Natural Language Inference in Coq

Stergios Chatzikyriakidis & Zhaohui Luo

Springer

Springer

Natural Language Inference in Coq

Stergios Chatzikyriakidis · Zhaohui Luo

Abstract In this paper we propose a way to deal with natural language inference (NLI) by implementing Modern Type Theoretical Semantics in the proof assistant Coq. The paper is a first attempt to deal with NLI and natural language reasoning in general by using the proof assistant technology. Valid NLIs are treated as theorems and as such the adequacy of our account is tested by trying to prove them. We use Luo's Modern Type Theory (MTT) with coercive subtyping as the formal language into which we translate natural language semantics, and we further implement these semantics in the Coq proof assistant. It is shown that the use of a MTT with an adequate subtyping mechanism can give us a number of promising results as regards NLI. Specifically, it is shown that a number of inference cases, i.e. quantifiers, adjectives, conjoined noun phrases and temporal reference among other things can be successfully dealt with. It is then shown, that even though Coq is an interactive and not an automated theorem prover, automation of all of the test examples is possible by introducing user-defined automated tactics. Lastly, the paper offers a number of innovative approaches to NL phenomena like adjectives, collective predication, comparatives and factive verbs among other things, contributing in this respect to the theoretical study of formal semantics using MTTs.

S. Chatzikyriakidis (✉) · Z. Luo
Department of Computer Science,
Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK
e-mail: stergios.chatzikyriakidis@cs.rhul.ac.uk

Z. Luo
e-mail: zhaohui.luo@hotmail.co.uk

S. Chatzikyriakidis
Open University of Cyprus, Nicosia, Cyprus

1 Introduction

Natural language inference (NLI), i.e. the task of determining whether an NL hypothesis can be inferred from an NL premise, has been an active research theme in computational semantics in which various approaches have been proposed [see, for example MacCartney (2009) and some of the references therein]. In this paper, we study NLI based on formal semantics in modern type theories with coercive subtyping (Luo 2012) and its implementation in the proof assistant Coq (The Coq Development Team 2007).

A Modern Type Theory (MTT) is a dependent type theory consisting of an internal logic, which follows the propositions-as-types principle. This latter feature along with the availability of powerful type structures make MTTs very useful for formal semantics. Research on MTTs has been extremely fruitful in analyzing NL semantics and a number of problematic phenomena in NL semantics have been managed to be tackled. Earlier work by Sundholm (1986) and Ranta (1994), among others, have managed to deal in a rather adequate way with a number of semantic phenomena like e.g quantifiers, anaphora and donkey sentences among other things. The second author of the current paper has further developed MTT-based semantics via employing the impredicative type theory UTT (Luo 1994) enriched by an adequate subtyping mechanism, i.e. coercive subtyping (Luo 1999; Luo et al. 2012). MTT semantics has now gradually become a serious alternative to Montague (1974) semantics as regards formal semantics.

A proof assistant is a computer system that assists the users to develop proofs of mathematical theorems. A number of proof assistants implement MTTs. For instance, the proof assistant Coq (2007) implements pCIC, the predicative Calculus of Inductive Constructions,[1] and supports some very useful tactics that can be used to help the users to automate (parts of) their proofs. Proof assistants have been used in various applications in computer science (e.g., program verification) and formalised mathematics (e.g., formalisation of the proof of the 4-colour theorem in Coq).

The above two developments, the use of the MTT semantics on the one hand and the implementation of MTTs in proof assistants on the other, has opened a new research avenue: the use of existing proof assistants in dealing with NLI. In this paper, we present our work as regards NLI by implementing MTT semantics for NL in Coq. The purpose is to show how an interactive proof assistant such as Coq can help deal with NLI. In particular, we implement MTT semantics in Coq and then use Coq to reason about these semantics by dealing with various examples from the FraCas test suite (Cooper et al. 1996). What we would like to show is that a large class of NLI cases can be straightforwardly dealt with under this approach, which basically treats NLIs as valid theorems in Coq. Also, we believe that in many cases

---

[1] pCIC is a type theory that is rather similar to Luo's UTT, especially after its universe *Set* became predicative since Coq 8.0. A main difference is that UTT does not have co-inductive types. The interested reader is directed to Goguen's Ph.D. thesis (1994) as regards the meta-theory of UTT.

we have given satisfactory and innovative semantic treatments of various semantic phenomena like e.g. the generalization of Σ types not only to adjectives but to VP adverbs, comparatives and factive verbs, which we believe are useful in themselves.[2] Furthermore, it is shown that many NLI cases, in fact all of the NLI cases we have dealt with, can be automatedly performed by an automated combination of Coq's in-built proof tactics.

The paper is structured as follows: in Sect. 2 we present an introduction to formal semantics based on MTTs. Specifically, we concentrate on how to use type theory, in particular the Unified Theory of dependent Types (UTT) with the addition of coercive subtyping, to represent NL formal semantics. In Sect. 3, we present a very short introduction to the way the Coq proof assistant works. In Sect. 4, we discuss the implementation of MTT semantics in Coq, in order to deal with various examples from the FraCas test suite. Lastly, the issue of doing automated theorem proving in interactive theorem provers is discussed in the final section, together with some informal comparison with other relevant work as well as some directions for future work.

2 Formal Semantics in Modern Type Theories

In this section, we give a brief introduction to the formal semantics based on Modern Type Theories (MTTs) (Ranta 1994; Luo 2010, 2012). A MTT is a variant of a class of type theories in the tradition initiated by the work of Martin-Löf (1975, 1984), which have dependent types and inductive types, among other things. We choose to call them Modern Type Theories in order to distinguish them from Church's (1940) simple type theory that is commonly employed within the Montagovian tradition in formal semantics.

Among the variants of MTTs, we are going to employ the Unified Theory of dependent Types (UTT) Luo (1994) with the addition of the coercive subtyping mechanism [see, for example, Luo (1999), Luo et al. (2012) and below]. UTT is an impredicative type theory in which a type $Prop$ of all logical propositions exists.[3] This stands as part of the study of linguistic semantics using MTTs rather than simply typed ones, including the early studies such as Sundholm (1986) and Ranta (1994) inter alia.

2.1 Formal Semantics Based on MTTs: The Basics

In semantics based on MTTs, the basic ways to interpret various linguistic categories are as follows, with basic examples shown in Table 1, where we also compare them to those in Montague semantics:

- A sentence (S) is interpreted as a proposition of type $Prop$.
- A common noun (CN) can be interpreted as a type.

---

[2] See next section for the definition of Σ types.

[3] This is similar to simple type theory where a type t of truth values exists.

Table 1 Examples in formal semantics

	Example	Montague semantics	MTT-based semantics
CN	Man, human	$[\![man]\!]$, $[\![human]\!]$: $e \to t$	$[\![man]\!]$, $[\![human]\!]$: $Type$
IV	Talk	$[\![talk]\!]$: $e \to t$	$[\![talk]\!]$: $[\![human]\!] \to Prop$
ADJ	Handsome	$[\![handsome]\!]$: $(e \to t) \to (e \to t)$	$[\![handsome]\!]$: $[\![man]\!] \to Prop$
MCN	Handsome man	$[\![handsome]\!]([\![man]\!])$	Σm: $[\![man]\!].[\![handsome]\!](m)$: $Type$
S	A man talks	$\exists m$: $e.[\![man]\!](m)\&[\![talk]\!](m)$	$\exists m$: $[\![man]\!].[\![talk]\!](m)$: $Prop$

- A verb (IV) can be interpreted as a predicate over the type D that interprets the domain of the verb (i.e., a function of type $D \to Prop$).
- An adjective (ADJ) can be interpreted as a predicate over the type that interprets the domain of the adjective (i.e., a function of type $D \to Prop$).
- Modified common nouns (MCNs) can be interpreted by means of Σ-types (see below).

In what follows, we shall give further explanations of various aspects of MTT-based semantics, explicating along the way the basic features of MTTs and coercive subtyping. We try to bring out the linguistic relevance of the system used rather than being meticulous as regards the formal details in each case.

2.2 Common Nouns as Types and Many-Sortedness of MTTs

A key difference between formal semantics based on MTTs and Montague semantics lies in the interpretation of common nouns (CNs) which is in turn based on the fact that MTTs are essentially 'many-sorted' logical systems.

In Montague (1974) semantics, the underlying logic [Church's simple type theory (Church 1940)] can be seen as 'single-sorted' in the sense that there is only one type e of all entities. The other types such as t of truth values and the function types generated from e and t do not stand for types of entities. In this respect, there are no fine-grained distinctions between the elements of type e and as such all individuals are interpreted using the same type. For example, *John* and *Mary* have the same type in simple type theories, 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 and as such one can make fine-grained distinctions between individuals and further 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.[4]

An important trait of MTT-based semantics is the interpretation of common nouns (CNs) as *types* (Ranta 1994) rather than sets or predicates (i.e., objects of type $e \to t$)

---

[4] Of course, the need for type fine-grainedness is not an uncontroversial claim. As one of the reviewers notes, there is considerable literature claiming that this type of 'sortal' incorrectness is due to pragmatic factors. However, there is a huge literature claiming to the contrary. This paper takes the stance that type fine-grainedness is indeed needed, following in this respect researchers like Pustejovsky (1995), Asher (2011), Ranta (1994), Fox and Lappin (1990), Bassac et al. (2010) among many others.

as it is the case within the Montagovian tradition. 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. Modified common nouns (MCNs in Table 1) can be interpreted by means of Σ-types, types of dependent pairs.[5] For instance, 'handsome man' can be interpreted as the type $\Sigma m : [\![man]\!]. [\![handsome]\!](m)$, the type of pairs of a man and a proof that the man is handsome.

This many-sortedness (i.e., the fact that there are many types in an MTT) has the welcoming result that a number of semantically infelicitous sentences like *the ham sandwich walks*, which are however syntactically well-formed, can be explained easily given that a verb like *walk* will be specified as being of type $animal \rightarrow Prop$ while the type for the ham sandwich will be $[\![food]\!]$ or $[\![sandwich]\!]$, which is not compatible with the typing for *walk*:

(1) *the ham sandwich* : $[\![food]\!]$
(2) *walk* : $[\![animal]\!] \rightarrow Prop$

The idea of common nouns being interpreted as types rather than predicates has been argued in Luo (2012) on philosophical grounds as well. There, the second author argues that Geach's observation that common nouns in contrast to other categories have criteria of identity that enable common nouns to be compared, counted or quantified, has an interesting link with the constructive notion of 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.[6]

Interpreting CNs as types rather than predicates has also a significant methodological implication: the various subtyping relations one may consider in formal semantics become compatible. For instance, in representing NL semantics, one may introduce various subtyping relations by postulating a collection of subtypes (physical objects, informational objects, eventualities, etc.) of the type of entities (Asher 2011). It is clear that, if CNs are interpreted as predicates as in the traditional Montagovian setting, introducing such subtyping relations would cause major problems: even some basic semantic interpretations would go wrong and it is very difficult to deal with some linguistic phenomena like e.g. copredication satisfactorily. Instead, if CNs are interpreted as types, as in Type Theoretical semantics based on MTTs, copredication can be given a straightforward and satisfactory treatment (Luo 2010).

Remark 1 An anonymous reviewer notes (correctly) that there are cases where *the ham sandwich* : $[\![food]\!]$ might be interpreted as referring to an animate entity, e.g. in the case where *ham sandwich* is interpreted as the man who ordered the ham sandwich. In this instance local coercions can be introduced, an issue discussed in

---

[5] See the discussion in Sect. 2.4 for details w.r.t Σ types.

