Proceedings of SALT Volume 20, 1–15, 2010
doi: (no doi yet)

Type-Theoretical Semantics with Coercive Subtyping

Zhaohui Luo
Royal Holloway, Univ of London

Abstract In the formal semantics based on modern type theories, common nouns are interpreted as types, rather than as functional subsets of entities as in Montague grammar. 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 functional subsets. The theory of coercive subtyping adequately extends the modern type theories with a notion of subtyping and, as shown in this paper, plays a very useful role in making type theories more expressive for formal semantics. In particular, it gives a satisfactory treatment of the type-theoretic interpretation of modified common nouns and allows straightforward interpretations of interesting linguistic phenomena such as copredication, whose interpretations have been found difficult in a Montagovian setting. We shall also study some type-theoretic constructs that provide useful representational tools for formal lexical semantics, including how the so-called dot-types for representing logical polysemy may be expressed in a type theory with coercive subtyping.

Keywords: Coercive Subtyping, Formal Semantics, Lexical Semantics, Type-Theoretical Semantics, Type Theory

1 Introduction

Church's simple type theory (Church 1940), as employed in Montague grammar (Montague 1974), has traditionally served as a logical language for formal semantics. Powerful alternatives, arguably more advantageous ones, may be offered by the modern type theories such as Martin-Löf's type theory (Nordström, Petersson & Smith 1990; Martin-Löf 1984) and UTT (Luo 1994). By a *type-theoretical semantics*, we mean a formal semantics based on a modern type theory. The theory of *coercive subtyping* (Luo 1997, 1999) 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 more expressive for formal semantics.[1]

In linguistic semantics, there have been a lot of interesting developments including, for example, the Generative Lexicon Theory (Pustejovsky 1995) in lexical

1 The idea of using coercive subtyping in linguistic semantics was considered in (Luo & Callaghan 1998). Part of the current paper may be seen as a further development of those initial ideas.

semantics. However, the research so far has failed to provide a satisfactory formal account to explain the important linguistic phenomena in the lexical theories. Most of the employed formalisms are based on (extensions of) Montague grammar and unable to capture the linguistic phenomena satisfactorily. This paper studies type-theoretical semantics and shows that modern type theories, together with the theory of coercive subtyping, may offer a powerful language in which interesting lexical phenomena such as logical polysemy (Pustejovsky 1995) and copredication (Asher 2009) can be properly interpreted.

This paper also studies some type-theoretic constructs that provide useful representational tools for formal lexical semantics. We shall study how the *dot-types*, as proposed by Pustejovsky (1995) in studying logical polysemy in lexical semantics, can be formally expressed in a type theory with coercive subtyping. Also studied, though briefly, are *coercion contexts* for representing linguistic phenomena such as reference transfers in local contexts.

In Section 2, we give a brief introduction to modern type theories and the theory of coercive subtyping and discuss several relevant issues to introduce the background and some notational conventions. The use of coercive subtyping in type-theoretical semantics is discussed in Section 3. A formal treatment of the dot-types is given in Section 4, followed by a brief discussion in Section 5 on the representation of local coercions by coercion contexts in type theory.

2 Modern Type Theories and Coercive Subtyping

2.1 Modern Type Theories

Modern type theories may be classified into the predicative type theories such as Martin-Löf's type theory (Nordström, Petersson & Smith 1990; Martin-Löf 1984) and the impredicative type theories such as the Calculus of Constructions (CC) (Coquand & Huet 1988) and the Unifying Theory of dependent Types (UTT) (Luo 1994). In computer science, modern type theories have been implemented in the proof assistants such as Agda (Agda 2008) and Coq (Coq 2007) and used in applications to formalisation of mathematics and verification of programs.

We assume that the reader be familiar with the simple type theory (Church 1940) (or Montague grammar (Montague 1974)), compared with which modern type theories have several distinctive features that are briefly described below.

Dependent Types: Π-types and Σ-types. Modern type theories contain *dependent types*. An example is the so-called Σ-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, if Man is the

type of men and *handsome* is a predicate over men (this is a *Man*-indexed family of propositions/types), then $\Sigma(Man, handsome)$ 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 the 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 and others (such as Dummett and Prawitz 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.[2] 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 are sometimes 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:[3]

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

Embedded Logic. According to the propositions-as-types principle (Curry & Feys 1958; Howard 1980), a type theory may have an *embedded logic* (or sometimes

2 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'.

3 Being an essential property, canonicity has been considered by Martin-Löf since the early days of development of his type theory. However, as far as the author is aware, the term 'canonicity' appeared in (Altenkirch, McBride & Swierstra 2007) for the first time.

called the *internal logic*). For example, the logical proposition $A\&B$ corresponds to the product type $A \times B$ and a pair of a proof of A and a proof of B corresponds to a 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.

The embedded logic in the simple type theory (Church 1940) is the higher-order predicate logic. For Martin-Löf's type theory, the embedded logic is first-order[4] and, for impredicative type theories, 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 or 'impredicative', in the technical jargon.)

