Natural Language Reasoning Using proof-assistant technology: Rich Typing and beyond*

Stergios Chatzikyriakidis
Dept of Computer Science,
Royal Holloway, Univ of London
Egham, Surrey TW20 0EX, U.K;
stergios.chatzikyriakidis@cs.rhul.ac.uk

Zhaohui Luo
Dept of Computer Science,
Royal Holloway, Univ of London
Egham, Surrey TW20 0EX, U.K;
zhaohui@cs.rhul.ac.uk

Abstract

In this paper, we study natural language inference based on the formal semantics in modern type theories (MTTs) and their implementations in proof-assistants such as Coq. To this end, the type theory UTT with coercive subtyping is used as the logical language in which natural language semantics is translated to, followed by the implementation of these semantics in the Coq proof-assistant. Valid inferences are treated as theorems to be proven via Coq's proof machinery. We shall emphasise that the rich typing mechanisms in MTTs (much richer than those in the simple type theory as used in the Montagovian setting) provide very useful tools in many respects in formal semantics. This is exemplified via the formalisation of various linguistic examples, including conjoined NPs, comparatives, adjectives as well as various linguistic coercions. The aim of the paper is thus twofold: a) to show that the use of proof-assistant technology has indeed the potential to be developed into a new way of dealing with inference, and b) to exemplify the advantages of having a rich typing system to the study of formal semantics in general and natural language inference in particular.

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 MTTs with coercive subtyping (Luo, 2012b) and its implementation in the proof assistant Coq (Coq, 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. The use of MTTs for NL semantics has been proposed with exciting results as regards various issues of NL semantics, ranging from quantification and anaphora to adjectival modification, co-predication, belief and context formalization. (Sundholm, 1989; Ranta, 1994; Boldini, 2000; Cooper, 2005; Fox and Lappin, 2005; Retoré, 2013; Ginzburg and Cooper, forthcoming; Luo, 2011a; Luo, 2012b; Chatzikyriakidis and Luo, 2012; Chatzikyriakidis and Luo, 2013a). Recently, there has been a systematic study of MTT semantics using Luo's UTT with coercive subtyping (type theory with coercive subtyping, henceforth TTCS) (Luo, 2010; Luo, 2011a; Luo, 2012b; Chatzikyriakidis and Luo, 2012; Chatzikyriakidis and Luo, 2013a; Chatzikyriakidis and Luo, 2013b; Chatzikyriakidis and Luo, 2014). This is the version of MTT used in this paper. More specifically, the paper concentrates on one of the key differences between MTTs and simple typed ones, i.e. rich typing. Rich typing will be shown to be a key ingredient for both formal semantics in general and the study of NLI in particular.

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 (Coq, 2007) implements pCIC, the predicative Calculus of Inductive Constructions[1] and sup-

*This work is supported by the research grant F/07-537/AJ of the Leverhulme Trust in the U.K.

[1] pCIC is a type theory that is rather similar to 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 PhD the-

ports 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 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, two different goals are to be achieved: a) on a more practical level, to show how proof-assistant technology can be used in order to deal with NLI and b) on a theoretical level, the significance of rich typing for formal semantics and NLI in particular. These two different aspects of the paper will be studied on a par, by concentrating on a number of NLI cases (quite a lot actually) that are adequately dealt with on a theoretical level via rich typing and the implementation of the account making use of rich type structures in Coq on a more practical level. We shall also consider how to employ dependent typing in the coercive subtyping framework to formalise linguistic coercions.

2 Rich typing in MTTs

A Modern Type Theory (MTT) is a variant of a class of type theories in the tradition initiated by the work of Martin-Löf (Martin-Löf, 1975; Martin-Löf, 1984), which have dependent and inductive types, among others. We choose to call them Modern Type Theories in order to distinguish them from Church's simple type theory (Church, 1940) 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.[2] This stands as part of the study of linguistic semantics using MTTs rather than simply typed ones. In particular, in this paper we discuss a number of key issues as regards the typing system, which will be shown to allow more fine-grained distinctions and expressivity compared to classical simple typed systems as these are used in mainstream Montagovian semantics.

2.1 Type many-sortedness and CNs as types

In Montague semantics (Montague, 1974), 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 $John\colon [\![man]\!]$ and $Mary\colon [\![woman]\!]$, where $[\![man]\!]$ and $[\![woman]\!]$ are different types.

