Formal Semantics in Modern Type Theories with Coercive Subtyping

Zhaohui Luo*

Abstract. In the formal semantics based on modern type theories, common nouns are interpreted as types, rather than as predicates of entities as in Montague's semantics. This brings about important advantages in linguistic interpretations but also leads to a limitation of expressive power because there are fewer operations on types as compared with those on predicates. The theory of coercive subtyping adequately extends the modern type theories and, as shown in this paper, plays a very useful role in making type theories more expressive for formal semantics. It not only gives a satisfactory solution to the basic problem of 'multiple categorisation' caused by interpreting common nouns as types, but provides a powerful formal framework to model interesting linguistic phenomena such as copredication, whose formal treatment has been found difficult in a Montagovian setting. In particular, we show how to formally introduce dot-types in a type theory with coercive subtyping and study some type-theoretic constructs that provide useful representational tools for reference transfers and multiple word meanings in formal lexical semantics.

1 Introduction

Church's simple type theory [Chu40], as employed in Montague's semantics [Mon74], has traditionally served as a logical language for formal semantics. Powerful alternatives, arguably more advantageous ones, may be offered by the modern type theories (MTTs) such as Martin-Löf's type theory [NPS90,ML84] and Unifying Theory of dependent Types (UTT) [Luo94]. The theory of *coercive subtyping* [Luo97,Luo99] adequately extends the modern type theories with a notion of subtyping and, as shown in this paper, plays a very useful role in making modern type theories suitable and more expressive for formal semantics.[1]

First, coercive subtyping gives a satisfactory solution to the basic problem of 'multiple categorisation', as discussed by Ranta [Ran94], and hence provides an important basic representational mechanism for the formal semantics based on modern type theories. In such a semantics, common nouns (CNs) are interpreted

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

[1] The idea of using coercive subtyping in linguistic semantics was considered in [LC98]. Parts of [Luo10] and the current paper may be seen as a further development of those initial ideas.

as types, rather than as predicates as in Montague's semantics.[2] Such a way to interpret CNs is natural since a modern type theory is 'many-sorted' in the sense that there are many types, as compared with the 'single-sorted' simple type theory in Montague's semantics, where there is only one type of entities. However, this has led to some problems as well as important advantages (see below for the latter). A basic problem, the problem of 'multiple categorisation', is that, interpreting CNs as types, a verb can now have many different types. For example, 'walk' may be interpreted as a predicate that can range over both the types that interpret 'man' and 'handsome man' ($[\![man]\!]$ and $[\![handsome\ man]\!]$, respectively) because, for instance, we need to interpret 'John walks' when 'John' is either a 'man' or a 'handsome man'. A natural way to deal with this is to introduce some notion of subtyping: for $[\![walk]\!]$ that ranges over $[\![man]\!]$, if $[\![handsome\ man]\!]$ is a subtype of $[\![man]\!]$, then $[\![walk]\!]([\![John]\!])$ is well-typed even when 'John' is a 'handsome man'. Coercive subtyping, as studied in [Luo97,Luo99] and other papers, provides us with such a subtyping framework. It adequately extends modern type theories with a suitable notion of subtyping and provides a solution to the above problem.

Secondly, the formal semantics based on MTTs with coercive subtyping provides us with more advantageous treatments of many interesting linguistic phenomena that involve subtyping relations. In linguistic semantics, there have been a lot of interesting developments including, for example, the Generative Lexicon Theory developed by Pustejovsky [Pus95] and some advanced studies in lexical semantics such as that by Asher [Ash11]. Many case studies and their analyses are based on subtypes of entities. For example, in a Montagovian setting, people may consider the types PHY/INFO of physical/informational entities, which are subtypes of the type e of all entities, and interpret the adjective 'heavy' and CN 'book' as follows:

$$[\![heavy]\!] : (\text{PHY} \to t) \to (\text{PHY} \to t)$$
$$[\![book]\!] : \text{PHY} \bullet \text{INFO} \to t$$

where the dot-type $\text{PHY} \bullet \text{INFO}$ [Pus95] is a type of entities with both physical and informational aspects (and a book has both). However, this has a problem: the way that CNs are interpreted as predicates seems to be incompatible with the subtyping mechanism. For instance, in such a setting, it even becomes rather difficult to interpret the modified CN 'heavy book': in order to apply $[\![heavy]\!]$ to $[\![book]\!]$, we would need the following subtyping relation:

$$\text{PHY} \bullet \text{INFO} \to t \ \leq \ \text{PHY} \to t,$$

i.e., we would need (by contravariance)

$$\text{PHY} \leq \text{PHY} \bullet \text{INFO}.$$

[2] The idea of interpreting CNs as types is further investigated in [Luo12] where, in particular, Geach's observation [Gea62] that CNs have criteria of identity is studied to justify this idea and, furthermore, it is pointed out that proof irrelevance should be adopted in a modern type theory in order for CNs to be interpreted as types adequately. See §2.1 for more on this.

But, both intuitively and formally, this is not the case! Actually, in our formal treatment of dot-types in §4, the opposite is true: PHY•INFO $\leq$ PHY. As shown in this paper, because CNs are interpreted as types in the formal semantics based on modern type theories, the interpretation of 'heavy book' becomes straightforward as expected. We show that modern type theories, together with the theory of coercive subtyping, may offer a powerful language in which interesting lexical phenomena such as copredication [Pus95,Ash11] can be properly interpreted.

This paper also studies how the dot-types (or sometimes called dot-objects), as proposed by Pustejovsky [Pus95] in studying logical polysemy and cocomposition in lexical semantics, can be formally introduced in a type theory with coercive subtyping. Informally, a dot-type $A \bullet B$ is a type of pairs with the following two important requirements:

- $A \bullet B$ is only well-formed if A and B do not share common components, and
- both projections, one from $A \bullet B$ to A and the other from $A \bullet B$ to B, are coercions in the coercive subtyping framework.

We claim that this formal treatment of the dot-type operator captures its essence and, hence, can be used in a formal semantics to model copredication and other phenomena in a satisfactory way.

Modern type theories with coercive subtyping also provide us with various mechanisms very useful in describing linguistic phenomena in formal semantics. These include, for example, how to introduce formal notions of coercion contexts and local coercions in order to represent linguistic phenomena such as reference transfers in particular contexts and multiple contextual word meanings. Such mechanisms further strengthen the claim that MTTs with coercive subtyping provide powerful languages for formal semantics.

In §2, we give brief introductions to formal semantics based on modern type theories and to the theory of coercive subtyping, discussing several relevant issues of the background and introducing notational conventions. Two basic applications of coercive subtyping in the formal semantics based on MTTs, as briefly introduced above, are further discussed in §3: (1) the multiple categorisation problem and its solution; and (2) the subtyping problem in copredication and its treatment with coercive subtyping. This is followed by a formal treatment of dot-types in type theory with coercive subtyping in §4. Type-theoretical mechanisms of coercion contexts and local coercions are discussed in §5.

2 Modern Type Theories and Coercive Subtyping

2.1 Modern Type Theories and Formal Semantics

Modern type theories may be classified into the predicative type theories such as Martin-Löf's type theory [NPS90,ML84] and the impredicative type theories such as the Calculus of Constructions (CC) [CH88] and the Unifying Theory of dependent Types (UTT) [Luo94]. In computer science, modern type theories have been implemented in the proof assistants such as Agda [Agd08] and Coq

[Coq07] and used in applications to formalisation of mathematics and verification of programs.