[6] See Luo (2012) for more details on this.

Luo (2010). Other cases like *John thinks a ham sandwich can walk* are not difficult to treat either. Assuming that John's belief context (in the constructive sense) might have different type declarations than the default context (roughly the current world), one can get a straightforward solution to these cases as well. In effect, John's belief context might involve different type declarations e.g. *the ham sandwich*: $[\![human]\!]$ or *walk*: $[\![object]\!] \rightarrow Prop$.[7] The reviewer further asks how cases like *ham sandwiches do not walk* will be treated in such a framework. There are a number of ways to do that. The first one is to assume a generic operator (widely used in Montagovian frameworks), that besides turning the predicate into a generic predicate, will further introduce the most general type possible to the predicate, i.e. its typing will be: $Gen : \Pi A : CN, (A \rightarrow Prop) \rightarrow ([\![object]\!] \rightarrow Prop)$. In case one considers other negative sentences like *the ham sandwich does not walk* or *a ham sandwich does not walk* to be semantically meaningful, s/he can even define VP negation to be not of type $\Pi A : CN, (A \rightarrow Prop) \rightarrow (A \rightarrow Prop)$ but of type $\Pi A : CN, (A \rightarrow Prop) \rightarrow ([\![object]\!] \rightarrow Prop)$. This will solve the problem. Thus, a number of solutions to this problem exist within MTTs.[8]

2.3 Subtyping in Formal Semantics

As briefly explained above, because of many-sortedness of MTTs, CNs can be interpreted as types. For instance, in a Montagovian setting, all of the verbs below are given the same type $e \rightarrow t$, but in an MTT, we can have

(3) *drive*: $[\![human]\!] \rightarrow Prop$

(4) *eat*: $[\![animal]\!] \rightarrow Prop$

(5) *disappear*: $[\![object]\!] \rightarrow Prop$

which have different domain types. This has the advantage of disallowing interpretations of some infelicitous examples like *the ham sandwich walks*.

However, interpreting CNs by means of different types could lead to serious undergeneralizations without a subtyping mechanism. For instance, consider the interpretation of the sentence 'A man talks' in Table 1: for m of type $[\![man]\!]$ and $[\![talk]\!]$ of type $[\![human]\!] \rightarrow Prop$, the function application $[\![talk]\!](m)$ is only well-typed because we have that $[\![man]\!]$ is a subtype of $[\![human]\!]$.

---

[7] For the notion of belief using MTT semantics, see Ranta (1994) and Chatzikyriakidis and Luo (2013) among others.

[8] Another anonymous reviewer asks what we do in cases of words like *work*, where an indication of two different senses exists, i.e. $work : [\![human]\!] \rightarrow Prop$ and $work : [\![method]\!] \rightarrow Prop$. Even though we have not looked at the problem at its full scale, the second author has proposed the use of overloading with Unit types for these cases, encoding the different senses of the same verb (Luo 2011). Furthermore, on the level of CNs there is considerable work by the authors and colleagues on dot.types. The interested reader should consult (Luo 2011; Chatzikyriakidis and Luo 2012; Xue and Luo 2012; Asher and Luo 2012) for more details.

Coercive subtyping Luo (1999), Luo et al. (2012) provides an adequate framework to be employed for MTT-based formal semantics (Luo 2010, 2012).[9] It 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: $\mathfrak{C}_B[a]$ is legal (well-typed) and equal to $\mathfrak{C}_B[c(a)]$.

As an example, in the case that both $[\![man]\!]$ and $[\![human]\!]$ are base types, one may introduce the following as a basic subtyping relation:

(6) $[\![man]\!] < [\![human]\!]$

In case that $[\![man]\!]$ is defined as a composite Σ-type (see Sect. 2.4 below for details), where $male: [\![human]\!] \rightarrow Prop$:

(7) $[\![man]\!] = \Sigma h: [\![human]\!].\, male(h)$

we have that (6) is the case because the above Σ-type is a subtype of $[\![human]\!]$ via the first projection π_1:

(8) $(\Sigma h: [\![human]\!].\, male(h)) <_{\pi_1} [\![human]\!]$

Equipped with this coercive subtyping mechanism, the undergeneration problems can be straightforwardly solved while still retaining the ability to rule out semantically infelicitous cases like *the ham sandwich walks*. In effect, many-sortedness in MTTs turns out to be superior to single sortedness in the simple type theories, at least in this respect. Furthermore, many inferences involving monotonicity on the first argument in generalized quantifiers can be directly captured using the subtyping mechanism. An inference of the sort exemplified in example (13) below, can be captured given that $[\![man]\!] < [\![human]\!]$:

(9) Some man runs $\Rightarrow$ Some human runs

Thus, an $x: [\![man]\!]$ can be used as an $x: [\![human]\!]$, and as such the inference goes through for 'free' in a way.[10] Another case where the subtyping along with type many-sortedness has welcoming results concerns dot-types, i.e. complex types for common nouns encoding more than one semantic aspect. A classic example is *book*, which has been assumed to have both an informational and a physical aspect. Consider the following sentence:

(10) John picked up and mastered the book.

---

[9] It is worth mentioning that subsumptive subtyping, i.e. the traditional notion of subtyping that adopts the subsumption rule (if $A \leq B$, then every object of type A is also of type B), is inadequate for MTTs in the sense that it would destroy some important metatheoretical properties of MTTs [see, for example, §4 of Luo et al. (2012) for details].

[10] This kind of inferences can be straightforwardly proven in Coq by using a standard analysis for quantifier *some* plus the subtyping relation $[\![man]\!] < [\![human]\!]$.

In the first conjunct, the physical aspect is used, while in the second the informational aspect. We assume the following types for *pick up* and *master*:

$$[\![pick\ up]\!] \ : \ [\![human]\!] \to \text{PHY} \to Prop$$
$$[\![master]\!] \ : \ [\![human]\!] \to \text{INFO} \to Prop$$

Given the subtyping relationship ($*$) as well as contravariance of subtyping for the function types, we get, both of the conjuncts can be interpreted satisfactorily.[11]

2.4 Some Type Constructions in MTTs

In this subsection, we shall discuss several type constructors as well as some more advanced features of MTTs (like for example universes) focusing on the way these can be used in formal semantics.

Dependent Σ-Types One of the basic features of MTTs is the use of Dependent Types. A dependent type is a family of types depending on some values. Here we explain two basic constructors for dependent types, Σ and Π, both highly relevant for the study of linguistic semantics.

The constructor/operator Σ is a generalization of the Cartesian product of two sets that allows the second set to depend on values of the first. For instance, if $[\![human]\!]$ is a type and $male : [\![human]\!] \to Prop$, then the Σ-type $\Sigma h : [\![human]\!].\ male(h)$ 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 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 its dependent case, Σ-types can be used to interpret linguistic phenomena of central importance, like for example adjectival modification (Ranta 1994).[12] For example, *handsome man* is interpreted as a Σ-type (85), the type of handsome men (or more precisely, of those men together with proofs that they are handsome):

(11) $\Sigma m : [\![man]\!].\ [\![handsome]\!](m)$

where $[\![handsome]\!](m)$ is a family of propositions/types that depends on the man m.

Adjectival modification is however notoriously difficult to deal with, given that besides examples of adjectives like *carnivorous* or *handsome*, there exists a number

---

[11] See Luo (2011) for more details on this proposal as well as Xue and Luo (2012) for an implementation of dot-types in the proof assistant Plastic.

[12] Σ-types can also provide the tools for the proper semantic interpretation of the so-called 'Donkey-sentences' (Sundholm 1986).

of other more difficult categories, i.e. privative adjectives like *fake* or non-committal adjectives like *alleged*. Within the Montagovian tradition, these different adjectival categories have been dealt with by using a number of meaning postulates in each case. The authors Chatzikyriakidis and Luo (2013) have proposed a way of dealing with all adjectival categories using the framework presented in this paper. In particular, it was shown that meaning postulates were not needed for most of the cases, the exception being cases of non-committal adjectives. The idea in this paper is to use typing alone to capture what in the Montagovian tradition is done via meaning postulates. We consider capturing inference via typing alone rather via the use of meaning postulates an advantage of MTTs compared to simple typed ones. Also, Σ types have been successfully used in order to provide an adequate account of anaphora, (Ranta 1994). Such a proposal is based on the expresiveness of dependent typing and cannot be maintained in a system where dependent typing is not an option (e.g. in a Montagovian setting).[13]

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 (12) is interpreted as (13):

(12) Every man walks.

(13) $\Pi x : [\![man]\!].[\![walk]\!](x)$

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. For this reason, Π-types can be used, universally quantifying over the universe CN. (14) the type for VP adverbs[14] while (15) is the type for quantifiers:

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

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

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

Note that the above types are polymorphic in nature. In general, MTTs support polymorphism, a mechanism which has been argued by researchers like Fox and Lappin (1990) and Lappin (to appear) to be needed for NL semantics. Type polymorphism is not available in simple type theories.

---

[13] For a recent approach to anaphora using dependent typing see Grudzinska and Zawadowski (2014).

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

Type Universes   An advanced feature of MTTs, which will be shown to be very relevant in interpreting NL semantics, is that of universes. Informally, a universe is a collection of (the names of) types put into a type (Martin-Löf 1984).[15] 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,

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

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, $[\![man]\!]$: CN. Thus, the universe includes the collection of the names that interpret common nouns. For example, in CN, we shall find the following types:

(16) $[\![man]\!]$, $[\![woman]\!]$, $[\![book]\!]$, …

(17) $\Sigma m \colon [\![man]\!].[\![handsome]\!](m)$

(18) $G_R + G_F$

where the Σ-type in (17 is the proposed interpretation of 'handsome man' and the disjoint sum type in (18) is that of 'gun' (the sum of real guns and fake guns—see above).

Having introduced the universe CN, it is now possible to explain (14) and (15). The type in (15) says that for all elements A of type CN, one gets the function type $(A \rightarrow Prop) \rightarrow Prop$. The idea is that the element A becomes the type used. To illustrate how this works let us imagine the case of quantifier *some* which has the typing in (15). The first argument needed, has to be of type CN. Thus *some human* is of type $([\![human]\!] \rightarrow Prop) \rightarrow Prop$ given that the A here is $[\![human]\!]$: CN (A becomes the type $[\![human]\!]$ in $([\![human]\!] \rightarrow Prop) \rightarrow Prop$). Then given a predicate like $walk \colon [\![human]\!] \rightarrow Prop$, we can apply *some human* to get $[\![some\ human]\!]([\![walk]\!]) \colon Prop$.

The idea of universes has been proved useful in giving an account of NL coordination in an MTT. Specifically, in Chatzikyriakidis and Luo (2012), we have introduced a universe of Linguistic Types, $LType$ to capture the flexibility associated with NL coordination. [16]

---

[15] There is quite a long discussion on how these universes should be like. 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 $U \colon U$ and Π-types) is shown to be paradoxical (Girard 1971) 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 [see Luo (1994) for more details].

[16] An anonymous reviewer was questioning the use of MTTs, and their advantages against other systems for formal semantics, i.e. Montague Grammar, DRT and Davidsonian semantics. In this paper, we argue for rich type theories instead of simply typed ones. DRT, Davidsonian semantics as well as other systems for formal semantics are not typed systems. However, fusions of DRT with simple type theory have been attempted successfully (Muskens 2005). In principle, fusions of MTTs with DRT are possible. A discussion on whether MTTs consistute a better alternative than any preceding formal semantics system is, we are afraid, out of the scope of this paper.