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 $e \rightarrow t$) 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. The interpretation of CNs as Types is also a prerequisite in order for the subtyping mechanism to work. This is because, assuming CNs to be predicates, subtyping would go wrong given contravariance of function types.[3]

2.2 Subtyping

Coercive subtyping (Luo, 1999; Luo et al., 2012) provides an adequate framework to be employed for MTT-based formal semantics (Luo, 2010; Luo, 2012b).[4] It can be seen as an abbreviation mechanism: A is a (proper) subtype of B ($A < B$) if

sis (Goguen, 1994) as regards the meta-theory of UTT.

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

[3]See (Chatzikyriakidis and Luo, 2013b) for more information. See also (Luo, 2012a) for further philosophical argumentation on the choosing to represent CNs as types.

[4]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).

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

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

In case that $[\![man]\!]$ is defined as a composite Σ-type (see §2.3 below for details), where $male\colon [\![human]\!] \to Prop$:

(2) $[\![man]\!] = \Sigma h\colon [\![human]\!].\, male(h)$

we have that (1) is the case because the above Σ-type is a subtype of $[\![human]\!]$ via the first projection π_1:

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

We will see in the next section the importance of the coercive subtyping mechanism when dealing with NLI.

2.3 Dependent typing and universes

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 $male\colon [\![human]\!] \to Prop$, then the Σ-type $\Sigma h\colon [\![human]\!].\, male(h)$ 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). For example, $handsome\ man$ is interpreted as a Σ-type (4), the type of handsome men (or more precisely, of those men together with proofs that they are handsome):

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

where $[\![handsome]\!](m)$ is a family of propositions/types that depends on the man m.[5]

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 $A \to B$ 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 (5) is interpreted as (6):

(5) Every man walks.

(6) $\Pi x\colon [\![man]\!].[\![walk]\!](x)$

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).[6] For example, one may want to collect all the names of the types that interpret common nouns into a universe $\mathrm{CN}\colon Type$. The idea is that for each type A that interprets a common noun, there is a name $\overline{A}$ in CN. For example,

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

[5] Adjectival modification is a notoriously difficult issue and as such not all cases of adjectives can be captured via using a Σ type analysis. For a proper treatment of adjectival modification within this framework, see (Chatzikyriakidis and Luo, 2013a).

[6] 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).

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

(7) $[\![man]\!]$, $[\![woman]\!]$, $[\![book]\!]$, ...

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

(9) $G_R + G_F$

where the Σ-type in (8 is the proposed interpretation of 'handsome man' and the disjoint sum type in (9) is that of 'gun' (the sum of real guns and fake guns – see above).[7] Interesting applications of the use of universes can be proposed like for example, their use in giving the types for quantifiers and VP adverbs as extending over the universe CN (Luo, 2011b) as well as coordination extending over the universe of all linguistic types LType (Chatzikyriakidis and Luo, 2012).

3 NL Inference in Coq

Coq is a dependently typed interactive theorem prover implementing the calculus of Inductive Constructions (pCiC, see (Coq, 2007)). Coq, and in general proof-assistants, provide assistance in the development of formal proofs. The idea is simple: you use Coq in order to see whether statements as regards anything that has been either predefined or user-defined (definitions, parameters, variables) can be proven or not. In the case of NLI, the same idea applies: once the semantics of NL words are defined, then these semantics can be reasoned about by using Coq's proof mechanism. In this sense, valid NLIs can be seen as theorems, or better valid NLIs must be theorems.

A very simple case of semantic entailment, that of example (10), will therefore be formulated as the following theorem in Coq (11):

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

(11) Theorem x: John walks $\rightarrow$ some man walks

Now, 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 (11) 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.

3.1 The FraCas test suite

In this section we present how implementing MTT NL semantics in Coq can deal with various cases of NLI inference. For this reason, we use examples from the FraCas test suite. 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,

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

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 what follows, we show how the use of a rich type system can deal with NLI adequately (at least for the cases looked at) from both a theoretical and an implementational point of view.

3.2 Rich typing and NLI

3.2.1 Quantifiers

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

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

Treating adjectival modification as involving a Σ type where the first projection is always a coercion as in (Luo, 2011a), we get *Irish delegate* to be a subtype of *delegate*, i.e. $[\![Irishdelegate]\!] <$

[7] The use of disjoint sum types was proposed by (Chatzikyriakidis and Luo, 2013a) in order to deal with privative modification. The interested reader is directed there for details.