A formal semantics of natural languages based on modern type theories is in the tradition of the Montague semantics [Mon74] but the powerful type structures in a modern type theory provide new useful mechanisms for formal semantics of various linguistic features, some of which have been found difficult to describe in the Montagovian setting.

Basics of formal semantics based on MTTs. The Montague semantics is based on Church's simple type theory [Chu40], which is a *single-sorted* logic. In Montague's semantics, there is a universal type e of entities: a common noun or a verb is interpreted as a function of type $e \to t$ and an adjective as a function of type $(e \to t) \to (e \to t)$, where t is the type of truth values.

In contrast, a modern type theory can be considered as a *many-sorted* logical system, where there are many sorts called *types* that may be used to stand for the domains to be represented. These types include:

- the *propositional types* (or logical propositions),
- the *inductive types* such as the type of natural numbers and Σ-types of dependent pairs, and
- other more advanced type constructions such as *type universes*.

Each of these is further explained below in this subsection.

Because of this many-sortedness, it is natural to interpret the noun phrases as types. Here are several basic interpretation principles one may adopt in a formal semantics based on MTTs:[3]

- Common nouns are interpreted as types. For instance, the CNs 'man' and 'human' can be interpreted as types $[\![man]\!]$ and $[\![human]\!]$, respectively.
- An adjective is interpreted as a predicate over the type that interprets the domain of the adjective. For instance, 'handsome' may be interpreted as a predicate $[\![handsome]\!] : [\![man]\!] \to Prop$, where $Prop$ is the type of logical propositions (see below for more on $Prop$).
- When the modifying adjectives are subsective, modified CNs are interpreted as Σ-types such as $\Sigma([\![man]\!], [\![handsome]\!])$, a type of handsome men. (See below for more details on Σ-types and Footnote 8 for comments on non-subsective adjectives).

As observed by Geach [Gea62], a component of meaning of CNs is the criteria of identity and, as discussed in [Luo12], it may be argued that this offers a justification for CNs to be interpreted as types in an MTT.

[3] Many basic ideas of developing formal semantics in Martin-Löf's type theory have been studied in [Ran94]. Ranta himself may not regard his work as studying logical semantics (see, for example, the preface of [Ran94]). However, if one looks at it from a technical (and non-philosophical) point of view, the work has studied the basics of formal semantics in a modern type theory and made many valuable proposals.

Furthermore, the framework of coercive subtyping [Luo99] provides us with additional richer means for formal semantics, as discussed in [Luo10,Luo11b] and the current paper.

Embedded logic. A modern type theory has an *embedded logic* (or *internal logic*) based on the propositions-as-types principle [CF58,How80]. For example, in Martin-Löf's predicative type theory, the logical proposition A & B corresponds to the product type $A \times B$ (a special case of Σ-type – see below) and a pair of a proof of A and a proof of B corresponds to an object of the product type. Similarly, this correspondence extends to other logical operators: the logical implication ($\supset$) corresponds to the function types ($\rightarrow$), the universal quantifier ($\forall$) to the dependent Π-types, etc.

For Martin-Löf's type theory, the embedded logic is first-order and, for impredicative type theories such as ECC/UTT [Luo94], the embedded logics are second-order or higher-order, where there is a type $Prop$ of logical propositions. Formally, $Prop$ is a totality and one can quantify over it to form other propositions (and this process is regarded as 'circular' by predicativists [Fef05] or 'impredicative', in the technical jargon).[4] As pointed out in [Luo12], proof irrelevance in an MTT is crucially important for types to be used to interpret CNs adequately (for example, when Σ-types are used to interpret modified CNs). By proof irrelevance, we mean that, informally, any two proofs of the same logical proposition are the same. It is worth remarking that, while proof irrelevance can be formulated straightforwardly for MTTs such as UTT where there is a clear distinction between logical propositions and data types, it is unclear how it should or can be done for those MTTs without such a distinction (eg, Martin-Löf's type theory).

In this paper, we shall use $Prop$ in linguistic interpretations. In a type theory with type $Prop$ of propositions, an assertive sentence is interpreted as a proposition of type $Prop$ and a verb or an adjective as a predicate of type $A \rightarrow Prop$, where A is the domain whose objects the verb or adjective can be meaningfully applied to. For instance, consider the following sentence:

(1) John is handsome.

With $[\![John]\!] : [\![man]\!]$ and $[\![handsome]\!] : [\![man]\!] \rightarrow Prop$, the above sentence (1) is interpreted as proposition $[\![handsome]\!]([\![John]\!])$ of type $Prop$.

Types in modern type theories. We assume that the reader be familiar with the simple type theory [Chu40] (or Montague's semantics [Mon74]), compared with which modern type theories have several distinctive features that are briefly described below.

[4] Intuitively, $Prop$ is very much like the type t in the simple type theory. A main difference is that, in modern type theories, we have explicit proof terms of logical propositions.

Dependent types. Modern type theories contain *dependent types*, examples of which include Π-types and Σ-types. The so-called Σ-types are types of dependent pairs. If A is a type and B is an A-indexed family of types, then $\Sigma(A, B)$ is a type, consisting of pairs (a, b) such that a is of type A and b is of type $B(a)$. For instance, $\Sigma([\![man]\!], [\![handsome]\!])$ is a type of handsome men (or more precisely, of those men together with proofs that they are handsome). When $B(x)$ is a constant type (i.e., always the same type no matter what x is), a Σ-type degenerates into a product type of non-dependent pairs. For Σ-types (and product types), there 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 $\Sigma(A, B)$.

There are other dependent types such as the Π-type $\Pi(A, B)$ of dependent functions which, in the non-dependent case, degenerates to the function type $A \to B$. The objects of $\Pi(A, B)$ are λ-functions f which can be applied to any object a of type A to form $f(a)$ of type $B(a)$ (or just B in the non-dependent case).

Inductive types and canonical objects. In a modern type theory, most of the types are inductively-defined, examples of which include the types of natural numbers, trees, ordinals, etc. The above Σ-type is another example.

An important feature of a modern type theory is that its meaning theory, as advocated by Martin-Löf [ML96] and others (such as Dummett [Dum75,Dum91] and Prawitz [Pra74] in the wider context of proof-theoretic semantics), is based on the notion of *canonical object*. An inductively-defined type consists of its canonical objects. For example, the type of natural numbers consists of the canonical numbers 0, 1, 2, ..., and the other natural numbers all compute to canonical numbers.[5] For instance, $3+4$ is a natural number because it computes to the canonical number 7.

The notion of canonical object is so important that modern type theories may also be called *type theories with canonical objects*. Based on it, every inductive type is equipped with an induction principle (so-called elimination rule) expressing that, in order to prove a property for all objects of the inductive type, one only has to prove it for all of its canonical objects. The modern type theories have the following important property:

- *Canonicity*: Any closed object of an inductive type is definitionally equal to a canonical object of that type.

Universes. Other more advanced features in a modern type theory are useful in developing formal semantics. For example, one may collect (the names of) some types into a type called a *universe* [ML84]. Introducing universes can be considered as a reflection principle: such a universe reflects those types whose names are its objects.

[5] The notion of computation is also in the centre of the meaning theory of modern type theories. Intuitively, in a modern type theory, every process of computation starting from a well-typed term terminates and it computes to a (unique) 'normal form'.