3 NL Semantics and Inference in Coq: An Introduction

Coq is a dependently typed interactive theorem prover, implementing the calculus of Inductive Constructions [pCiC, see The Coq Development Team (2007)]. Coq, and in general proof assistants, provide assistance in the development of formal proofs. Specifically, the use of Coq has been extremely fruitful and a number of exciting results have been produced via its use, notably the proof of the four-colour theorem (see Gonthier 2005) and CompCert, a formally verifiable compiler for C (see Blazy et al. 2006) among others. The idea is simple: you use Coq in order to see whether statements as regards anything that has been either pre-defined or user-defined (definitions, parameters, variables) can be proven or not. In order to see how this works, imagine the following three variables . One may want to check whether the following statement involving these variables is a theorem:

(19) $(P \rightarrow Q) \rightarrow (Q \rightarrow R) \rightarrow (P \rightarrow R)$

First, we define P Q and R to be of type $Prop$. Then we "inform" Coq that we want to prove this as a theorem. The command $Theorem$ is used for this reason:

(20) $Theorem\ Propositional : (P \rightarrow Q) \rightarrow (Q \rightarrow R) \rightarrow (P \rightarrow R)$

The above will put Coq into proof-mode, where the user is asked to interactively guide the assistant to the proof. In order to do this, the user has a number of proof-tactics that s/he can use. More complicated tactics can be further defined and a number of libraries with complementary tactics exist. For the case interested, using the $intro$ tactic three times will result in the introduction of $(P \rightarrow Q)$, $(Q \rightarrow R)$ and P as assumptions. We can now use these assumptions to see whether we can construct a proof for the conclusion. The result will be:

```
1 subgoal

  H  :  P  ->  Q
  H0  :  Q  ->  R
  H1  :  P
  ============================
  R
```

Now, the tactic *apply* can be used. This tactic takes an argument which can be decomposed into a premise and a conclusion (e.g. $Q \rightarrow R$), with the conclusion matching the goal to be proven (R), and creates a new goal for the premise. Thus, $apply\,H0$ will create the new goal Q:

```
1 subgoal

  H  :  P  ->  Q
  H0  :  Q  ->  R
  H1  :  P
  ============================
   Q
```

We can do the same with H, thus apply $H0$ will P as the goal:

```
   1 subgoal

  H : P -> Q
  H0 : Q -> R
  H1 : P
  ============================
   P
```

At this point, the command *assumption* can be used, which matches the conclusion with an identical premise, i.e. $H1$. With this, Coq notifies us that a proof has been found:

```
1 subgoal

  H : P -> Q
  H0 : Q -> R1
  H1 : P
  ============================
   P


Propositional < assumption.
Proof completed.
```

But how can such an assistant be used in order to reason about NL semantics? As already said, Coq implements an MTT (pCiC). For this reason it is highly suitable to implement our MTT semantics, given that this is quite close to UTT with coercive subtyping, i.e. the MTT used in this paper. Indeed, this has been already noted by Luo et al. and some first attempts at implementing MTT semantics in Coq have been made (Luo 2010, 2011; Chatzikyriakidis and Luo 2012). However, this is not all that Coq has to offer. Given that Coq is a powerful theorem prover, it can further reason about the implemented semantics. In fact, one may very well use Coq's proving ability to prove valid NL inferences, in the same sense it is used for proving valid mathematical or logical theorems. Given that semantic entailment corresponds to an implication relation between two different semantic structures, entailment relations can be translated into constructed theorems that need to be proven. In such a context, a valid semantic entailment will very simply mean the implication relation between the two semantic structures is a valid theorem. A very simple case of semantic entailment, that of example (21), will therefore be formulated as the following theorem (named *ex*) in Coq (22):

(21)   John walks $\Rightarrow$ some man walks

(22)   Theorem ex: John walks $\rightarrow$ some man walks

Then depending on the semantics of the individual lexical items one may or may not prove the theorem that needs to be proven in each case. Inferences like the one shown in (22) are easy cases in Coq. Assuming the semantics of *some* which specify that

given any A of type CN and a predicate of type $A \rightarrow Prop$, there exists an $x : A$ such that $P(x) : Prop$, such cases are straightforwardly proven.

A few notes about the lexical entries. We use Coq's $Prop$ type, corresponding roughly to the type of truth-values (t) in Montague Semantics. We define CN to be Coq's *Set* Universe and interpret CNs like *man, human* as being of type CN (thus we have for example *man*, *human*: CN).[17] Verbs are defined as predicates requiring arguments of type $A : CN$. The exact type of this A argument depends on the verb itself. For example, *walk* is defined as being of type $Animal \rightarrow Prop$. Subtyping relations are supported by Coq's coercion mechanism and thus all the relevant subtyping relations can be declared.[18] Adjectives are defined as predicates, and adjectival modification as Σ types (see the discussion in Sect. 2). Quantifiers and VP adverbs are defined as types ranging over the universe CN [see (14) and (15)]. For the example at hand, the following are declared:

```
CN:=Set.
Parameter Man Human Animal: CN.
Parameter John: Man.
Axiom mh: Man->Human. Coercion mh: Man>->Human.
Axiom ha: Human->Animal. Coercion mh:Human>->Animal.
Definition some:= fun A:CN, fun P:A->Prop=> exists x:A, P(x).
Definition walk: Animal->Prop
```

We have introduced CN as being Coq's *Set* type, declared Man, $Human$ and $Animal$ to be of type CN, further introduced the relevant subtyping relations and lastly introduced $walk$. With $walk$ as being of type $[\![human]\!] \rightarrow Prop$ and $John$ as being of type $[\![man]\!]$ with $[\![man]\!] < [\![human]\!]$, we can prove the theorem in (22) quite easily. We first use the proof tactic *intro* to move the implicans to the hypotheses. Then, we apply *unfold* to *some* (*unfold some*). *Unfold* does what it says: it unfolds the definition associated with a lexical entry (if there is a definition).:

```
ex < unfold some.
1 subgoal
===========================
walk John -> exists x : Man, walk x
```

We use *intro* to move the antecedent as a premise. Then, we can existentially instantiate $x : Man$ with $John : Man$:

```
ex < intro.
1 subgoal
H : walk John
===========================
exists x : Man, walk x
ex < exists John.
```

---

[17] In Luo's MTT, CN is the universe containing the names that interpret CNs. Since the possiblity of introducing new universes is not an option we approximate this idea by having CN being of type set.

[18] Note that ambiguous paths are not allowed and as such given two types A and B (with A, B: CN), there is no possiblity of defining both $A < B$ and $B < A$.

```
1 subgoal
H : walk John
============================
walk John
```

The tactic *assumption* finishes the proof. This example as well as all of the examples discussed in this paper can also be proven automatically by using Coq's predefined automation tactics or via using user-defined automation tactics.[19]

4 NL Inference with FraCas Examples in Coq

The FraCas Test Suite (Cooper et al. 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 NL inference 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. For instance,

(23)   Smith, Jones and Anderson signed the contract.
       Did Jones sign the contract? [Yes]
(24)   No delegate finished the report.
       Did any delegate finished the report on time? [No]

The examples are quite simple in format but are designed to cover a very wide spectrum of semantic phenomena, e.g. generalized quantifiers, conjoined plurals, tense and aspect related phenomena, adjectives and ellipsis, among others.

In this section, we use a number of these examples in FraCas (except the last Sect. 4.8 on collective predicates) to exemplify the way MTT-semantics implemented in Coq can effectively deal with a number of NLI cases. The formulation we are going to follow will transform the question in each example of the FraCas test suite into a declarative hypothesis that needs to be proven.[20] All of these examples are formalised in Coq;[21] the Coq code and proof for the first example, as described in Sect. 4.1 below, can be found in Appendix "An Example from the FraCas Test Suite". In the last part of the section, the results of an evaluation against a subset of the FraCas test suite is shown, highlighting the most important cases.

Remark 2 The current paper deals with quite a lot of semantic phenomena that each of them deserve a discussion and thorough analysis on their own right. Thus, the paper deals with a number of diverse semantic issues ranging from adjectives and

---

[19] See the discussion on automation in Sect. 5.

[20] The same modification can be also found in MacCartney (2009). In general, if one uses a theorem prover to deal with NL inference [e.g. analyses in the style of Blackburn and Bos (2005), Bos and Markert (2005, 2006)] such modifications are necessary.

[21] The source codes can be obtained by sending an email request to the first author: stergios.chatzikyriakidis@cs.rhul.ac.uk.

quantifiers to adverbs, factive complements and reasoning with tense and aspect. The reader should have in mind that all these issues are far from solved in the formal semantics literature. For the needs of this paper, we cannot go into a thorough analysis for each and every issue separately. As such, some of the proposals might not be able to cover the whole range of phenomena associated with these constructions. For example, the account we have presented here for adverbs is restricted to what we call veridical adverbs, given these type of adverbs are the ones involved in the FraCas test suite. There is a vast number of complexities associated with the semantics of adverbs and the non-homogeneity of the specific class is notorious (Maienborn and Schafer 2011).[22] The reader is advised to have these remarks in mind in reading the main core of the paper.

4.1 Quantifiers and Monotonicity

A great deal of the FraCas examples are cases of inference that result from the monotone properties of quantifiers. Examples concerning monotonicity on the first argument are very easily treated in a system encoding an MTT with coercive subtyping, by employing the subtyping relations between CNs (c.f., Sect. 2.3).

To put this claim in context, let us look at the following example (3.55) from the FraCas test suite:

(25)   Some Irish delegates finished the survey on time.
       Did any delegate finish the report on time [Yes]

In an MTT-based semantics, the sentences in the above example become:[23]

(26) $\exists s : [\![Irishdelegate]\!], [\![on\_time]\!][\![finish]\!](s, (the\,report))$
       Let Q $= \exists s : [\![delegate]\!], [\![on_time]\!][\![finish]\!](s, (the\,report))$. Is Q true?

where $[\![finish]\!]: [\![object]\!] \rightarrow [\![human]\!] \rightarrow Prop$, $[\![on\_time]\!]: forall\,A : CN, (A \rightarrow Prop) \rightarrow (A \rightarrow Prop)$,[24] $[\![Irishdelegate]\!] < [\![delegate]\!] < [\![human]\!]$ and $[\![report]\!] < [\![object]\!]$. Some notes of explanation on the subtyping relation $[\![Irishdelegate]\!] < [\![delegate]\!]$: we follow the second author's implementation of Σ-types as the Coq record types (Luo 2011), and we interpret the modified CN *Irish delegate* as the following record type:

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

---

[22] See however Chatzikyriakidis (2014) for a more thorough look at adverbs from an MTT perspective.

[23] For this first example, we shall detail its formal semantics in a type theory. You can also find the Coq implementation of this in Appendix "An Example from the FraCas Test Suite". For the examples later on, we shall omit such details.

[24] This is the type for VP adverbs used in Luo (2011). We will see later on that a slightly modified type will be used for VP adverbs. For the moment, we keep this type given that it does not play any role whatsoever in proving the inference.

It is a well-known fact that dependent record types are equivalent to Σ types, so without having to get into details on the specific Coq syntax, this is just equivalent to $\Sigma d : [\![delegate]\!].[\![Irish]\!](d)$. Given the subtyping relation between *Irish delegate* and *delegate* (reflected by means of the syntactical notation :> in Coq), (25) is correctly captured.

It is straightforward to prove the above formula Q, using pretty much the same tactics as these are discussed in Sect. 3, namely *intro*, *apply* and *exists*. The exact way this is done, is shown in the Appendix "An Example from the FraCas Test Suite".