$[\![delegate]\!]$. This is basically all that Coq needs in order to prove the inference.[8]

Moving on to quantifier cases involving monotonicity on the second argument, we notice that these are more difficult to get since an adjunct (e.g. a PP) is involved in deriving the inference:

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

The type proposed for VP adverbs by Luo (Luo, 2011b) is based on the idea of a type universe of CNs. As already said in the introduction, type universes a universe is a collection of (the names of) types put into a type. In this respect, one can form the universe CN which basically stands for the collection of names interpreting common nouns. The type proposed for VP adverbs makes use of this CN universe and assumes quantification over it (Chatzikyriakidis and Luo, 2013a; Chatzikyriakidis and Luo, 2012):

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

However, in order to derive the inference needed in cases of monotonicity on the second argument cases, this typing alone is not enough. Σ types can be used in order to slightly modify the typing. In order to do this, we first introduce an auxiliary object ADV as follows:

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

This reads as follows: for any common noun A and any predicate v over A, $ADV(A, v)$ is a pair (p, m) such that for any $x\colon A$, $p(x)$ implies $v(x)$. Taking the sentence (14) as an example, for the CN *delegate* and predicate $[\![finish]\!]$[9], we define on time to be the first projection of the auxiliary object (16) which is of type (15):

(17) $on\ time = \lambda A : \text{CN}.\lambda v : A \to Prop.$
$\pi_1(ONTIME(A, v))$

As a consequence, for instance, any delegate who finished the survey on time ($p(x)$) in (16) did finish the survey ($v(x)$).

[8]For details on the semantics of the other lexical items like e.g. VP adverbs in the sentence, see the following discussion. Also, following Luo (Luo, 2011a) we implement Σ-types as dependent record types in Coq. Again, see (Chatzikyriakidis and Luo, 2013b) for details.

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

3.2.2 Conjoined NPs

Inference involving conjoined NPs concerns cases like the one shown below:

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

In (Chatzikyriakidis and Luo, 2012), a polymorphic type for binary coordinators that extends over the constructed universe $LType$, the universe of linguistic types was proposed. This can be extended to n-ary coordinators. For example, the coordinator *and* may take three arguments, as in the premise of (18). In such cases, the type of the coordinator, denoted as and_3 in semantics, is:

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

Intuitively, we may write this type as $\Pi A\colon LType.A^3 \to A$. For instance, the semantics of (18) is (20), where c is 'the contract':

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

In order to consider such coordinators in reasoning, we consider the following auxiliary object (similarly to the auxiliary object ADV) and define and_3 as follows:

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

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

Having defined the coordinators such as *and* in such a way, we can get the desired inferences. For example, from the semantics (20), we can infer that 'Jones signed the contract', the hypothesis in (18).[10] Coordinators such as *or* can be defined in a similar way.

3.2.3 Comparatives

Inference with comparatives can also be treated by using Σ types. Two ways of doing this will be proposed, one not involving and one involving measures. We shall consider *shorter than* as a typical example. Intuitively, *shorter than* should be

[10]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 building of new universes). Instead, we shall use an existing universe in Coq in conducting our examples for coordination.

of type $Human \rightarrow Human \rightarrow Prop$ as in the following example:

(23) Mary is shorter than John.

We assume that there be a predicate $short\colon Human \rightarrow Prop$, 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 $COMP$ of the following Σ-type and define *shorter than* to be its first projection:

(24) $COMP\colon \Sigma p\colon Human \rightarrow Human \rightarrow Prop.\forall h_1, h_2, h_3 : Human.$ $p(h_1, h_2) \wedge p(h_2, h_3) \supset p(h_1, h_3) \wedge \forall h_1, h_2 : Human.p(h_1, h_2) \supset short(h_2) \supset short(h_1)$.

(25) $[\![shorter\_than]\!] = \pi_1(COMP)$

With the above, we can easily show that the inferences like (26) can be obtained as expected.[11]

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

Given the definition in $COMP$ according to which if two elements stand in a $COMP$ relation (meaning that the first argument is shorter than the second one), and there is also a third element standing in a $COMP$ relation with the second, then by transitivity defined in $COMP$, this third element also stands in a $COMP$ relation with the first, i.e. the third element is shorter than the first.