In formal semantics, universes can be employed to help semantic interpretations. For instance, one may consider the universe CN : $Type$ of all common noun interpretations and, for each type A that interprets a common noun, there is a name $\overline{A}$ in CN. For example,

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

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, $[\![man]\!] : \text{CN}$.

As another example, the universe CN can be used to give semantic interpretations to adverbs. An adverb modifies a verb (an adjective) to result in a verb (adjective) phrase.[6] Since in MTTs verbs and adjectives are interpreted as predicates over a variety of domains (rather than over a single domain as in the Montagovian setting), adverbs such as 'loudly' in 'John talked loudly' and 'simply' in 'That idea is simply ridiculous' would be interpreted as of type

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

For instance, for $[\![talk]\!] : [\![human]\!] \to Prop$, the following phrase (2) can be interpreted as (3), which is of type $[\![human]\!] \to Prop$:

(2) talk loudly

(3) $[\![loudly]\!]([\![human]\!], [\![talk]\!])$

2.2 Coercive Subtyping

Coercive subtyping [Luo97,Luo99,LSX12] is a general theory of subtyping for modern type theories. In computer science, coercive subtyping has been studied and implemented in many proof assistants such as Coq [Coq07,Saï97], Lego [LP92,Bai99], Matita [Mat08] and Plastic [CL01], and used effectively in interactive theorem proving. In this paper, coercive subtyping is applied to linguistic semantics.

We shall now introduce informally the basics of coercive subtyping and explain why the extension is adequate for modern type theories.

Basics of coercive subtyping. The basic idea of coercive subtyping is to consider subtyping 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 $\mathfrak{C}_B[c(a)]$. (See Figure 1 for a pictorial explanation.)

[6] There are other adverbs. For example, an adverb may modify sentences to result in new sentences and, as in Montague grammar [Mon74], such adverbs are interpreted as functions from $Prop$ to $Prop$.

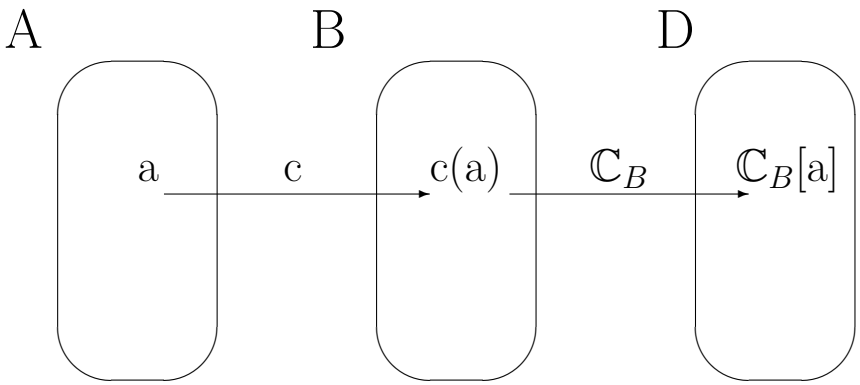

Fig. 1. Pictorial explanation of $A <_c B$

For instance, one may consider the type of men to be a subtype of the type of human beings by declaring a coercion between them:

$$[\![man]\!] <_m [\![human]\!] .$$

If we assume that 'shout' and 'John' be interpreted as

$$[\![shout]\!] : [\![human]\!] \to Prop$$
$$[\![John]\!] : [\![man]\!]$$

then the interpretation (5) of the following sentence (4) is well-typed:

(4) John shouts.
(5) $[\![shout]\!]([\![John]\!])$

The reason that (5) is well-typed is that $[\![man]\!]$ is now a subtype of $[\![human]\!]$. In general, this is reflected by the following *coercive definition rule*:

$$\frac{\Gamma \vdash f : (B)C \quad \Gamma \vdash a : A \quad \Gamma \vdash A <_c B : Type}{\Gamma \vdash f(a) = f(c(a)) : C}$$

expressing that an appropriate coercion can be inserted to fill up the gap in a term. (See [Luo99,LSX12] for a formal presentation with complete rules.)

Notation We shall adopt the following notational abbreviations, writing

- $A < B$ for '$A <_c B : Type$ for some c', and
- $A \leq B$ for '$A = B : Type$ or $A < B$'. □

Subsumptive subtyping v.s. coercive subtyping. In set theory, one has the subset relation between sets. Similarly, in a type theory, one can consider a notion of subtyping between types. To mimic the subset relation, the subtyping relation

is traditionally captured by *subsumptive subtyping* characterised by means of the following *subsumption rule*:

$$(*) \qquad \frac{a : A \qquad A \text{ is a subtype of } B}{a : B}$$

which says that, if A is a subtype of B, every object of type A is also of type B.

Unfortunately, subsumptive subtyping is incompatible with canonicity and cannot be employed in a modern type theory. This is because that the induction principles (the elimination rules) do not take into account of the objects introduced by subsumptive subtyping relations and, as a consequence, the subsumptive rule $(*)$ would introduce inconsistency in a modern type theory.[7]

Coercive subtyping, on the other hand, is a suitable subtyping framework for modern type theories. As compared with subsumptive subtyping, coercive subtyping does not introduce new objects into a type. In the framework of coercive subtyping, $A < B$ means that there is a (unique) coercion that maps any object of A to an object of B. This is consistent with the idea of canonical object – if B is an inductive type, we do not need to change its elimination rule, since B will still have the same objects even if $A < B$. Furthermore, the extension with coercive subtyping is *adequate*, as explained below.

Conservativity: adequacy of the coercive subtyping extension. When using a type theory for formal semantics, a basic requirement is that the type system has a consistent embedded logic. For example, the higher-order logic embedded in the simple type theory is consistent and this is the basis for it to be employed in the Montague semantics. When one wants to extend a type theory in order to represent certain linguistic phenomena, one of the first things one has to make sure is that the extension does not in any way jeopardise the consistency of the embedded logic.

Such a requirement applies to the extension of modern type theories with coercive subtyping and, fortunately, it meets the requirement. In fact, the coercive subtyping extension is not only consistent but *conservative* as long as the employed coercions are coherent [SL02,LSX12]. Informally, coherence means that any two coercions between the same two types must be the same (see [Luo99] for a formal definition.) As coercions may be declared by the users, they must be *coherent* to be employed correctly (and to guarantee conservativity). For a type theory with nice meta-theoretic properties such as Strong Normalisation (and hence logical consistency), its extension with coercive subtyping has those properties, too. Intuitively, this says that the coercive subtyping extension is adequate and can safely be used in various applications, including linguistic semantics.

[7] It is not difficult to see this if one is familiar with how the induction principles of inductive types are formulated in a modern type theory. As it requires some formal technical details, we omit it here. See §4 of [LSX12] for more on this.

3 Coercive Subtyping in MTT-based Formal Semantics

As explained above, the most important difference between the Montague semantics and the formal semantics based on modern type theories is that common nouns are interpreted as predicates in the former, but as types in the latter. Interpreting common nouns as types brings about important advantages in linguistic interpretations (see, for example, later in this section on copredication) but has some unwelcome consequences because there are fewer operations on types as compared with those on predicates.