Similar considerations apply to the examples like the one shown below (FraCas example 3.49):

(28) A Swede won a Nobel Prize.
       Every Swede is a Scandinavian.
       Did a Scandinavian win a Nobel Prize? [Yes]

Given the subtyping relation *Swede* $<$ *Scandinavian* $<$ *human* the above inference is correctly predicted. Note that the second premise is expressed via means of the subtyping relation *Swede* $<$ *Scandinavian*. Specifically, we have $[\![win]\!] : [\![object]\!] \rightarrow [\![human]\!] \rightarrow Prop.$ and $prize < object$. The subtyping relation $[\![Nobel\_Prize]\!] < [\![prize]\!]$ is true because $[\![Nobel\_Prize]\!]$ may be defined as $\Sigma p : [\![prize]\!]. Nobel(p)$ which is a subtype of $[\![prize]\!]$ via the first projection (c.f., Sect. 2.3).

Adverbial Modifications: A Digression We now move to cases involving monotonicity on the second rather than the first argument. Such an example with upwards monotonicity is shown below:

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

Monotonicity on the second argument can be treated in a similar way as above. However, the above example (29) is a little bit trickier since an adjunct, i.e. the PP *on time*, is involved. As mentioned in Sect. 2.4, VP adverbs such as *on time* are given the following type (to repeat (14) here):

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

In order to consider such adverbial phrases in inference, we make use of an auxiliary object ADV:

(31) $ADV : \Pi A : \text{CN}\Pi v : A \rightarrow Prop. \Sigma p : A \rightarrow Prop.\forall x : A.p(x) \supset v(x)$

For any common noun A and any predicate v over A, $ADV(A, v)$ is a pair (p, m) such that for any $x : A$, $p(x)$ implies $v(x)$. Taking the sentence (29) as an example,

for the CN *delegate* and predicate $[\![finish]\!]$,[25] we define

(32) $on\ time = \lambda A\colon \text{CN}.\lambda v\colon A \to Prop.\ \pi_1(ADV(A, v))$

which is of type (30). As a consequence, for instance, any delegate who finished the survey on time did finish the survey.

Note that the Σ-type in (31) might be considered as a general form of conjunction and, thinking in this way, it is not difficult to see that the above analysis is intuitively compatible with a Davidsonian analysis of VP adverbs where the adverb modifies an event argument (Davidson 1967).

A similar example involving downward monotonicity on the second argument is shown below:

(33) No delegate finished the report.
      Did any delegate finish the report on time [No]

In the above case, the only thing we need to do is turn the hypothesis into its negation and then try to prove it, as we did in Coq.

4.2 Conjoined Noun Phrases

In the section of the FraCas test suite involving inferences with conjoined NPs, one can find the following example:

(34) Smith, Jones and Anderson signed the contract.
      Did Jones sign the contract? [Yes]

In Chatzikyriakidis and Luo (2012), we have proposed a polymorphic type for binary coordinators that extends over the constructed universe $LType$, the universe of linguistic types. This can be extended to *n*-ary coordinators. For example, the coordinator *and* may take three arguments, as in the premise of (34). In such cases, the type of the coordinator, denoted as and_3 in semantics, is:

(35) $and_3\colon \Pi A\colon LType.\ A \to A \to A \to A.$

Intuitively, we may write this type as $\Pi A\colon LType.\ A^3 \to A$. For instance, the semantics of (34) is (36), where c is 'the contract':

(36) $[\![sign]\!](and_3(s, j, a), c)$

In order to consider such coordinators in reasoning, we consider the following auxiliary object (similarly as in the last subsection when we consider adverbial phrases) and

---

[25] Note that $[\![finish]\!]\colon [\![human]\!] \to Prop < [\![delegate]\!] \to Prop.$

define and_3 as follows:

(37) $AND_3 : \Pi A : LType. \Pi x, y, z : A. \Sigma a : A. \forall p : A \rightarrow Prop. p(a) \supset$
      $p(x) \wedge p(y) \wedge p(z)$.

(38) $and_3 = \lambda A : LType.\lambda x, y, z : A. \pi_1(AND_3(A, x, y, z))$

Having defined the coordinators such as *and* in such a way, we shall have the desired inference as expected. For example, from the semantics (36), we can infer that 'Jones signed the contract', the hypothesis in (34).[26]

Coordinators such as *or* can be defined in a similar way. More complex examples like the one shown below can be also proven:

(39) Either Smith, Jones or Anderson signed the contract.
      If Smith and Anderson did not sign the contract, did Jones sign the contract?
      [Yes]

Remark 3 It's worth remarking that a general definition of a logical conjunction *And* for arbitrary n-ary arguments is possible. We start with the typing:

(40) $And : \Pi n : Nat. \Pi A : LType. \Pi v : (Vec(A, n + 2)) \rightarrow Prop$.

We can define *And* by induction on Nat:

(41) $And((2, A)[a, b]) = a \wedge b$
      $And(n + 1, conc_V(a, v)) = a \wedge (And\ (n\ v))$

Based on the above, a general auxiliary AND can be defined as follows:

(42) $AND : \Pi n : Nat. \Pi A : LType. \Pi v : (Vec(A, n + 2)). \Sigma a : A. \forall p : A \rightarrow$
      $Prop. p(a) \supset (And\ (n + 2) A\ (map_V v))$
      where $map_V : (n : Nat)(A, B : Type)(f : A \rightarrow B)Vect(A, n) \rightarrow$
      $Vect(B, n)$ is defined as follows:
      $map_V(0, A, B, f, nil_A) = nil_B$
      $map_V(n + 1, A, B, f, cons_V(a, v)) = cons_V(f(a), map_V(n, A, B, f, v))$

4.3 Adjectives

Inferences involving adjectives pose a number of difficulties given the semantic asymmetries associated with different classes of adjectives. The semantics of adjectives is a notoriously difficult issue in theoretical semantics and a number of approaches have been put forth, particularly in a classical Montagovian setting [see, for example, Montague (1973), Kamp (1975) and Partee (2007, 2010)]. In Modern Type Theories,

---

[26] A note about Coq is in order here: building new universes is not an option in Coq (or, put in another way, Coq does not support us to build a new universe). Instead, we shall use an existing universe in Coq in conducting our examples for coordination.

Σ-types have been proposed for intersective adjectives Ranta (1994) and, recently, the current authors have studied adjectives more systematically using the framework used in this paper (Chatzikyriakidis and Luo 2013).

The FraCas test suite uses different terminology than those usually found in the literature on the formal semantics of adjectives. The basic classification is between affirmative and non-affirmative adjectives:

(43) Affirmative: Adj(N) $\Rightarrow$ (N)

(44) Non-affirmative: Adj(N) $\not\Rightarrow$ (N)

We shall follow this latter terminology in this paper.

4.3.1 Affirmative and Non-affirmative Adjectives

Cases of affirmative adjectives are handled well with the existing record mechanism already used for adjectives. The following inference as well as similar inferences are correctly captured, given that a CN modified by an intersective adjective is interpreted as a Σ-type which is a subtype of the CN via means of the first projection:

(45) John has a genuine diamond $\Rightarrow$ John has a diamond.

Non-affirmative adjectives involve cases like *former*. The problem with these types of adjectives is that they do not give rise to categorical intuitions as regards inference. Thus, the following inference is valid for some but non-valid for some others:[27]

(46) John is a former president $\Rightarrow$ John is not a president

The same goes for adjectives like *fake*.[28] We leave the discussion concerning adjectives like *former* until temporal inference is going to be discussed. We will then propose an account, assuming that *former* is indeed non-affirmative.

4.3.2 No Comparison Class Adjectives

Adjectives like *four-legged* do not need reference to a comparison class (FraCas 3.202):

(47) Every mammal is an animal.
Is every four-legged mammal a four-legged animal? [Yes]

Assuming that four_legged is of type $Animal \rightarrow Prop$ and given that $Mammal < Animal$, the above inference is correctly predicted.

---

[27] This is something that has been noted in the literature, see Partee (2007). Note that in the FraCas test suite, the inference in (46) is valid.

[28] In the case of *fake*, Partee (2010) tried to provide an account where *fake* is treated as a subsective adjective, i.e. affirmative in the classification given in the FraCas test suite, via using the disjoint union type.

4.3.3 Opposites

This section deals with adjectives of the same comparison class which are opposites of one another like e.g. *large* and *small*. For these adjectives, the following inferences hold:

(48)   Small(N) $\Rightarrow$ $\neg$ Large(N).

(49)   Large(N) $\Rightarrow$ $\neg$ Small(N)

(50) $\neg$ Small(N) $\nRightarrow$ Large(N).

(51) $\neg$ Large(N) $\nRightarrow$ Small(N)

What is the most difficult part here is the avoidance of the inferences (50) and (51). Something which is not small might not be large, given that sizes do not come in the form of a binary opposition. Thus, a way to treat this is to introduce another size, let us say *normalsized*, and have *small* hold in case the negation of both *large* and *normalsized* hold:[29]. We introduce the following:

(52)   Definition Small:= fun A:CN,fun x: A=> ~Large x $\wedge$ ~Normalsized x.

This approach is successful in getting the inferences as regards opposites right.[30]

4.3.4 Extensional Comparison Classes

Adjectives like *large* and *small* are only relevant for the comparison class they refer to. Thus, inferences like the following are found:

(53)   All mice are small animals.
          Mickey is a large mouse.
          Is Mickey a large animal? [No]

In order to deal with these cases, we introduce a polymorphic type for adjectives like these ranging over the universe CN.[31] The type for *large* is shown in (54), which is (55) in Coq's notation:

(54) $large : \Pi A : \text{CN}. \ A \rightarrow Prop$

(55)   Parameter large : forall A:CN, A$\rightarrow$Prop

Using the above type, we have many instances of *large* depending on the choice of A. If $A = Man$ then we get $large(Man) : Man \rightarrow Prop$; if $A = Animal$, we

---

[29] Of course, depending on context more fine grained distinctions might be needed but the idea is applicable to these cases as well.

[30] *Small* is defined after *Large* has been declared. The opposite is also possible, i.e. defining *Large* after *Small* has been declared first. This might seem strange from a theoretical point of view, but for implementation purposes it is not.

[31] This is based on the authors' analysis of subsective adjectives (Chatzikyriakidis and Luo 2013).

get $large(Animal) : Animal \rightarrow Prop$, and so on. In this respect, we get different 'larges' as such for different *As*. With this, one can capture the meaning of subsective adjectives, i.e. that if something is *A* (where *A* an adjective), it is only large for its class denoted by the CN (a large mouse is thus only large as a mouse, but not as an elephant). This way of treating subsective adjectives will correctly account for the inferences like that in (53).[32]

4.3.5 A Note on Intersective Adjectives

Intersective adjectives comprise a class of adjectives in the theoretical literature on adjectives which can be characterized by the following inferential schema:

(56)   Affirmative: $Adj(N)(x) \Rightarrow Adj(x) \wedge N(x)$

Thus, *carnivorous man* means something that is *carnivorous* and a *man*. With intersective adjectives, one should be able to get inferences like the following:

(57) Adj_{inter} man $\Rightarrow$ Adj_{inter} human

A concrete example would be *carnivorous man* implying *carnivorous human*. Given that coercions according to Luo's MTT propagate via the various type constructors, we have: $\Sigma([\![man]\!], carnivorous < \Sigma([\![human]\!], carnivorous)$.[33]

4.4 Comparatives

Comparatives such as *shorter than* can be given semantics either directly or by means of an explicit measure. We shall consider both alternatives.

  Comparatives without Measures. We shall consider *shorter than* as a typical example. Intuitively, *shorter than* should be of type $Human \rightarrow Human \rightarrow Prop$ as in the following example:

(58)   Mary is shorter than John.