3.2.4 Factive/Implicative verbs

This section concerns inference cases with various types of verbs that presuppose the truth of their complement like for example factive or implicative verbs. Example (27) is an example of such a verb, while (28) is not:

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

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

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

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

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

In effect, a similar reasoning to the one used in dealing with VP adverbs is proposed. In effect, an auxiliary object is firstly used, followed by the definition of know as the first projection of the Σ type involved in the auxiliary object. With this, the inference (27) can be obtained as expected. Intensional verbs like *believe* on the other hand do not imply their arguments and inferences like (28) cannot be shown to be valid inferences.

3.2.5 Adjectival inference

As a last example of the use of rich typing in order to deal with NLI, we discuss NLI cases involving adjectives. In (Chatzikyriakidis and Luo, 2013a) we have shown that the use of subtyping, Σ types and universes can give us a correct account of at least intersective and subsective adjectives. Note that the original Σ type analysis proposed by researchers like Ranta (Ranta, 1994) is inadequate to capture the inferential properties of either intersective or subsective adjectives. The FraCas test suite has a rather different classification. One major distinction is between affirmative and non-affirmative adjectives shown below:

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

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

Concentrating on affirmative adjectives for the moment, we see that a Σ type analysis is enough in these cases.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.

Cases of subsective adjectives are discussed in the section dubbed as *extensional comparison*

[11] In giving a full analysis of compratives, one may further consider measures. Such an account is also possible using Σ types, in effect extending the account just proposed for comparatives. The idea is basically to extend the above account using dependent typing over measures. Such an account can be found in (Chatzikyriakidis and Luo, 2013b)

classes in the FraCas test suite. There, cases of adjectival inference involving adjectives like *small* and *large* are discussed. Cases like these can be handled using a typing which quantifies over a universe. In the case of *large* and *small* this universe is the universe CN:[12]

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

With this typing, cases like the one shown below are correctly treated:

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

Lastly, one should be able to take care of inferences associated with intersective adjectives like the one shown below:

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

A concrete example would be *black man* implying *black human*. Given that coercions according to Luo's MTT propagate via the various type constructors, we have: $\Sigma(\llbracket man \rrbracket, black) < \Sigma(\llbracket human \rrbracket, black)$. [13]

4 Linguistic Coercions in MTTs with Coercive Subtyping

Besides being crucial for MTT-semantics, coercive subtyping (Luo, 1999; Luo et al., 2012) also provides us a framework to interpret various linguistic coercions (Asher and Luo, 2012). Besides explaining the basic mechanisms, we shall also show (in §4.3) that dependent types have interesting applications in dealing with situations with sophisticated coercions in MTT-semantics.

4.1 Basic coercions

The basic coercive subtyping mechanism that coerces $f(a)$ into $f(c(a))$ by inserting the coercion c into a gap between f and a, suffices to represent many linguistic coercions. For example, consider

(36) Julie enjoyed a book.

The MTT-semantics of (36) is (37):

(37) $\exists x\colon \llbracket book \rrbracket.\ \llbracket enjoy \rrbracket(j, x)$

where

(38) $\llbracket enjoy \rrbracket\colon Human \to Event \to Prop.$

However, the domain type of $\llbracket enjoy \rrbracket(j)$ is $Event$, which is different from $Book$! Then, how can $\llbracket enjoy \rrbracket(j, x)$ in (37) be well-typed? The answer is that, in the framework of coercive subtyping and, in particular, under the assumption of the following coercion:

(39) $Book <_{reading} Event$

$\llbracket enjoy \rrbracket(j, x)$ is coerced into (and, formally, equal to) $\llbracket enjoy \rrbracket(j, reading(x))$ and hence well-typed. Informally, the sentence (36) is coerced into (40):

(40) Julie enjoyed reading a book.

Note that, in the above, we have considered only one possible coercion (39): from 'enjoy a book' to 'enjoy reading a book'. As we noted in the previous section, however, there are in fact context-dependent 'multiple coercions': e.g., (36) could have meant 'Julie enjoyed writing a book'; there could also be several reading events of that book. Coercive subtyping requires contextual uniqueness of coercions[14], we must restrict the scope/context using *local coercions* (Luo, 2011a).