For example, one can easily define a subset relation between the functional subsets represented by predicates of entities: for s and s' of type $e \to t$, $s \subseteq s'$ if and only if $\forall x : e.\ s(x) \supset s'(x)$. This notion of subset has been used substantially in various semantic investigations based on the Montague semantics. However, such a notion between types is not that straightforward. As we have discussed in §2.2, the traditional notion of subtyping, subsumptive subtyping, is incompatible with the notion of canonical object in modern type theories. Instead, the theory of coercive subtyping [Luo99,LSX12] adequately extends the modern type theories which, as we show in this section, can be employed to play a very useful role in, for example, giving a satisfactory treatment of the type-theoretic interpretations of modified common nouns and allowing straightforward interpretations of interesting linguistic phenomena such as copredication, whose interpretations have been found difficult in the Montagovian setting.

Common nouns as types. In an MTT, the interpretations of CNs like 'man', 'human' and 'book' are types:

$$[\![man]\!],\ [\![human]\!],\ [\![book]\!] : Type,$$

and verbs and adjectives are interpreted as predicates: for example, we have

$$[\![heavy]\!] : [\![book]\!] \to Prop$$
$$[\![read]\!] : [\![human]\!] \to [\![book]\!] \to Prop$$

When the modifying adjectives are subsective,[8] the modified common noun phrases are interpreted by means of Σ-types of dependent pairs: for instance,

$$[\![heavy\ book]\!] = \Sigma([\![book]\!], [\![heavy]\!]).$$

[8] There are non-subsective adjectives. For example, adjectives may be privative; examples include *fake* and *counterfeit*. Partee [Par10] has argued that the so-called privative adjectives should actually be analysed as subsective adjectives whose treatment calls for the employment of some special kind of coercions. In fact, this can be formalised in an MTT by means of the disjoint union types with the associated injection operators as coercions. For instance, let G_R and G_F be the type of (real) guns and that of fake guns, respectively. Then, $G = G_R + G_F$ is the type of all guns and we can declare $G_R <_{inl} G$ and $G_F <_{inr} G$, where inl and inr are the two injection operators for the disjoint union type G. More details can be found in [Luo11a].

To interpret common nouns as types has its advantage in that it effectively distinguishes *meaningless* and *false* expressions. For example, consider the verb 'talk':

(6) In the Montague semantics: $[\![talk]\!]_M : e \to t$

(7) In the formal semantics based on MTTs: $[\![talk]\!]_T : [\![human]\!] \to Prop$

Now consider the following sentence (8), which is given interpretation (9) in Montague's semantics and (10) in the formal semantics based on MTTs:

(8) A table talks.

(9) $\exists x : e.\ [\![table]\!]_M(x)\ \&\ [\![talk]\!]_M(x)$

(10) $\exists t : [\![table]\!]_T\ .\ [\![talk]\!]_T(t)$

where $[\![table]\!]_M : e \to t$ is a predicate while $[\![table]\!]_T$ is a type. Now, the term (9) is well-typed (and false), while the term (10) is simply not well-typed, i.e., meaningless. We contend that, in this respect, the formal semantics based on MTTs captures the meanings in a better way: the sentence (8) is usually regarded as meaningless (unless in some fictional world) as in formal semantics based on MTTs.

Coercive subtyping: basic applications. A basic problem in formal semantics based on MTTs has been the so-called problem of 'multiple categorisation' of verbs (p60 of [Ran94]). Because CNs are interpreted as types and, in particular, modified CNs as Σ-types, a verb may need to have many different types. Consider, for instance,

$$[\![John]\!] : [\![man]\!]$$
$$[\![W\&P]\!] : [\![heavy\ book]\!] = \Sigma([\![book]\!], [\![heavy]\!])$$

where W&P abbreviates the book 'War and Peace'. Now, how do we interpret the following sentences?

(11) John reads a book.

(12) Somebody reads 'War and Peace'.

(13) John reads 'War and Peace'.

Note that the type of $[\![read]\!]$ is $[\![human]\!] \to [\![book]\!] \to Prop$. To interpret the above sentences directly would require the following three terms to be well-typed:

(14) $\exists b : [\![book]\!]\ .\ [\![read]\!]([\![John]\!], b)$.

(15) $\exists h : [\![human]\!]\ .\ [\![read]\!](h, [\![W\&P]\!])$.

(16) $[\![read]\!]([\![John]\!], [\![W\&P]\!])$.

But none of the above terms is well-typed. $[\![read]\!]$ requires its first argument to be of type $[\![human]\!]$ and its second of type $[\![book]\!]$, but $[\![John]\!]$ is of type $[\![man]\!]$ and $[\![W\&P]\!]$ is of type $[\![heavy\ book]\!]$. Or, put in another way, how could we reflect the following facts:

– A man is a human.
– A heavy book is a book.

Such phenomena are captured by means of coercive subtyping. For the first case, we declare that $[\![man]\!]$ is a subtype of $[\![human]\!]$:[9]

$$[\![man]\!] < [\![human]\!] .$$

For the second case, we have

$$[\![heavy\ book]\!] < [\![book]\!],$$

and, in fact, we declare in general that the first projection π_1 of a Σ-type is always a coercion, for any type A and any A-indexed family of types B:

$$\Sigma(A, B) <_{\pi_1} A.$$

Furthermore, the subtyping relations propagate through the type constructors such as Π and Σ (and, in the non-dependent cases, $\to$ and $\times$). For instance, they propagate through the function types, contravariantly: if $A' \leq A$ and $B \leq B'$, then $A \to B \leq A' \to B'$. For example,

$$[\![human]\!] \to [\![book]\!] \to Prop < [\![man]\!] \to [\![heavy\ book]\!] \to Prop.$$

With these subtyping relations, the terms in (14-16) are well-typed and they interpret the sentences in (11-13), respectively.

Remark The above problem has been a fundamentally troublesome one for a formal semantics based on MTTs. It seems that what we need is an appropriate notion of subtyping and, with this provided by coercive subtyping, the MTT-based semantics, as initially studied in [Ran94], seems to provide us with a very promising alternative to the Montague semantics. □

As another example, let's consider the dot-types as studied in [Pus95].[10] Let PHY and INFO be the types of physical objects and informational objects, respectively. One may consider the dot-type PHY•INFO as the type of the objects with both physical and informational aspects. Intuitively, a dot-type is a subtype of its constituent types: PHY • INFO < PHY and PHY • INFO < INFO. A book may be considered as having both physical and informational aspects, reflected as: $[\![book]\!]$ < PHY • INFO. By contravariance,

$$\text{PHY} \to Prop < \text{PHY} \bullet \text{INFO} \to Prop < [\![book]\!] \to Prop$$
$$\text{INFO} \to Prop < \text{PHY} \bullet \text{INFO} \to Prop < [\![book]\!] \to Prop$$

[9] There is another possibility: that is when $[\![man]\!]$ is defined to be the Σ-type $\Sigma([\![human]\!], [\![male]\!])$. In such a case, $[\![man]\!]$ is a subtype of $[\![human]\!]$ by means of the first projection being a coercion.

[10] Dot-types will formally be studied in §4. Here, we study the example informally.