We assume that there be a predicate $short : Human \rightarrow Prop$, expressing that a human is short. Intuitively, if Mary is shorter than John and John is short, then so is Mary. Furthermore, one should be able to take care of the transitive properties of comparatives. Thus, if *A* is *COMP* than *B* and *B* is *COMP* than *C*, then *A* is also *COMP* than *C*. All these can be captured by considering *SHORTER_THAN* of the

---

[32] The interested reader is directed to Chatzikyriakidis and Luo (2013) for more information on the treatment of subsective as well as the other types of adjectives in MTT with coercive subtyping.

[33] In Coq, we cannot have the first projection as a general coercion. Instead, we have to declare it for the instances we need. This is a weakness of Coq that does not allow us to implement the more general treatment. Such a general coercion is possible to get declared in Plastic, an interactive theorem prover that implements Luo's UTT and coercive subtyping (Callaghan and Luo 2001).

following Σ-type and define *shorter than* to be its first projection:

(59) $SHORTER\_THAN: \Sigma p: Human \rightarrow Human \rightarrow Prop. \forall h_1, h_2, h_3 :$
       $Human. \ p(h_1, h_2) \wedge p(h_2, h_3) \supset p(h_1, h_3) \wedge \forall h_1, h_2 : Human.p(h_1, h_2) \supset$
       $short(h_2) \supset short(h_1).$

(60) $[\![shorter \ than]\!] = \pi_1(SHORTER\_THAN)$

With the above, we can easily show that the inferences like (61) can be obtained as
expected.

(61)   John is shorter than George.
       George is shorter than Stergios.
       Is John shorter than Stergios? [Yes]

Comparatives with Measures. In giving an analysis of compratives, one may consider
measures, taking into consideration different degrees of the measure used in each case,
e.g height in the case of comparatives like *short*, weight in the case of adjectives like
heavy or speed in the case of adjectives like *fast*. For example, we can analyze *shorter
than* as a relation between nouns that do come with implicit measures, in which the
first noun has less height than the second.

   Such measures can be taken care of explicitly by extending the above treatment by
dependent typing over measures. Let's consider *shorter than* as an example, taking
heights to be measured by the type *Height* of numbers such as 1.70.[34] We are then
led to consider the family of types *Human: Height $\rightarrow$ Type* indexed by heights:
Human(n) is the type of humans of height *n*. Then, *shorter than* is defined as fol-
lows:[35, 36]

(62) $SHORTER\_THAN: \Pi i, j: Height. \ \Sigma p: Human(i) \rightarrow Human(j) \rightarrow$
       $Prop. \forall h_1 : Human(i) \forall h_2 : Human(j). \ p(h_1, h_2) \leftrightarrow i < j.$

(63) $[\![shorter \ than]\!](i, j) = \pi_1(SHORTER\_THAN(i, j)): Human(i) \rightarrow$
       $Human(j) \rightarrow Prop$

   We can now take care of the inferences like (64) as expected:

(64)   John is shorter than George.
       George is 1.70.
       Is John less than 1.70 tall? [Yes]

---

[34] Here we do not spell out the type *Height*. One might take *Height* to be the type of natural numbers
and use 170 to stand for 1.70, etc.

[35] The transitive properties of comparatives are not encoded in this example for reasons of simplicity. One
may very well do so having as a guide the previous entry without measures.

[36] This is a bi-implcation, given that if the height of human *x* is less than the height of another human *y*,
then it is also the case that *x* is shorter than *y*. The definition also works as an implication.

To see the details, the semantics of the above sentences are given in (67), where J and G are the semantics of *John* and *George*, involving height parameters:

(65) $J, G : \Sigma x : height.HUMAN(x)$

We can further define j and g as the second projection of J and G respectively:

(66) $j, g : \pi_2(J, G)$

With these at hand, (64) can be formulated as follows:

(67) $[\![shorter\ than]\!](J, G)$.
    $g = 1.70$.
    Is Q true, where $Q = j < 1.70$?

It is easy to show that the above inference (67) can be proven in Coq.

It may be worth remarking that superlatives can be defined once comparatives are: for example, for any $x : Human$, $shortest(x)$ if and only if x is shorter than or equal to any $y : Human$. A similar treatment can account for the rest of the examples involving comparatives in the FraCas test suite.

Remark 4 It is of course highly desirable to generalize the account of comparatives to other similar adjectives. This is not difficult. For example, in the case with measures one can define a general auxiliary object $COMP$ that will, besides height, deal with other quantities expressed as natural numbers as well, e.g. weight, speed etc. Similarly to the type Height, we can accordingly define the types of Weight, Speed to be the type of natural numbers. Then, a general COMP auxiliary will be possible:

(68) $COMP: \Pi i, j: nat.\ \Sigma p: Human(i) \rightarrow Human(j) \rightarrow Prop.\ \forall h_1 :$
    $Human(i) \forall h_2 : Human(j).\ p(h_1, h_2) \leftrightarrow i < j.$

Then comparatives like smaller_ than, thinner_ than, slower_ than can all of them be defined as:

(69) $[\![comp]\!](i, j) = \pi_1(COMP(i, j)): Human(i) \rightarrow Human(j) \rightarrow Prop$

4.5 Temporal Reference

A way to deal with temporal reference without employing a temporal logic of some sort, is to introduce a type $Time$ of times to deal with the extra parameter of *Time* [see e.g. Ranta (1994) for such a view].

With such a type $Time$, one provides a very simple model of tense and, over $Time$, we have a precedence relation $\leq$ and a specific object $now: Time$, standing for 'the current time' or 'the default time'. Simple tenses like the simple present or the simple past can then be easily captured.[37] Also, in this model, verbs are assumed

---

[37] One may even employ this model to capture composite tenses like the past perfect, but we do not discuss this here. See Ranta (1994) for an idea of how this can be done within such a framework.

to involve a *Time* argument as well.[38] Thus a verb like *walk* is not simply of type $\llbracket human \rrbracket \rightarrow Prop$ anymore, but rather of $\llbracket human \rrbracket \rightarrow Time \rightarrow Prop$.

The type *Time* can be specified as an inductive type in an MTT, where one may consider the following as one of its constructors:[39]

(70) $date : DATE \rightarrow Time$

where $DATE$ consists of the triples (y, m, d) where y ranges over integers to represent years, m over Jan to Dec to represent months, and d over the days $1, 2, \ldots$ to represent days.[40] For example, $date(1970, Oct, 5)$ stands for the time 'Oct 5, 1970'.

Now, consider the inferences like the following example:

(71) Last year John signed the contact.
      Today is June 18, 2013.
      Did John sign the contract in 2012? [Yes]

With the above, the above sentences in (71) are interpreted as those in (72), where $c = \llbracket the\ contract \rrbracket$:

(72) $\exists t : Time, \exists m : month, \exists d : day.date(year(now) - 1, m, d) \wedge m \leq 12 \wedge d \leq 30 \wedge sign(j, c, t)$.
      $now = date(2013, June, 18)$.
      Is Q true, where $\exists t : Time, \exists m : month, \exists d : day.date(2012, m, d) \wedge m \leq 12 \wedge d \leq 30 \wedge sign(j, c, t)$?

This can now be shown to be valid inference in Coq. Cases involving temporal adverbs like *yesterday, today* or PPs like *next month, next year* can be treated accordingly.

Similarly examples like the following can be accounted for, assuming that currently identifies the time of the proposition to be equal with the *default time*. The *Time* argument of the proposition has already been identified as being the *default time* via means of the present tense verb and as such, examples like the one below are very easily proven to be valid inferences:

(73) ITEL has a factory in Birmingham.
      Does ITEL currently have a factory in Birmingham? [Yes]

The sections in the FraCas test suite that deal with *in* and *for* adverbials need a solid account of lexical aspect as well as a fuller account of tense which at the present

---

[38] The assumption that verbs involve an event/situation argument goes back at least to Davidson (1967). See Davidson (1967) and reference therein, for a history of events/situations in linguistic theory.

[39] An inductive type is specified by a number of constructors whose types must be strictly positive [see, for example, Chapter 9 of Luo (1994) for formal details]. *Time* as an inductive type may have other constructors but we only detail *date* here.

[40] Note that, in detail, the range of days depends on the year and month. This can be represented by means of dependent types: the type $Day(y, m)$ depends on $y : Year$ and $m : Month$: for example, because there are only 28 days in Feb of 1970, $Day(1970, Feb) = \{1, 2, \ldots, 28\}$, the enumeration type consisting of 1, 2, …, 28 only. Formally, $DATE$ can be defined as $\Sigma y : Year. \Sigma m : Month. Day(y, m)$.

we do not have to offer. We leave these sections unresolved until such an account is provided. However, some of the inferences can be effectively dealt with in pretty much the same way as the monotone on the second argument examples. One such example is shown below:

(74)   Smith lived in Birmingham for two years.
       Did Smith live in Birmingham? [Yes]

Defining *for two years* in the same sense as a veridical VP adverb like *on time*, can provide us with a correct prediction for the above example. Again, we should stress that a full account of *for* and *in* adverbials needs a solid account of aspect that we at the moment do not have, so we leave this as an issue for future research.

4.5.1 The Case of Former

Adjectives such as *former* or *past* may be treated in the temporal model we have considered.[41] We assume that some CNs are indexed by the time parameter. For example, instead of being interpreted just as a type, a CN like *president* is interpreted as a family of types indexed by $t : Time$:

(75) $president(t) : $ CN.

For example, as $now : Time$ stands for the 'current time', $president(now)$ is the president at the current time.

With the above mechanisms available, we can now interpret CNs modified by *former* as follows:,[42]

(76) $[\![former\ president]\!] = \neg president(now) \wedge \exists t : Time.\ t < now \wedge president(t).$

In general, we have $[\![former]\!] : (Time \rightarrow$ CN$) \rightarrow$ CN,[43] obtained by abstracting *president* in the above definition: for any $p : Time \rightarrow$ CN,

(77) $[\![former]\!](p) = \neg p(now) \wedge \exists t : Time.\ t < now \wedge p(t).$

With $president : Time \rightarrow$ CN, we have $[\![former\ president]\!] = [\![former]\!]$ $([\![president]\!])$.

This kind of analysis will predict that former president entails a past president but not a current president.

---

[41] Another approach to dealing with such adjectives is to follow Partee (2007) and assume that *former* behaves similarly to privative adjectives like *fake* or *imaginary*. If so, one may follow the proposed MTT-interpretation by the authors to use the disjoint union type to interpret *former*. See Chatzikyriakidis and Luo (2013) for details.

[42] For understandability of the readers who are unfamiliar with MTTs, we abuse the notation here, using $\neg A$ to stand for $A \rightarrow \emptyset$, $\wedge$ for $\times$ and $\exists$ for Σ. One may ignore these formal details.

[43] In Coq this is translated as $(Time \rightarrow$ CN$) \rightarrow Prop$ given that definitions always end in Prop.

4.6 Epistemic, Intensional and Reportive Attitudes

This section involves verbs taking a sentential argument. The difference is between verbs that presuppose the truth of their complements and verbs that do not:

(78) Smith knew that Itel had won the contract 1991.
     Did Itel win the contract in 1991? [Yes]

(79) Smith believed that Itel had won the contract 1991.
     Did Itel win the contract in 1991? [Don't know]

Again, we will not dwell on a discussion on how suitable semantics for attitude verbs should be given. There are so many issues to take into consideration in this respect, starting with questions as general as 'what is belief', that such a discussion cannot be carried out here. However, we can provide an account of these types of inferences without necessarily solving the issues associated with the semantics of Attitude verbs.