4.2 Local Coercions

In many situations, it is necessary to limit the scope of a coercion. (36) furnishes an example: with the formal coercion (39), (37) is the correct interpretation of (36). However, there may be several possible coercions and hence (36) may have several meanings: which one to use can only be decided contextually. But note that coherence in coercive subtyping (contextual uniqueness of coercions) is necessary for formal semantics to deal with ambiguity. In such situations, we use local coercions to limit the scope of applicability of coercions. For instance, if (36) is used to mean (40) or 'Julie enjoyed writing a book', we exploit the following two coercions for (36):

(41) **coercion** $Book <_{reading} Event$ **in** (37)

[12] Other more restricted universes will be needed for adjectives like *skilful* given that we may want to avoid constructions like *skilful table*. Universe subtyping can take care of these issues. In effect, one can introduce a subuniverse of CN containing the names of the types $\llbracket human \rrbracket$ and its subtypes only. Let us call this universe CN_H, which is a subtype of CN: $\text{CN}_H <$ CN. Now *skillful* extends over this more restricted universe. See (Chatzikyriakidis and Luo, 2013a) for more detalis.

[13] Cases of non-committal and privative adjectives will not be dealt with in this paper for reasons of space. The interested reader is directed to (Chatzikyriakidis and Luo, 2013a) for a treatment of these types of adjectives within the MTT setting discussed in this paper.

[14] This refers to the notion of *coherence*, the requirement that any two coercions between the same two types (in the same context) be the same. See (Luo, 1999; Luo et al., 2012) for its formal definition.

(42) **coercion** $Book <_{writing} Event$ **in** (37)

Note that such interpretations involve different local coercions and can be used in the same context. There is no ambiguity or confusion as to which coercion is to be employed, but we must make clear the scope of each one of the coercions, over what terms they are operative.

Local coercions have a dual notion – coercion contexts, which are contexts (in type theory) which may contain coercion entries of the form $A <_c B$ as well as entries of the usual form $x\colon A$. Coercion contexts occur left to the $\vdash$-sign. One can move a coercion entry in a coercion context to the right-hand side of the $\vdash$-sign to form a local coercion, while the inversion of this moves the coercion in a local coercion to the left. These constructs are governed by the relevant inference rules, some of which are discussed in, for example, (Luo, 2011a).

4.3 Dependent Types in Coercion Semantics

Sometimes, a simple scoping restriction is not enough. For example, consider

(43) Jill just started *War and Peace*, which Tolstoy finished after many years of hard work. But that won't last because she never gets through long novels.

It is not difficult to see that in (43) the scopes of the reading and writing coercions overlap intertwiningly, and so restrictions on the scopes of coercions will not be sufficient here to ensure uniqueness to eliminate ambiguity.

In many such cases, dependent typing proves to be useful. Indeed, this is the first time in the literature, as far as we know, that dependent types have been shown to be useful directly in the formal semantics of linguistic coercions.

For example, for the above sentences (43), instead of $Event$, we may consider the family of types

$$Evt\colon Human \to Type;$$

intuitively, for any $h\colon Human$, the dependent type $Evt(h)$ is the type of events conducted by h. Now, we can assume that the verbs $start$, $finish$ and $last$ have type $\Pi h\colon Human.\ (Evt(h) \to Prop)$ and $read$ and $write$ have type $\Pi h\colon Human.\ (Book \to Evt(h))$. Furthermore, we can consider the following parameterised coercions, for any $h\colon Human$,

$$Book <_{c(h)} Evt(h),$$

where the coercion $c(h)$ is the function from $Book$ to $Evt(h)$ defined as follows: for any $b\colon Book$,

$$c(h,b) = \begin{cases} write(h,b) & \text{if } h \text{ wrote } b, \\ read(h,b) & \text{otherwise.} \end{cases}$$

where we have simplified the second case by assuming that one would read a book if he/she has not written it. (One may think of other actions to consider more subcases here.) Having the above, we can now interpret (43) as follows (in a simplified form):

(44) $start(j, wp)$
$\&\ finish(t, wp)$
$\&\ \neg last(j, wp)$
$\&\ \forall lb\colon LBook.finish(j, \pi_1(lb))$

where $LBook \equiv \Sigma b : Book.long(b)$ is the type that interprets the CN 'long book' and π_1 is the first projection operator that takes a long book and returns the book itself. In the coercive subtyping framework, (44) is coerced into (and equal to) the following:

(45) $start(j, c(j, wp))$
$\&\ finish(t, c(t, wp))$
$\&\ \neg last(j, c(j, wp))$
$\&\ \forall lb\colon LBook.\ finish(j, c(j, \pi_1(lb)))$

which is (equal to)

(46) $start(j, read(j, wp))$
$\&\ finish(t, write(t, wp))$
$\&\ \neg last(j, read(j, wp))$
$\&\ \forall lb\colon LBook.\ finish(j, c(j, \pi_1(lb)))$

Note that, in the last conjunct, the coercion c is still present – $c(j, \pi_1(lb))$ cannot be reduced furthermore because lb is a variable.