Therefore, for example, for $[\![burn]\!] : \textsc{Phy} \to Prop$ and $[\![boring]\!] : \textsc{Info} \to Prop$, the following phrase (17) can be interpreted by term (18), as intended:

(17) burn a boring book

(18) $\exists b : \Sigma([\![book]\!], [\![boring]\!]).\ [\![burn]\!](b)$

Remark In Montague's semantics (and its extensions), common nouns are interpreted as predicates of type $e_0 \to t$, where e_0 is a subtype of the type e of entities. For instance, $[\![book]\!] : \textsc{Phy} \bullet \textsc{Info} \to t$ and $[\![heavy]\!] : (\textsc{Phy} \to t) \to (\textsc{Phy} \to t)$. In such a situation, in order to interpret, e.g., 'a heavy book', one would have to apply $[\![heavy]\!]$ to $[\![book]\!]$ by requiring, for example, $\textsc{Phy} \bullet \textsc{Info} \to t$ to be a subtype of $\textsc{Phy} \to t$, which is not the case – type clashes would happen, leading to unnatural and complicated treatments [Ash08]. □

Copredication. Another example to illustrate the use of coercive subtyping is the interpretation of copredication. Copredication has been studied by [Ash11] among others, where the following example is considered:

(19) John picked up and mastered the book.

The idea is that the interpretations of the phrases 'pick up' and 'master' should be of the same type so that the use of 'and' in the above sentence can be interpreted in a straightforward way. Now, when we consider the types Phy and Info as above, it is natural that these phrases have the following types:

$$[\![pick\ up]\!] : [\![human]\!] \to \textsc{Phy} \to Prop$$
$$[\![master]\!] : [\![human]\!] \to \textsc{Info} \to Prop$$

By coercive subtyping (and contravariance for function types), we have

$$\begin{aligned} [\![pick\ up]\!] \ : \ & [\![human]\!] \to \textsc{Phy} \to Prop \\ < \ & [\![human]\!] \to \textsc{Phy} \bullet \textsc{Info} \to Prop \\ < \ & [\![human]\!] \to [\![book]\!] \to Prop \end{aligned}$$

$$\begin{aligned} [\![master]\!] \ : \ & [\![human]\!] \to \textsc{Info} \to Prop \\ < \ & [\![human]\!] \to \textsc{Phy} \bullet \textsc{Info} \to Prop \\ < \ & [\![human]\!] \to [\![book]\!] \to Prop \end{aligned}$$

In other words, $[\![pick\ up]\!]$ and $[\![master]\!]$ can both be used in contexts where terms of type $[\![human]\!] \to [\![book]\!] \to Prop$ are required and, therefore, the interpretation of the sentence (19) can proceed straightforwardly as intended.

Remark In a Montagovian setting, the interpretations of such sentences with copredication can become rather sophisticated (see, for example, [AP05]). This is

because, in Montague's semantics, common nouns are interpreted as predicates. Such an interpretation seems incompatible with the subtyping relationships involving PHY and INFO. In a formal semantics based on MTTs with coercive subtyping, where common nouns are interpreted as types, the interpretation of sentences with copredication is quite straightforward. □

4 Dot-Types in Type Theory with Coercive Subtyping

Dot-types, sometimes called dot objects or complex types, are proposed by Pustejovsky in his Generative Lexicon Theory [Pus95]. Although the meaning of a dot-type is intuitively clear, its proper formal account seems surprisingly difficult and tricky.[11] Researchers have made several proposals to model dot-types including, for example, [AP05] and [Coo07,Coo11]. There are arguments about whether these do capture and therefore give successful formal accounts of dot-types and the author contends that the issue has not been settled.

In the following, we present a type-theoretic treatment of dot-types with the help of coercive subtyping[12] which, we believe, gives an adequate formal account of dot-types and can hence be used in an MTT-based formal semantics to interpret, for instance, copredication (as in §3) and logical polysemy as studied in [Pus95].

In §3, we have already introduced the dot-types PHY • INFO informally with examples. An important feature is that, to form a dot-type $A \bullet B$, its constituent types A and B should not share common parts. This is consistent with Pustejovsky's idea as expressed in the following paragraph:

> *Dot objects have a property that I will refer to as* inherent polysemy. *This is the ability to appear in selectional contexts that are contradictory in type specification.* [Pus05]

In the same spirit, a dot-type $A \bullet B$ can only be formed if the types A and B do not share any components. For instance,

- PHY • PHY should not be a dot-type because its constituent types are the same type PHY.
- PHY • (PHY • INFO) should not be a dot-type because its constituent types PHY and PHY • INFO share the component PHY.

The notion of component is formally defined as follows.

[11] See, for example, [Ash08] for an interesting discussion on various choices of representation, where you can also find a semantic account of the meaning of dot-types in category theory.

[12] The idea of using coercive subtyping to model dot-types was considered in [LC98] where, however, the product types and the associated projection coercions were proposed; this is known to be incoherent from Y. Luo's PhD thesis [Luo05].

Definition 1 (components) *Let $T : Type$ be a type in the empty context. Then, $\mathcal{C}(T)$, the set of components of T, is defined as:*

$$\mathcal{C}(T) =_{df} \begin{cases} \textsc{Sup}(T) & \text{if the normal form of } T \text{ is not of the form } X \bullet Y \\ \mathcal{C}(T_1) \cup \mathcal{C}(T_2) & \text{if the normal form of } T \text{ is } T_1 \bullet T_2 \end{cases}$$

where $\textsc{Sup}(T) = \{T' \mid T \leq T'\}$. □

The rules for the dot-types are given in Figure 2.[13] Note that, in the formation rule, $\langle\rangle$ is the empty context and we require that the constituent types do not share common components: $\mathcal{C}(A) \cap \mathcal{C}(B) = \emptyset$. Once well-formed, a dot-type behaves somehow like a product type: intuitively, its objects are pairs and the projections p_1 and p_2 correspond to the projection operations π_1 and π_2, respectively. However, there are two important differences between dot-types and product types:

- The constituent types of a dot-type do not share components, while those of a product type can.
- The projections p_1 and p_2 for dot types are *both* coercions; this is OK (see Proposition 2 below). For product types, however, only one of them can be coercions for, otherwise, coherence would fail [Luo05].

According to the rules in Figure 2, $A \bullet B$ is a subtype of A and a subtype of B. In other words, an object of the dot-type $A \bullet B$ can be regarded as an object of type A, in a context requiring an object of A, and similarly for B. This is crucial for, for example, the well-typedness of terms in the examples in §3 that involve the dot-type $\textsc{Phy} \bullet \textsc{Info}$. Finally, the subtyping relations are propagated through the dot-types, by means of the coercions d_1, d_2 and d as specified in the last three rules in Figure 2.

Since the constituent types of a well-formed dot-type do not share components, it is straightforward to prove the following coherence property. (Recall that, informally, coherence means that there is at most one coercion between any two types.)

Proposition 2 (coherence) *The coercions p_1, p_2, d_1, d_2 and d are coherent together.* □

Note that coherence is important as it guarantees the correctness of employing the projections p_1 and p_2 and the propagation operators d_1, d_2 and d as coercions, and hence the subtyping relationships $A \bullet B < A$ and $A \bullet B < B$.

Remark If the constituent types of a dot-type shared a common component, coherence would fail. For instance, $\textsc{Phy}$ and $\textsc{Phy} \bullet \textsc{Info}$ share the component

[13] The general forms of the derivation rules in type theory are $\Gamma \vdash J$, where Γ is a context (see §5 for more explanations of contexts) and J is a judgement of the forms such as $t : T$ or $A <_c B$.