What we need is to encode that some epistemic verbs presuppose their argument's truth while others do not. For instance, *know* belongs to the former class and its semantics is given as follows:

(80) $KNOW = \Sigma p : Human \rightarrow Prop \rightarrow Prop. \forall h : Human \forall P : Prop. \, p(h, P) \supset P$

(81) $[\![know]\!] = \pi_1(KNOW)$

With this, the inference (78) can be obtained as expected. Intensional verbs like *believe* on the other hand do not imply their arguments and inferences like (79) cannot be shown to be valid inferences.

In the FraCas test suite there are also examples concerning 'veridicality'; this is basically the property that verbs like *know* show—'know P' $\Rightarrow$ P, so we do not need to discuss these cases again.

4.7 Substitution and Existential Instantiation

Substitution refers to the ability of substituting two equivalent terms and retaining the meaning after substitution, as (82) shows:

(82) Smith saw Jones sign the contract.
     Jones is the chairman of ITEL
     Did Smith see the chairman of ITEL sign the contract?[Yes]

Substitutions like those in the second premise above can be easily done in Coq via the *replace* tactic. Thus, cases like these are easy to capture.

There are also examples where existential quantifiers and their instantiations are involved. For example,

(83)   Smith knows that Jones signed the contract.
        Jones is a person.
        There is a person such that Smith knows he signed the contractn [Yes]

Existential quantification is introduced to the semantics because of the second sentence; this becomes clear if we spell out the semantic interpretations of the sentences in (83) as those in (84) below, where $c = [\![the\ contract]\!]$:

(84) $[\![know]\!](s, sign(j, c))$.
        $\exists x : Person.\ j = x$.
        $\exists x : Person.\ [\![know]\!](s, sign(x, c))$.

It is easy to see that the first two imply the third.

4.8 Collective Predication

We want to be able to get the following inferences (note that these cases are not part of the FraCas test suite):

(85)   Stergios and Zhaohui met $\Rightarrow$ Stergios met Zhaohui and Zhaohui met Stergios

(86)   Stergios and Zhaohui hit each other $\Rightarrow$ Stergios hit Zhaohui and Zhaohui hit Stergios

(87)   Stergios and Zhaohui are Greek and Chinese respectively $\Rightarrow$ Stergios is Greek

For such collective predicates, we use the Vector-analysis proposed by the authors in Chatzikyriakidis and Luo (2012). Verbs like *meet* in their collective guise take a vector argument with at least two elements (i.e., an object of type $Vec(Human, n + 2)$), as given in (88).[44] Thus, reciprocal predicates like *meet* take one vector argument (with n at least 2).[45] This account can also give us a natural treatment of reflexives like *each other*. The idea is that *each other* in English turns a transitive predicate into an intransitive one whose sole argument is a vector whose length is at least 2, as in (89). Thus, the two arguments of a transitive verb like *kill*, say $John$ and $Mary$ in *John killed Mary*, are put together into a single vector argument, with the verb turning into an intransitive verb. Lastly, *respectively* can be seen as a big functor which takes two vector arguments and returns a proposition, as in (90).

(88) $[\![meet]\!]: \Pi n: Nat.\ Vec([\![human]\!], n + 2) \rightarrow Prop$

---

[44] $Vec(A, n)$ can be seen as a collection of elements of type A with an explicit *nat* argument counting the elements.

[45] Note that reciprocal predicates can be seen as cases after the functor *each_other* has been applied. In a sense, the semantics of reciprocals are similar to regular transitive predicates after *each_other* has been applied. See the following discussion on *each_other*.

(89) $[\![each\ other]\!]: \Pi A: \mathrm{CN}, \Pi n: Nat.\ (A \to A \to Prop) \to Vec(A, n+2) \to Prop$

(90) $[\![respectively]\!]: \Pi A: \mathrm{CN}.\Pi n: Nat.\ Vec(A, n+2) \to Vec((A \to Prop), n+2) \to Prop$

With the above typings, in order to get expected inferences, we need to assume more information concerning these words. For example, for *each other*, we assume that the following be true: for any $A: \mathrm{CN}, n: nat, P: A \to A \to Prop$ and $v: Vec(A, n+2)$,

(91) $[\![each\ other]\!](A, n, P, v) \supset \forall i, j : nat.\ i \le n+1 \wedge j \le n+2 \wedge i \ne j \supset P(v_i, v_j) \wedge P(v_j, v_i),$

where if $v = (a_1, \ldots, a_{n+2})$ then $v_i = a_i$ and $v_j = v_j$.

The above says that given an $A: CN$, an $n: nat$, a $P: A \to A \to Prop$, the type $Vec((A \to Prop), n+2) \to Prop$ is returned and for any two *nat* arguments i, j that are smaller than or equal to $n+2$, we get both $P(v_i, v_j) \wedge P(v_j, v_i)$. With this, it is straightforward to get the expected inference in (86). Similar lexical entries can be given for *meet* and *respectively*, covering inferences (85) and (87) as well.[46]

Remark 5 A promising aspect of using vector types is that they can potentially be used for the proper semantic treatment of some of the non-classical quantifiers including, for example, *exactly three* or *most*.[47] We do not know how far one can go with vector types as regards a general way of dealing with plurals. We have not yet explored the possibilities as well as the consequences of this proposal with respect to a general theory of plurals. This is a topic which we will pursue in future work. However, one can already see a way to treat cases of negated plurals like the ones shown below:

(92) Just one accountant attended the meeting
Did no accountant(s) attend the meeting? [No]

(93) Just one accountant attended the meeting
Did any accountant(s) attend the meeting? [Yes]

We can assume that plural CNs are in the plural part of CN, CN_{pl}, with $\mathrm{CN}_{pl} < \mathrm{CN}$. Now, we can consider typings of quantifiers with vectors for plural CNs, something along the line of the following type:

(94) $\Pi n: Nat.\Pi A: \mathrm{CN}_{pl}.(Vec([\![human]\!], n) \to Prop) \to Prop$

The above typing works as follows: first it takes two arguments, one of type *nat* and one of type CN_{pl}. Then, this is followed by a predicate of type $(Vec(A, n) \to Prop)$.

---

[46] On the assumption that *meet* and *respectively* are also assumed to involve extra information in the same vein with *each other*.

[47] Also, this does not mean that the Vector-treatments are superior as compared to some existing constructive semantic accounts of these quantifiers [see, for example, Sundholm (1989)].

This would presumably be the typing for a plural predicate, like e.g. *walk*.[48] For example, a quantifier like *three* can have the following type:

(95) $[\![three]\!]: (\Pi A : CN_{pl}.((Vec(A, 3) \rightarrow Prop) \rightarrow Prop)$

Three men will be defined as:

(96) $[\![three\ men]\!]: (((Vec(Man, 3) \rightarrow Prop) \rightarrow Prop)$

Then, given the explicit *nat* argument one can deal with cases like *exactly three* by further specifying that only vectors with $n = 3$ will make the proposition true, and all other vectors of $n < 3$ or $n > 3$ will make the proposition false. This will for example be needed for cases like the following:

(97)   Exactly two people came $\not\Rightarrow$ three people came

Again more work is needed in order to see how using vector types can develop into a viable alternative on dealing with quantifiers. We leave the issue open for the moment.[49]

4.9 Evaluating Against a Subset of the FraCas Test Examples

In this section we have evaluated our approach against a subset of examples from the FraCas test suite. For the needs of this paper, we have used examples from these 4 sections: quantifiers, adjectives, comparatives and conjoined NPs. Overall, the system was evaluated against 77 examples from 4 sections. 72/77 examples were correctly captured (approx. 93.5 %). What is more important, all of the examples were managed to be proven automatically (see the discussion in the next section).

4.9.1 Quantifier Section

For this section, we evaluate against 35 examples, 7 from each §3.1 in the FraCas test suite. We follow the following tactic in choosing the examples: we either take the first 7 examples from each subsection, or if a number of consecutive examples are similar in terms of the way they are proven, some of these are skipped to the next one.[50] We used the modified GF parser as this was designed to deal with the FraCas examples in Ljunglof and Siverbo (2011). For the moment we do not have an automatic translation between a well-formed grammatical input and the syntax of

---

[48] There are a number of details as to how the regular entry for something like walk ($Animal \rightarrow Prop$) and its plural version ($\Pi n : nat, vector\ Animal\ n \rightarrow Prop$) are related but this is something that we cannot discuss here. See however the discussion in Chatzikyriakidis and Luo (2012).

[49] However, look at a first way this can be used in order to deal with inferences involving this kind of quantifiers in Sect. 4.9.1.

[50] For example we have skipped examples 3.68 and 3.69 in the FraCas test suite given the similarity with 3.67.

the proof assistant, so this process cannot be done automatically at the moment. After parsing, we formulate the FraCas examples as theorems. We get the correct results in all examples in this case (35/35). Some notes are at hand as regards the actual translation to the logical language. In some instances, we introduced non-compositional entries, e.g. $right\_to\_live\_in\_Europe : CN$ as a simple token. We did that for reasons of brevity and only in cases these did not affect the outcome of the proof in any way. With this note, let us see some representative examples, starting with the first section:

(98) An Italian became the world's greatest tenor.
       Was there an Italian that became the world's greatest tenor [Yes]

Here we define $world's\_greatest\_tenor$ non-compositionally as CN, but only for reasons of brevity.[51] The example is easily proven, given that the existential requirement of the hypothesis is given by the quantifier *an* in the premise. A further example from the second subsection (3.17 in the FraCas test suite):

(99) An Irishman won the Nobel prize for literature.
       Did an Irishman win a Nobel prize [Yes]

In this example, $for\_literature$ is treated in the same sense as on_time. In particular, it has an identical lexical entry, i.e. it is defined as the first projection of the auxiliary object ADV. This suffices to prove the example. One last example from the same section (3.69 in the FraCas test suite):

(100)   Every resident of the North American continent can travel freely within Europe
            Every Canadian is a resident of a North American continent
            Can every Canadian travel freely within Europe [Yes]

In the above example, we treat $resident\_of\_the\_North\_American\_continent$ non-compositionally. The second premise is encoded as a subtype relation between $Canadian\, resident$ and $resident\_of\_the\_North\_American\_continent$. This suffices to prove the example.

4.9.2 Section on Adjectives

For the adjectival section, we have tested our account against 16 examples spanning across four subsections: §3.5.2–§3.5.5 in the FraCas test suite. In this section, correct results were obtained for 14/16 examples. Two of the examples were predicted to produce *yes* as an answer where the desired result was *do not know*. One of these is shown below:[52]

(101)   All legal authorities are law lecturers
            All law lecturers are legal authorities
            Are all competent legal authorities competent law lecturers? [Don't know]

---

[51] It can be defined compositionally as a Σ type.

[52] This is in fact the only case of those tested where the prover finds a proof where it should not have.

The prover finds a proof for the above, given the two premises. Note that in the cases which are called extensional comparison classes in the FraCas test suite, the account makes the correct predictions, e.g. cases like the following:

(102)   All legal authorities are law lecturers
        All law lecturers are legal authorities
        Are all fat legal authorities fat law lecturers? [Yes]

Other interesting cases involve examples like the following (3.208 in the FraCas test suite):

(103)   Mickey is a small animal
        Dumbo is a large animal
        Is Micky smaller than Dumbo? [Yes]

In order to deal with this example, one has to relate small with its comparative. In effect, what we did is introduce a condition which says that for all elements that are of a bigger size than small (e.g. normalsize or large), the smaller than relation holds between these elements and the small element. This suffices to prove such examples. Similar considerations apply to other comparatives.

