.

LUO Z. (2012b). Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy*, **35**(6), 491–513.

LUO Z. (2014). Formal Semantics in Modern Type Theories : Is It Model-theoretic, Proof-theoretic, or Both ? *Invited talk at Logical Aspects of Computational Linguistics 2014 (LACL 2014), Toulouse. LNCS 8535*, p. 177–188.

MACCARTNEY B. (2009). *Natural Language Inference*. PhD thesis, Stanford Universisty.

MACCARTNEY B., GALLEY M. & C.D. MANNING I. (2008). A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP-08*, p. 802–811.

MARTIN-LÖF P. (1971). An intuitionistic theory of types. manuscript.

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

PULMAN S. (2013). Second order inference in NL semantics. Talk given at the KCL Language and Cognition seminar, London.

PUSTEJOVSKY J. (1995). *The Generative Lexicon*. MIT.

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

ROMANO L., KUYLEKOV M., SZPEKTOR I., DAGAN I. & LAVELLI A. (2006). Investigating a generic paraphrase-based approach for relation extraction. In *Proceedings of EACL 2006.*, p. 409–416.

SUNDHOLM G. (1986). Proof theory and meaning. In D. GABBAY & F. GUENTHNER, Eds., *Handbook of Philosophical Logic III : Alternatives to Classical Logic*, p. 471–506. Reidel.

SUNDHOLM G. (1989). Constructive generalized quantifiers. *Synthese*, **79**(1), 1–12.