Formation Rule

$$\frac{\Gamma \ valid \quad \langle\rangle \vdash A : Type \quad \langle\rangle \vdash B : Type \quad \mathcal{C}(A) \cap \mathcal{C}(B) = \emptyset}{\Gamma \vdash A \bullet B : Type}$$

Introduction Rule

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B \quad \Gamma \vdash A \bullet B : Type}{\Gamma \vdash \langle a, b\rangle : A \bullet B}$$

Elimination Rules

$$\frac{\Gamma \vdash c : A \bullet B}{\Gamma \vdash p_1(c) : A} \qquad \frac{\Gamma \vdash c : A \bullet B}{\Gamma \vdash p_2(c) : B}$$

Computation Rules

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B \quad \Gamma \vdash A \bullet B : Type}{\Gamma \vdash p_1(\langle a, b\rangle) = a : A} \qquad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B \quad \Gamma \vdash A \bullet B : Type}{\Gamma \vdash p_2(\langle a, b\rangle) = b : B}$$

Projections as Coercions

$$\frac{\Gamma \vdash A \bullet B : Type}{\Gamma \vdash A \bullet B <_{p_1} A : Type} \qquad \frac{\Gamma \vdash A \bullet B : Type}{\Gamma \vdash A \bullet B <_{p_2} B : Type}$$

Coercion Propagation

$$\frac{\Gamma \vdash A \bullet B : Type \quad \Gamma \vdash A' \bullet B' : Type \quad \Gamma \vdash A <_{c_1} A' : Type \quad \Gamma \vdash B = B' : Type}{\Gamma \vdash A \bullet B <_{d_1[c_1]} A' \bullet B' : Type}$$

where $d_1[c_1](x) = \langle c_1(p_1(x)), p_2(x)\rangle$.

$$\frac{\Gamma \vdash A \bullet B : Type \quad \Gamma \vdash A' \bullet B' : Type \quad \Gamma \vdash A = A' : Type \quad \Gamma \vdash B <_{c_2} B' : Type}{\Gamma \vdash A \bullet B <_{d_2[c_2]} A' \bullet B' : Type}$$

where $d_2[c_2](x) = \langle p_1(x), c_2(p_2(x))\rangle$.

$$\frac{\Gamma \vdash A \bullet B : Type \quad \Gamma \vdash A' \bullet B' : Type \quad \Gamma \vdash A <_{c_1} A' : Type \quad \Gamma \vdash B <_{c_2} B' : Type}{\Gamma \vdash A \bullet B <_{d[c_1,c_2]} A' \bullet B' : Type}$$

where $d[c_1, c_2](x) = \langle c_1(p_1(x)), c_2(p_2(x))\rangle$.

Fig. 2. The rules for dot-types.

PHY. If PHY • (PHY • INFO) were a dot-type, there would be the following two coercions p_1 and $p_1 \circ p_2$:

$$\begin{array}{lll} \text{PHY} \bullet (\text{PHY} \bullet \text{INFO}) & <_{p_1} & \text{PHY} \\ \text{PHY} \bullet (\text{PHY} \bullet \text{INFO}) & <_{p_1 \circ p_2} & \text{PHY} \end{array}$$

which are between the same types but not equal – coherence would then fail. □

Dot-types have been implemented [XL12]. They are implemented as special types in the proof assistant Plastic [CL01], which implements UTT. One can use a proof assistant based on an MTT to implement the MTT-based formal semantics and then use the implemented dot-types to conduct experiments. (See, for example, [Luo11b,XL12] for further details.)

5 Coercion Contexts and Local Coercions

As claimed in the introduction, modern type theories are powerful languages for formal semantics. In this section, we shall study how to introduce useful mechanisms, called *coercion contexts* and *local coercions*, to represent contextual meanings in the framework of coercive subtyping. In particular, such mechanisms provide us with flexible representational mechanisms that are needed in formal semantics when coercive subtyping is employed.

5.1 Coercion Contexts in Type Theory

In some circumstances, the contexts describe either a special situation or a specific background. Many usages are only meaningful in such special situations or *local contexts* in which, for instance, the meanings of some words change.

Example 3 (reference transfer) *Consider the following utterance (cf., [Nun95]):*

(20) The ham sandwich shouts.

Assuming that the act of shouting requires that the argument be human (ie, $[\![shout]\!] : [\![human]\!] \to Prop$), it is obvious that sentence (20) is not well-formed, unless it is uttered by somebody in some special extralinguistic context (e.g., by a waiter in a café to refer to a person who has ordered a ham sandwich). □

In an MTT, such local contexts can be described by means of the formal notion of context. Traditionally, a context in type theory is of the form

$$x_1 : A_1,\ ...,\ x_n : A_n$$

where A_i is either a data type, in which case x_i is assumed to be an object of that type, or a logical proposition, in which case the proposition A_i is assumed to be true and x_i a proof of A_i. For example, one may have the following context:

$$m : [\![man]\!],\ hproof : [\![handsome]\!](m)$$

which assumes, in layman's terms, that 'm is a man' and 'm is handsome' (with 'hproof' being a proof).

The formal notion of context can be extended by coercion declarations or subtyping assumptions, as proposed in [Luo10]. A *coercion context* is a context whose entries may be of the form $A <_c B$ as well as the usual form $x : A$. For instance, the following context may be used to describe the special circumstances in a café:

(21) ..., $[\![ham\ sandwich]\!] < [\![human]\!]$, ...

where the subtyping assumption says that a ham sandwich can be coerced into a person (i.e., the person who has ordered a ham sandwich). In a context such as (21), the above sentence (20) can be interpreted satisfactorily as intended.[14]

Formally, coercion contexts obey the following rules, besides those usual formation rules for valid contexts:

$$\frac{\Gamma \vdash A : Type \quad \Gamma \vdash B : Type \quad \Gamma \vdash c : (A)B}{\Gamma,\ A <_c B\ \ valid} \qquad \frac{\Gamma,\ A <_c B,\ \Gamma'\ \ valid}{\Gamma,\ A <_c B,\ \Gamma' \vdash A <_c B : Type}$$

where $(A)B$ is the functional kind from A to B in the logical framework (see Chapter 9 of [Luo94] for formal details.) In other words, coercions can now be introduced in contexts and they are only valid 'locally' in the context where they are introduced. For example, sentence (20) can be reasonably interpreted in a context which contains the subtyping as shown in (21) and, otherwise, it may not.

Remark (coherent context) Note that validity of a context is not enough anymore for it to be legal. One needs to make sure that the context is *coherent*, in the sense that the declared coercions in the context do not lead to more than one coercion between two types. Since it requires some formal backgrounds to be treated more concisely, its details are omitted here. □

Example 4 *Consider the following example (adapted from [Mar07]):*

(23) Every linguist drinks a glass.

Let's assume that 'drink' be interpreted as:

$$[\![drink]\!] : [\![animated]\!] \to [\![liquid]\!] \to Prop.$$

[14] It may be worth noting that, here, it is still the verbs such as 'shout' that drive the coercion process: for example, it is $[\![shout]\!]$ that requires its argument to be of type $[\![human]\!]$, which then triggers the coercion to be inserted. Consider another sentence (thanks to the anonymous referee for this example):

(22) The ham sandwich tastes awfully.

Here it is the verb 'taste' that triggers a coercion to be inserted: since $[\![taste]\!] : [\![food]\!] \to Prop$, applying $[\![taste]\!]$ to a ham sandwich s to form $[\![taste]\!](s)$ would need a coercion for the subtyping relation $[\![ham\ sandwich]\!] < [\![food]\!]$.