All the other examples in the relevant sections can be straightforwardly proven.

4.9.3 Section on Conjoined NPs and Bare Plurals

We evaluate the system against §3.2.1 (conjoined NPs and conjoined N bars) and §3.2.3 (bare plurals), 15 examples in total. The first five examples are similar to (34), and are thus proven in a similar way. More advanced examples include the following:

(104)   Exactly three lawyers and three accountants signed the contract
        Did six lawyers signed the contract?

In order to prove examples like these, we use vector types to define quantifiers. *Exactly three* involves a vector of $n = 3$ and any other vector with $n > 3$ or $n < 3$ will make the proposition false. Thus, $[\![finish]\!](the\,contract)(Vec(Lawyer, 3) \not\Rightarrow [\![finish]\!](the\,contract)(Vec(Lawyer, 6)$.

The system gives a correct answer to 12/15 examples (80 %). Problematic cases include examples like the ones shown below:

(105)   Every representative or client was at the meeting
        Was every representative and client at the meeting?

The semantics provided for coordinators are not able to capture this case and should be revisited. Lastly, we treated bare plurals as involving an existential reading, so

examples with universal readings like the one shown below, were not captured:

(106)   Clients at the demonstration were impressed by the system's performance.
        Smith was a client at the demonstration
        Was Smith impressed by the system's performance? [Yes]

4.9.4 Section on Attitudes

The system was evaluated against §3.9 of the FraCas test suite, a total of 11 examples. All of the examples were correctly dealt with. §3.9.1 involves different kind of attitude verbs, some of them presupposing the truth of their complement (know) and some of them not (e.g. believe). The analysis as proposed in Sect. 4.6 suffices. Also, examples like the one shown below are correctly captured (no proof can be found):

(107)   Smith saw Jones sign the contract
        If Smith signed the contract, his heart was beating
        Did Smith see Jones' heart beating? [Do not know]

   Lastly, existential instantiation and substitution cases like examples (3.343) and (3.344) in the FraCas test suite are also correctly captured. E.g. in the example below, introducing an equality relation between $Jones : Man$ and $the\ chairman\_of\_Itel :$ $Human$, will suffice:

(108)   Smith saw Jones sign the contract
        John is the chairman of ITEL
        Did Smith see the chairman of ITEL sign the contract? [Yes]

5 NLI: Discussion on Different Approaches and Automation

5.1 Informal Comparison with Other Relevant Approaches

The most obvious difference between the system presented here and deep approaches to NLI that use first-order logic as their translation language like e.g. Bos and Markert (2005, 2006), is the use of a many-sorted typed system rather than an untyped one. This, in conjunction with the coercive subtyping mechanism, takes care of a number of inferences via typing only (e.g. monotone on the first argument or adjectival inferences). In systems translating to first-order logic, this information must be added separately as axioms.[53] Furthermore, dependent typing offers a number of welcome results. One such result was developed in this paper and concerns the employment of Σ types not only in dealing with existential[54] or adjectival modification but to interpret

---

[53] For example, the Montagovian meaning postulates for the different kinds of adjectives have to be defined as axioms [see e.g. Pulman (2013)]. In our case, and at least for intersective and subsective adjectives, their inferential properties are derived via typing only [see Chatzikyriakidis and Luo (2013) for more information].

[54] Even though we do not use Σ types to represent existential quantification.

adverbial modification as well as the semantics of factive verbs. Again, the advantage in this case is the ability of using dependent types in order to take care of the desired inferences without resorting to meaning postulates. Abstracting away from the details of each line of approach, like for example the phenomena that are treated in one of the approaches but not in the other,[55] the basic difference seems to boil down to the use of two rather different logical languages in interpreting NL semantics, first-order logic on the one hand and an MTT on the other.

However, and as already mentioned, the system presented in this paper is not yet a full-blown system, given that only the part of the inferential process is shown and not any of the other components of a successful NLI system, namely a wide-coverage parser recognizing grammatical strings of text as well as various components that perform some kind of pre-processing of the goal before the latter is handed to the prover. For example, in Bos and Markert (2006) a wide coverage CCG parser is used, while Background Knowledge is encoded via translating any relations found in WordNet (e.g. hyponymy relations) to first-order logic. The same is done for generic knowledge (e.g. passives, spatial information). Furthermore, deep approaches usually involve a shallow approach component as it is the case for example in Bos and Markert (2005) where some form of relation between the premise and the hypothesis are derived. This is done via searching for word overlaps between the premise and hypothesis by taking into consideration WordNet relations. This process results in the assignment of a similarity measure between the premise and hypothesis.[56] Such hybrid approach will be interesting to use once a more complete version of the system presented here is ready. Another idea we would like to use is that that of entailment approximation discussed in Bos and Markert (2005). The intuition behind it is simple and is based on the informal observation that when the prover has almost found a proof, the relation is usually an entailment. Of course, this is very difficult to formalize in practice for obvious reasons. In Bos and Markert (2005), a model builder is used for this reason. Again the idea is simple: in case we are dealing with an entailment, the entities of the model are the same in both the premise and hypothesis. In case we are not dealing with an entailment, the domain size is different. Domain size is then used to approximate entailment: larger distances in the domain size point to non-entailment, smaller distances to entailment. Thus, a possible future direction is to try and see how can the concept of entailment approximation be translated within our system. Obviously, we will not be using any kind of model builder but however, one can measure the domain size via the number of entities or relations between the entities that are needed in the local context of the proof in order for the premise or the hypothesis to be true. Then the same idea used in Bos and Markert (2005), based on the distance in the domain size of the entities plus relations between the premise and hypothesis, can be used.

---

[55] E.g. treatment of anaphora that is lacking in our account or the treatment of collective predication temporal reference lacking in deep approaches like Bos and Markert (2005, 2006), Pulman (2013).

[56] The account proposed in MacCartney (2009), as already mentioned, is a kind of hybrid approach with both a shallow and a deep component. It is out of the scope of this paper to look at the state-of-the-art shallow approaches to NLI. However, the interested reader is directed to MacCartney (2009) and references therein for more information on this type of approaches.

It is clear from the above that the next step for us will have to be the development of a full-blown NLI system. This will ideally involve the development of a parser using Ranta's Grammatical Framework (GF, Ranta 2011), its purpose being twofold: (a) parse grammatical Engish sentences and (b) linearize this parsed input into the syntax of Coq. In effect, Coq syntax is treated like an ordinary language. The system will involve an abstract syntax but two concrete syntaxes, one for English and one for the Coq language.[57] Additional information like BK or generic knowledge can be expressed via means of axioms or even typing (e.g. hyponymy relations) drawn from WordNet or similar sources (VerbNet, ConceptNet) . The same holds for generic knowledge. These all remain part of our future work and are in our opinion feasible.

Furthermore, and even though we have covered a number of issues in this paper, there are sections in the FraCas test suite that we have not tried yet. For example the section (or subsections) discussing issues relevant to the aspectual system have not yet been properly tried out. It is our intention to attempt a proper formalization as well as implementation of aspect in Coq and extend the preliminary implementation of tense as shown in this paper as well. Similar considerations apply to other sections of the FraCas test suite like e.g. the section dealing with inference in elliptical environments.[58] Lastly, if such a system is to have broader practical applications one needs to test against real text and not examples constructed ad hoc for the sake of testing various categories of inference as the FraCas test suite is basically doing. The next step will thus be to test the proposed account against the RTE challenge suites (Dagan et al. 2006). This, along with what we have mentioned already, consists the basis of what our future work is directed towards.

5.2 Interaction and Automation

Coq is an interactive theorem prover. As such, theorems, in our case NLIs, are proven interactively and not automatically. One may argue, that such a system is not really helpful for NLI, since what we want is a way to prove these inferences automatically. This is a valid point and of course we do agree. However, the idea of using an interactive theorem prover has a number of advantages. One of them is that by using an interactive theorem prover one is able to see the reason a given theorem cannot be proven. This last fact alone can be quite helpful in designing automated tactics for NLI. Furthermore, Coq itself has a number of built-in tactics that are designed to automate trivial parts of proofs. For example, some of our examples can be solved with *intuition* or *jauto* once the *cbv delta* tactic has been executed. *Cbv delta* replaces the occurrences of a defined notion by the definition itself in the current goal (or in any of the hypotheses) while *intuition* just looks for first-order intuitionistic logic tautologies. We can thus define a new tactic which first calls *cbv delta*, followed by *intuition* and *jauto* in order

---

[57] This is one of the core ideas of GF parsers, i.e defining one abstract syntax that corresponds to multiple concrete ones.

[58] Dealing with ellipsis successfully is of course largely dependent on the adequacy of the parser, given that if the parser succeeds in parsing elliptical constructions it will then linearize these structures into the Coq language where the elided information will be present. From this stage on, inferences are easy to be proven. However, this issue is left for future work.

to automate a number of example cases.[59] We have introduced such a tactic (AUTO in the source code) and indeed a number of example cases can be automatically be proven by using this tactic (e.g. 25, 103, 45). For more advanced cases, one can use more elaborate proof-techniques in order to achieve automation. For example in cases where a Σ type analysis is used, like e.g. in the case of VP adverbs, one needs to use destruct specifically for the auxiliary objects (e.g. *ADV* for ontime). One can thus devise an automatic tactic which is however context dependent, depending on the example. In the same case one might need to instruct Coq to apply a specific premise. One tactic that does both the aforementioned is shown below:

(109)   Ltac AUTO1 a b:= cbv; destruct a; eapply b; AUTO.

The above tactic can take care of the Σ type cases (note the use of AUTO within AUTO1). A similar more advanced automated tactic has been defined for cases of collective predication. These three tactics are then all we need to automate all our proof-examples. In order to achieve full automation, we can further use one composite tactic which tries one of the three tactics and succeeds in case one of them does. Assuming we have three tactics *a*, *b* and *c*, one can define the following tactic, say *d*:

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

Using this technique, one can actually automate all the examples discussed in this paper. Most of the cases can receive total automation while some of them, even though automatically proven, will need some extra guidance to the prover, for example instructing the prover to apply part of the automated tactic to specific elements. For example in the case of auxiliary objects, one has to instruct the prover to destruct these objects. Thus instead of just typing AUTO, one will have to type something like AUTO d, where d is the specific item we want to unfold. Currently, we are looking for ways to eliminate this as well, so automation does not need this kind of user aid in all cases. It will be very interesting to see how far one can go with automation, in particular whether automation is still possible when the examples are comprised of bigger texts, like e.g. some examples from the RTE challenges. In fact, proof-automation in Coq is an on-going research topic within the community and a number of researchers have provided interesting results like for example work by Wilson et al. (2010) on inductive proof-automation. Further work is needed on the feasibility of automation as regards NLI but the first results seem promising. We hope that this paper will be the start of a new research direction, in which MTT semantics (or in general formal semantics) and proof assistant technology work on a par in order to deal with NL reasoning.

6 Conclusion

In this paper we have presented the first attempt to use proof assistant technology in order to deal with NLI. Furthermore, this paper proposed the use of MTTs as the logical language for dealing with NLI. We provided an account of a number of cases

---

[59] *Jauto* is part of the *LibTactics* library, containing extra tactics beyond the standard ones.

from the FraCas test suite using Luo's TT with coercive subtyping. It was shown that using a considerably richer language than first-order logic, can give us a number of welcome results as regards NLI. In particular, the coercive subtyping mechanism as well as the use of dependent typing have been shown to be very helpful in dealing with various NLI cases. The account was then tested in Coq where the FraCas test suite examples were encoded as Coq theorems. Lastly, it was shown that one cannot only use Coq in order to reason about NL semantics, but to further automate the proof process by developing used-defined tactics.