5 Conclusions

In this paper we proposed to deal with NLI by making use of proof-assistant technology, in particular the proof-assistant Coq. It was shown that the combination of MTT semantics as well as the use of a proof-assistant that 'understands' so to say MTT semantics can provide us with encouraging results as regards the computational treatment of NLI. More specifically, the paper has concentrated on the importance and expressivity of MTTs as regards typing by exemplifying the use of a rich typing system in order to deal with a number of inference cases ranging from adjectival and adverbial

modification to conjoined/disjoined NPs, comparatives as well as factive/implicative verbs and type coercions.

References

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

P. Boldini. 2000. Formalizing context in intuitionistic type theory. *Fundamenta Informaticae*, 42(2):1–23.

S. Chatzikyriakidis and Z. Luo. 2012. An account of natural language coordination in type theory with coercive subtyping. In Y. Parmentier and D. Duchier, editors, *Proc. of Constraint Solving and Language Processing (CSLP12). LNCS 8114*, pages 31–51, Orleans.

S. Chatzikyriakidis and Z. Luo. 2013a. Adjectives in a modern type-theoretical setting. In G. Morrill and J.M Nederhof, editors, *Proceedings of Formal Grammar 2013. LNCS 8036*, pages 159–174.

S. Chatzikyriakidis and Z. Luo. 2013b. Natural language inference in coq. Submitted.

S. Chatzikyriakidis and Z. Luo. 2014. Hyperintensionality in modern type theories. Submitted manuscript.

A. Church. 1940. A formulation of the simple theory of types. *J. Symbolic Logic*, 5(1).

R. Cooper, D. Crouch, J. van Eijck, C. Fox, J. van Genabith, J. Jaspars, H. Kamp, D. Milward, M. Pinkal, M. Poesio, and S. Pulman. 1996. Using the framework. *Technical Report LRE 62-051r*. http://www.cogsci.ed.ac.uk/ fracas/.

R. Cooper. 2005. Records and record types in semantic theory. *J. Logic and Compututation*, 15(2).

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

C. Fox and S. Lappin. 2005. *Foundations of Intensional Semantics*. Blackwell.

J. Ginzburg and R. Cooper. forthcoming. Ttr for natural language semantics. In C. Fox and S. Lappin, editors, *Handbook of Contemporary Semantic Theory*. Blackwell.

J.-Y. Girard. 1971. Une extension de l'interpretation fonctionelle de gödel à l'analyse et son application à l'élimination des coupures dans et la thèorie des types'. *Proc. 2nd Scandinavian Logic Symposium*. North-Holland.

H. Goguen. 1994. *A Typed Operational Semantics for Type Theory*. Ph.D. thesis, University of Edinburgh.

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

Z. Luo. 1994. *Computation and Reasoning: A Type Theory for Computer Science*. Oxford Univ Press.

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

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

Z. Luo. 2011a. Contextual analysis of word meanings in type-theoretical semantics. In *Logical Aspects of Computational Linguistics (LACL'2011). LNAI 6736*, pages 159–174.

Zhaohui Luo. 2011b. Adjectives and adverbs in type-theoretical semantics. Notes.

Z. Luo. 2012a. Common nouns as types. In D. Bechet and A. Dikovsky, editors, *Logical Aspects of Computational Linguistics (LACL'2012). LNCS 7351*, pages 173–185.

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

B. MacCartney. 2009. *Natural Language Inference*. Ph.D. thesis, Stanford Universisty.

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

P. Martin-Löf. 1984. *Intuitionistic Type Theory*. Bibliopolis.

R. Montague. 1974. *Formal Philosophy*. Yale University Press.

A. Ranta. 1994. *Type-Theoretical Grammar*. Oxford University Press.

C. Retoré. 2013. The Montagovian generative lexicon $\lambda\mathsf{Ty}_n$: an integrated type-theoretical framework for compositional semantics and lexical pragmatics.

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