Now, since not every container contains drinks, there should be a special context in which the above sentence (23) can be interpreted. The coercion context should contain the following subtyping relations:

$$[\![glass]\!] < [\![beverage]\!], \quad [\![beverage]\!] < [\![liquid]\!], \quad [\![linguist]\!] < [\![animated]\!]$$

Then, in a coercion context with the above, (23) can be interpreted as:

(24) $\forall l : [\![linguist]\!] \,.\, \exists g : [\![glass]\!] \,.\, [\![drink]\!](l, g)$.

In the coercion context, (24) is well-typed. □

5.2 Local Coercions in Terms

As is well-known, word meanings are context sensitive. In some circumstances, a word appears in two places in a text (or even in the same sentence) and may need two different meaning explanations.

In the simple case of homonymy whose multiple meanings have unrelated types, the framework of coercive subtyping provides us with a useful overloading mechanism that can be used to represent the so-called sense enumeration model (SEM) [Pus95]: when a word w has n different meanings $[\![w]\!]_i$ with unrelated types A_i,[15] its meanings in the SEM can be represented by the coercions $c_i : \mathbf{1}_w \to A_i$ defined as

$$c_i(w) = [\![w]\!]_i : A_i,$$

where $\mathbf{1}_w$ is the unit type with w as its only object (see Appendix A of [Luo11b] for the formal details of unit types). This does not only represent the multiple meanings of a homonymy but provide the selection mechanism that will automatedly choose the correct meaning appropriately. (See Section 3 of [Luo11b] and [Luo11c] for further details of the proposal.)

Note that it is crucial that, in the above cases, the sematic types are *unrelated*: this allows us to employ the coercions c_i $(i = 1, ..., n)$ together in the same context without destroying the coherence requirement. However, in some circumstances, one may need to employ different coercions between the same types at the same time. Here are two examples for explanation. For the first example, let us consider the following sentence (25),[16] whose interpretation (26) involves two different coercions that, informally, insert 'writing' and 'reading', respectively.

(25) John finished a book last year and now everybody is enjoying it.

(26) John finished (writing) a book last year and now everybody is enjoying (reading) it.

These two coercions are both of type $Book \to Event$, ie, they are both functions from the type of books to the type of events. If such coercions are employed

[15] Formally, here '*unrelated*' means that, if A_i and A_j are the types of two different meanings of the homonymous word, then there is no A such that $A_i \leq A$ and $A_j \leq A$.

[16] This example is due to Nicholas Asher in a private communication.

together, it is incoherent – there are more than one coercion between the same types.

As another example, consider the following phrases that use the homonym 'bank':

(27) the bank of the river

(28) the richest bank in the city

The anonymous word 'bank' cannot be disambiguated by the typing of its semantic interpretations: both interpretations, $[\![bank]\!]_1$ for (27) and $[\![bank]\!]_2$ for (28), are types that interpret the common noun 'bank', i.e., they are of the same type CN, the universe of common noun interpretations (see the end of §2.1). Therefore, if we consider two coercions:

$$c_1 : \mathbf{1}_{bank} \to \text{CN}$$
$$c_1(bank) = [\![bank]\!]_1$$

$$c_2 : \mathbf{1}_{bank} \to \text{CN}$$
$$c_2(bank) = [\![bank]\!]_2$$

Both coercions are of the same type and cannot be used together as they are incoherent.

Such a problem can be solved by introducing *local coercions* – coercions that are only effective locally for some terms (expressions in type theory). For instance, the following two terms give the semantics of (27) and (28), respectively, and can be used together with no problem:

(29) **coercion** $\mathbf{1}_{bank} <_{c_1} \text{CN}$ **in** $[\![(27)]\!]$

(30) **coercion** $\mathbf{1}_{bank} <_{c_2} \text{CN}$ **in** $[\![(28)]\!]$

The coercions c_1 and c_2 are declared locally and, therefore, the phrases in (27) and (28) are given semantics (29) and (30), respectively.

Formally, local coercions may be introduced by the following rule:

$$\frac{\Gamma,\ A <_c B \vdash J}{\Gamma \vdash \textbf{coercion}\ A <_c B\ \textbf{in}\ J}$$

where J is any of the following four forms of judgement:

$$k : K, \quad k = k' : K, \quad K\ kind, \quad \text{and} \quad K = K'.$$

For instance, with $J \equiv k : K$, we have

$$\frac{\Gamma,\ A <_c B \vdash k : K}{\Gamma \vdash \textbf{coercion}\ A <_c B\ \textbf{in}\ k : K}$$

Intuitively, the coercions declared locally are only effective in the expressions in the scope of the keyword **in**. For instance, the coercion c_1 in (29) only takes its effects in $[\![(27)]\!]$ and, similarly, c_2 only in $[\![(28)]\!]$, as intended.

The key word **coercion** distributes through the components of J. For example, the following two judgements (31) and (32) are identified:

(31) **coercion** $A <_c B$ **in** $(k : K)$
(32) (**coercion** $A <_c B$ **in** k) : (**coercion** $A <_c B$ **in** K)

The introduction of local coercions broadens the scope of interpretation and provides a very useful tool for appropriate meaning explanations in formal semantics.

6 Conclusion

Studying the application of modern type theories to formal semantics, we have made three-fold contributions in this paper. First, we have demonstrated the crucial role that the theory of coercive subtyping plays in enriching MTTs to become suitable languages for formal semantics. Secondly, dot-types are studied and given a type-theoretic formal formulation that we claim to be adequate. Thirdly, we have studied some important type-theoretic mechanisms such as coercion contexts and local coercions so that MTTs, when equipped with them, become more powerful in practical applications to formal semantics.

In linguistic semantics, various notions of coercion have been proposed and studied (see, for example, [MS88]). Obvious future work includes that to study to what extent the theory of coercive subtyping covers the linguistic notions of type-shifting and type coercion (see, for example, [PR83,Par86,Pus11]). Some of the well-known type-shifting principles can be captured by means of coercive subtyping in a straightforward way. For example, consider the type-shifting law for NPs that lifts the e-type of an NP to its GQ-type $(e \to t) \to t$. In an MTT-based semantics where CNs are interpreted as types, this can be captured as a parameterised coercion $c[A]$: for any A of type CN (ie, A being a type that interprets a common noun),

$$A \ <_{c[A]} \ (A \to Prop) \to Prop,$$

where $c[A]$ is defined as $c[A](x) = \lambda P : A \to Prop.\ P(x)$. It is not difficult to verify that these coercions commute with the subtyping relations as expected: for example, for $Man = [\![man]\!] <_{mh} [\![human]\!] = Human$, we can define

$$g = \lambda f \lambda Q.\ f(Q) \ : \ [(Man \to Prop) \to Prop] \to [Human \to Prop] \to Prop,$$

and, furthermore,

$$g \circ c[Man] = c[Human] \circ mh.$$

Further research is needed to see how much this formal theory can reflect the linguistic notions of coercion.

It is one of our goals to implement a computer system for linguistic reasoning based on the formal semantics based on MTTs and the existing type theory based proof technology. We did some initial experiments in the Coq proof assistant

[Luo11b]. Much more work is needed in this respect to achieve its practical goals.

Acknowledgement Thanks to the anonymous referees for very useful comments and to the members of the Type Theory and Applications research group at RHUL for helpful discussions.