Appendix: Coq Code of Examples

All of the examples in this paper have been tried in the Coq proof assistant. The source codes can be obtained by sending an email request to stergios.chatzikyriakidis@cs.rhul.ac.uk. Here, we shall give an example with Coq tactics (Appendix "A More Advanced Example: Proving Peirce's Law") and some examples in linguistic semantics (Appendix "An Example from the FraCas Test Suite").

A More Advanced Example: Proving Peirce's Law

We want to prove that if the law of the excluded middle holds then so does Peirce's law.

```
Definition lem:= A \/ ~ A.
Definition Peirce:= ((A->B)->A)->A.
Theorem lemP: lem -> Peirce.
unfold Peirce. unfold LEM.unfold Peirce. intros. elim H.intros.
assumption.intros.apply H0.intros.absurd A.assumption. assumption.
```

We unfold the definitions, apply *intros* and *elim H*:

```
lemP < elim H.
2 subgoals
H : A \/ ~ A
H0 : (A -> B) -> A
==========================
A -> A
subgoal 2 is:
~ A -> A
```

Then, *intro*, *assumption* and *intro* again:

```
lemP < intros.
1 subgoal
```

```
H : A \/ ~ A
H0 : (A -> B) -> A
H1 : ~ A
============================
A
```

We use *apply H0* and now we have to prove $A \to B$. We apply *intro*:

```
lemP < intro.
1 subgoal
H : A \/ ~ A
H0 : (A -> B) -> A
H1 : ~ A
H2 : A
============================
B
```

We use *absurd A* and now we need to prove A and A, which can be done via two applications of *assumption*.

The above can be proved automatically as well, using automated user-defined tactics. For this case, we can define a tactic which unfolds all the definitions and then applies *tauto*, which tries intuitionistic propositional tautologies:

```
Ltac AUTO:= cbv delta;tauto
```

This suffices to prove our example automatically.

An Example from the FraCas Test Suite

FraCas example 3.55

(111)   Some Irish delegates finished the report on time.
         Did any delegate finish the report on time [Yes]

```
Parameter delegate report: CN
Record Irishdelegate : CN := mkIrishdelegate { c :> delegate; _ : Irish c }.
Parameter on_time: forall A:CN, (A -> Prop) -> (A->Prop).
Parameter finish: Object -> Human -> Prop.
Axiom so:survey->Object. Coercion so: survey>->Object. *subtyping*
Axiom dh:delegate->Human. Coercion dh: delegate>->Human. *subtyping*
Theorem IRISH: (some Irishdelegate)(On_time(finish(the report)))->(some delegate)
(On_time (finish(the report))).
```

We unfold the definitions for *a* and move the premise to the assumptions via intro and we apply the elimination tactic *elim*:

```
IRISH < elim H.
1 subgoal

H : exists x : Irishdelegate, On_time (finish (the report)) x
============================
forall x : Irishdelegate,
On_time (finish (the report)) x ->
exists x0 : delegate, On_time (finish (the report)) x0
```

We apply *intros*:

```
IRISH < intro.
1 subgoal
H : exists x : Irishdelegate, On_time (finish (the report)) x
x : Irishdelegate
H0 : On_time (finish (the report)) x
============================
exists x0 : delegate, On_time (finish (the report)) x0
```

With $x : Irishdelegate$ as an assumption, we can now substitute x_0 in the conclusion with x thanks to the subtyping mechanism:

```
IRISH < exists x.
1 subgoal
H : exists x : Irishdelegate, On_time (finish (the report)) x
x : Irishdelegate
H0 : On_time (finish (the report)) x
============================
On_time (finish (the report)) x
```

We apply assumption and the *proof* is over. The above can be proved using automated tactics as well. For the purposes of this paper the following tactic has been defined:

```
Ltac AUTO:= cbv delta;intuition;try repeat congruence;  jauto;intuition.
```

The above unfolds all definitions, then tries all intuitionistic first-order tautologies (*intuition*). Then, *congruence* deals with any equalities (for the example in question there are no equalities). Then *jauto* is applied, which is basically Coq's predefined *auto* tactic along with some pre-processing of the goal. Then again *intuition* is applied. This automated tactic can prove what we want (and much more).

References

Asher, N. (2011). *Lexical meaning in context: A web of words*. Cambridge: Cambridge University Press.

Asher, N., & Luo, Z. (2012). Formalisation of coercions in lexical semantics. *Sinn und Bedeutung 17*, Paris 223.

Bassac, C., Mery, M., & Retoré, C. (2010). Towards a type-theoretical account of lexical semantics. *Journal of Logic Language and Information*, *19*, 229–245.

Blackburn, P., & Bos, J. (2005). *Representation and inference for natural language*. Stanford: CSLI Publications.

Blazy, S., Dargaye, Z., & Leroy, X. (2006) Formal verification of a C compiler front-end. In *FM 2006: International symposium on formal methods. Lecture Notes in Computer Science* (Vol. 4085, pp. 460–475). Berlin: Springer. http://gallium.inria.fr/~xleroy/publi/cfront.pdf.

Bos, J., & Markert, K. (2005). Recognising textual entailment with logical inference. In *Proceedings of the 2005 conference on empirical methods in natural language processing (EMNLP)* (pp. 98–103).

Bos, J., & Markert, K. (2006). When logical inference helps determining textual entailment (and when it doesn't). In *Proceedings of the 2nd PASCAL challenges workshop on recognising textual entailment*.

Callaghan, P., & Luo, Z. (2001). An implementation of LF with coercive subtyping and universes. *Journal of Automated Reasoning*, *27*(1), 3–27.

Chatzikyriakidis, S. (2014). Adverbs in a modern type theory. In N. Asherl & S. Soloviev (Eds.), *Proceedings of LACL2014. LNCS 8535* (pp. 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.), *Proceedings of constraint solving and language processing (CSLP12)* (pp. 31–51). LNCS 8114, Orleans.

Chatzikyriakidis, S., & Luo, Z. (2013). Adjectives in a modern type-theoretical setting. In G. Morrill & J. Nederhof (Eds.), *Proceedings of formal grammar 2013* (pp. 159–174). LNCS 8036.

Church, A. (1940). A formulation of the simple theory of types. *The Journal of Symbolic Logic*, *5*(1), 56–68.

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/.

Dagan, I., Glickman, D., & Magnini, B. (2006). The PASCAL recognising textual entailment challenge. In J. Quionero-Candela, I. Dagan, B. Magnini & F. d'Alch-Buc (Eds.), *Machine learning challenges* (pp. 177–190). LNCS 3944.

Davidson, D. (1967). Compositionality and coercion in semantics: The semantics of adjective meaning. In N. Rescher (Ed.), *The logical form of action sentences* (pp. 81–95). Pittsburgh: University of Pittsburgh Press.

Davidson, D. (1967). The logical form of action sentences. In N. Rescher (Ed.), *The logic of decision and action*. Pittsburgh: University of Pittsburgh Press.

Fox, C., & Lappin, S. (1990). *Foundations of intensional semantics*. Oxford: Oxford University Press.

Girard, J. Y. (1971). Une extension de l'interpretation fonctionelle de Gödel à l'analyse et son application à l'élimination des coupures dans et la théorie des types'. In *Proceedings of 2nd Scandinavian logic symposium*, North-Holland.

Goguen, H. (1994). *A typed operational semantics for type theory*. Ph.D. thesis, University of Edinburgh.

Gonthier, G. (2005). *A computer-checked proof of the four colour theorem*. http://research.microsoft.com/~gonthier/4colproof.pdf.

Grudzinska, J., & Zawadowski, M. (2014). System with generalized quantifiers on dependent types for anaphora. In *Proceedings of EACL 2014*.

Kamp, H. (1975). Formal semantics of natural language. In E. Keenan (Ed.), *Two theories about adjectives* (pp. 123–155). Cambridge: Cambridge University Press.

Lappin, S. (To appear). Curry typing, polymorphism, and fine-grained intensionality. In S. Lappin & C. Fox (Eds.), *Handbook of contemporary semantic theory*. Oxford: Blackwell.

Ljunglof, P., & Siverbo, M. (2011). *A bilingual treebank for the FraCas test suite*. Clt project report, University of Gothenburg.

Luo, Z. (1994). *Computation and reasoning: A type theory for computer science*. Oxford: Oxford University Press.

Luo, Z. (1999). Coercive subtyping. *Journal of Logic and Computation*, *9*(1), 105–130.

Luo, Z. (2010). Type-theoretical semantics with coercive subtyping. *Semantics and Linguistic Theory 20 (SALT20), Vancouver*, *84*(2), 28–56.

Luo, Z. (2011). Contextual analysis of word meanings in type-theoretical semantics. In *Logical aspects of computational linguistics* (LACL'2011) (pp. 159–174). LNAI 6736.

Luo, Z. (2012). Common nouns as types. In D. Bechet & A. Dikovsky (Eds.), *Logical aspects of computational linguistics* (LACL'2012) (pp. 173–185). LNCS 7351.

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

Luo, Z., Soloviev, S., & Xue, T. (2012). Coercive subtyping: Theory and implementation. *Information and Computation*, *223*, 18–42.

MacCartney, B. (2009). *Natural language inference*. Ph.D. thesis, Stanford Universisty.

Maienborn, C., & Schafer, M. (2011). Adverbs and adverbials. In C. Maienborn, K. von Heusinger, & P. Portner (Eds.), *Semantics: An international handbook of natural language meaning* (pp. 1390–1420). Mouton: De Gruyter.

Martin-Löf, P. (1975). An intuitionistic theory of types: predicative part. In H. Rose & J. C. Shepherdson (Eds.), *Logic Colloquium'73*.

Martin-Löf, P. (1984). *Intuitionistic type theory*. Naples: Bibliopolis.

Montague, R. (1973). The proper treatment of quantification in ordinary English. In J. Hintikka, J. Moravcsik & P. Suppes (Eds.), *Approaches to natural languages* (pp. 221–242).

Montague, R. (1974). *Formal philosophy*. New Haven: Yale University Press.

Muskens, R. (2005). Sense and the computation of reference. *Linguistics and Philosophy*, *28*(4), 473–504.

Partee, B. (2007). Compositionality and coercion in semantics: The semantics of adjective meaning. In G. Bouma, I. Krämer & J. Zwarts (Eds.), *Cognitive foundations of interpretation* (pp. 145–161). Royal Netherlands Academy of Arts and Sciences.

Partee, B. (2010). Privative adjectives: Subsective plus coercion. In R. Bauerle & U. Reyle (Eds.), *Presuppositions and discourse: Essays offered to Hans Kamp* (pp. 123–155). Bingley: Emerald Group Publishing.

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*. Cambridge: MIT.

Ranta, A. (1994). *Type-theoretical grammar*. Oxford: Oxford University Press.

Ranta, A. (2011). *Grammatical framework: Programming with multilingual grammar*. Stanford: CSLI Publications.

Sundholm, G. (1986). Proof theory and meaning. In D. Gabbay & F. Guenthner (Eds.), *Handbook of philosophical logic III: Alternatives to classical logic* (pp. 471–506). Reidel.

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

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

Wilson, S., Fleuriot, A., & Smaill, A. (2010). Inductive proof automation for coq. In *Proceedings of the 2nd Coq Workshop*. EPTCS.

Xue, T., & Luo, Z. (2012). Dot-types and their implementation. In *Logical aspects of computational linguistics* (LACL 2012). LNCS 7351.