References

[Agd08] The Agda proof assistant (version 2). Available from the web page: `http://appserv.cs.chalmers.se/users/ulfn/wiki/agda.php`, 2008.

[AP05] Nicholas Asher and James Pustejovsky. Word meaning and commonsense metaphysics, 2005.

[Ash08] Nicholas Asher. A type driven theory of predication with complex types. *Fundamenta Informaticae*, 84(2):151–183, 2008.

[Ash11] Nicholas Asher. *Lexical Meaning in Context: A Web of Words*. Cambridge University Press, 2011.

[Bai99] Anthony Bailey. *The Machine-checked Literate Formalisation of Algebra in Type Theory*. PhD thesis, University of Manchester, 1999.

[CF58] Haskell B. Curry and Robert Feys. *Combinatory Logic*, volume 1. North Holland Publishing Company, 1958.

[CH88] Thierry Coquand and Gerard Huet. The calculus of constructions. *Information and Computation*, 76(2/3):95–120, 1988.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(1):56–68, 1940.

[CL01] Paul Callaghan and Zhaohui Luo. An implementation of LF with coercive subtyping and universes. *Journal of Automated Reasoning*, 27(1):3–27, 2001.

[Coo07] Robin Cooper. Copredication, dynamic generalized quantification and lexical innovation by coercion. *Proceedings of GL2007, the Fourth International Workshop on Generative Approaches to the Lexicon*, 2007.

[Coo11] Robin Cooper. Copredication, quantification and frames. *Logical Aspects of Computational Linguistics (LACL'2011). LNAI 6736*, 2011.

[Coq07] The Coq Development Team. *The Coq Proof Assistant Reference Manual (Version 8.1), INRIA*, 2007.

[Dum75] Michael Dummett. The philosophical basis of intuitionistic logic. In H. Rose and J. Shepherdson, editors, *Proc. of the Logic Colloquium, 1973*. North Holland, 1975. Reprinted in P. Benacerraf and H. Putnam (eds.), Philosophy of Mathematics: selected readings, Cambridge University Press.

[Dum91] Michael Dummett. *The Logical Basis of Metaphysics*. Duckworth, 1991.

[Fef05] Soloman Feferman. Predicativity. In S. Shapiro, editor, *The Oxford Handbook of Philosophy of Mathematics and Logic*. Oxford Univ Press, 2005.

[Gea62] Peter Geach. *Reference and Generality*. Cornell University Press, 1962.

[How80] William A. Howard. The formulae-as-types notion of construction. In J. Hindley and J. Seldin, editors, *To H. B. Curry: Essays on Combinatory Logic*. Academic Press, 1980.

[LC98] Zhaohui Luo and Paul Callaghan. Coercive subtyping and lexical semantics (extended abstract). *Logical Aspects of Computational Linguistics (LACL'98)*, 1998.

[LP92] Zhaohui Luo and Robert Pollack. LEGO Proof Development System: User's Manual. LFCS Report ECS-LFCS-92-211., Department of Computer Science, University of Edinburgh, 1992.

[LSX12] Zhaohui Luo, Sergei Soloviev, and Tao Xue. Coercive subtyping: theory and implementation. *Information and Computation*, 223 (2013), 2012.

[Luo94] Zhaohui Luo. *Computation and Reasoning: A Type Theory for Computer Science*. Oxford University Press, 1994.

[Luo97] Zhaohui Luo. Coercive subtyping in type theory. *Computer Science Logic 1996, Lecture Notes in Computer Science 1258*, 1997.

[Luo99] Zhaohui Luo. Coercive subtyping. *Journal of Logic and Computation*, 9(1):105–130, 1999.

[Luo05] Yong Luo. *Coherence and Transitivity in Coercive Subtyping*. PhD thesis, University of Durham, 2005.

[Luo10] Zhaohui Luo. Type-theoretical semantics with coercive subtyping. *Semantics and Linguistic Theory 20 (SALT20), Vancouver*, 2010.

[Luo11a] Zhaohui Luo. Adjectives and adverbs in type-theoretical semantics. Notes, 2011.

[Luo11b] Zhaohui Luo. Contextual analysis of word meanings in type-theoretical semantics. *Logical Aspects of Computational Linguistics (LACL'2011). LNAI 6736*, 2011.

[Luo11c] Zhaohui Luo. Type-theoretical semantics with coercive subtyping. Lecture notes at ESSLLI 2011 (for the course on Lexical Semantics, taught together with Nicholas Asher), Ljubljana, Slovenia, 2011.

[Luo12] Zhaohui Luo. Common nouns as types. *Logical Aspects of Computational Linguistics (LACL'2012). LNAI 7351*, 2012.

[Mar07] Renaud Marlet. When the generative lexicon meets computational semantics. In *Proc of the 4th Inter. Workshop on Generative Approaches to the Lexicon (GL 2007)*, 2007.

[Mat08] The Matita proof assistant. `http://matita.cs.unibo.it/`, 2008.

[ML84] Per Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

[ML96] Per Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1), 1996.

[Mon74] Richard Montague. *Formal Philosophy*. Yale University Press, 1974. (Collection edited by R. Thomason).

[MS88] Marc Moens and Mark Steedman. Temporal ontology and temporal reference. *Comput. Ling.*, 14:15–28, 1988.

[NPS90] Bengt Nordström, Kent Petersson, and Jan Smith. *Programming in Martin-Löf's Type Theory*. Oxford University Press, 1990.

[Nun95] Geoffrey Nunberg. Transfers of meaning. *Journal of Semantics*, 12(2):109–132, 1995.

[Par86] Barbara Partee. Noun phrase interpretations and type-shifting principles. In J. Groenendijk, D. de Jongh, and M. Stokhof, editors, *Studies in Discourse Representation Theory and the Theory of Generalised Quantifiers*, 1986.

[Par10] B. Partee. Privative Adjectives: Subsective plus Coercion. In R. Bauerle, U. Reyle, and T. Zimmermann, editors, *Presuppositions and Discourse: Essays Offered to Hans Kamp*, volume 21 of *Current Research in Semantics/Pragmatics Interface*. Emerald Group Publishing Ltd., 2010.

[PR83] Barbara Partee and Mats Rooth. Generalised conjunction abd type ambiguity. In Bauerle, Schwarze, and von Stechow, editors, *Meaning, Use, and Interpretation of Language*, 1983.

[Pra74] Dag Prawitz. On the idea of a general proof theory. *Synthese*, 27, 1974.
[Pus95] James Pustejovsky. *The Generative Lexicon*. MIT Press, 1995.
[Pus05] James Pustejovsky. A survey of dot objects. Manuscript, 2005.
[Pus11] James Pustejovsky. Mechanisms of coercion in a general theory of selection, 2011.
[Ran94] Aarne Ranta. *Type-Theoretical Grammar*. Oxford University Press, 1994.
[Saï97] Amokrane Saïbi. Typing algorithm in type theory with inheritance. *Proceedings of Principls of Programming Languages 1997*, 1997.
[SL02] Sergei Soloviev and Zhaohui Luo. Coercion completion and conservativity in coercive subtyping. *Annals of Pure and Applied Logic*, 113(1-3):297–322, 2002.
[XL12] Tao Xue and Zhaohui Luo. Dot-types and their implementation. *Logical Aspects of Computational Linguistics (LACL'2012). LNAI 7351*, 2012.