Remark *Prop* is very much like the type t in the simple type theory. The only difference is that, in modern type theories, we have explicit proof terms of logical propositions. In this paper, we shall use *Prop* in linguistic interpretations. For instance, a verb is interpreted as a predicate of type $A \rightarrow Prop$, where A is the type of entities to which the verb can be meaningfully applied. □

2.2 Coercive Subtyping

Coercive subtyping (Luo 1997, 1999) 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 (Coq 2007; Saïbi 1997), Lego (Luo & Pollack 1992; Bailey 1999), Matita (Matita 2008) and Plastic (Callaghan & Luo 2001), 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 in 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 $\mathfrak{C}_B[c(a)]$.

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

4 Please note that the embedded logic in Martin-Löf's type theory is not the ordinary first-order predicate logic. In particular, the existential quantifier is expressed by the 'strong' Σ-types, not the weak existential types, which only exist in impredicative type theories, not in Martin-Löf's type theory.

that `shout` be interpreted as $[\![\texttt{shout}]\!] : \mathit{Human} \to \mathit{Prop}$ and `John` as $[\![\texttt{John}]\!] : \mathit{Man}$, then the interpretation (1') of the following sentence (1) is well-typed:

(1) John shouts.

(1') $[\![\texttt{shout}]\!]([\![\texttt{John}]\!])$

The reason that (1') is well-typed is because that *Man* is now a subtype of *Human*. In general, this is reflected by the following *coercive definition rule*:

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

expressing that an appropriate coercion can be inserted to fill up the gap in a term. (See (Luo 1999) for a formal presentation with complete rules.)

Notation We shall adopt the following notational abbreviations, writing

- $A < B$ for '$A <_c B : \mathit{Type}$ for some c', and
- $A \leq B$ for '$A = B : \mathit{Type}$ 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 the notion of *subsumptive subtyping* characterised by means of the following *subsumption rule*:

$$(*) \qquad \frac{a : A \qquad A \leq B}{a : B}$$

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 subtyping relations and, as a consequence, the subsumptive rule $(*)$ would introduce inconsistency in a modern type theory.[5]

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

5 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.

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 Montague grammar. 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[6] (the proof method in (Soloviev & Luo 2002) can be used to show this). 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.

3 Coercive Subtyping in Type-Theoretical Semantics

In type-theoretical semantics, common nouns are interpreted as types (Ranta 1994), rather than as functional subsets of entities as in Montague grammar. (See below for more explanations on this.) This brings about important advantages in linguistic interpretations but has some unwelcome consequences because there are fewer operations on types as compared with those on functional subsets.

For example, one can easily define a subset relation between the functional sets 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 grammar. However, such a notion between types is not that straightforward. As we have discussed in Section 2.2, the traditional notion of subtyping, subsumptive subtyping, is incompatible with the notion of canonical object

6 As coercions may be declared by the users, they must be *coherent* to be employed correctly (and to guarantee conservativity). Essentially, coherence requires that the coercions between any two types be unique. See (Luo 1999) for a formal definition.

in modern type theories and therefore cannot be employed for a type-theoretical semantics. Instead, the theory of coercive subtyping (Luo 1999) 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 a Montagovian setting.

Type-Theoretical Semantics: Common Nouns as Types. Some of the basic ideas of developing formal semantics in a modern type theory have been studied by Ranta (1994), where various semantic issues of natural languages have been studied in Martin-Löf's type theory.[7] To compare a type-theoretical semantics with Montague grammar, one of the most basic differences is between the interpretations of common nouns. In Montague grammar, common nouns like `man` and `human` are interpreted as functional subsets (or predicates) of entities, i.e., as objects of type $e \to t$, where e is the type of entities and t the type of propositions. In a type-theoretical semantics based on modern type theories, on the other hand, common nouns are interpreted as *types*. For instance, the interpretations of `man`, `human` and `book` are types:

$$[\![\texttt{man}]\!],\ [\![\texttt{human}]\!],\ [\![\texttt{book}]\!] : \mathit{Type}.$$

This is natural in a modern type theory, which is 'many-sorted' in the sense that there are many types like $[\![\texttt{man}]\!]$ and $[\![\texttt{book}]\!]$ consisting of objects standing for different sorts of entities, while the simple type theory may be thought of as 'single-sorted' in the sense that there is the type e of all entities.

In a type-theoretical semantics, verbs and adjectives are interpreted as predicates. For example, we have

$$\begin{aligned} [\![\texttt{heavy}]\!] &: [\![\texttt{book}]\!] \to \mathit{Prop} \\ [\![\texttt{read}]\!] &: [\![\texttt{human}]\!] \to [\![\texttt{book}]\!] \to \mathit{Prop} \end{aligned}$$

where *Prop* is the type of propositions (see Section 2.1). Modified common noun phrases are interpreted by means of Σ-types of dependent pairs: for instance,

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

7 Ranta himself may not regard his work as studying logical semantics (see, for example, the preface of (Ranta 1994).) 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. Here, we are not going to conduct a full-scale evaluation of the current status of type-theoretical semantics, but only mention some basic ideas that will be relevant for our discussion.

Remark (meaningfulness) To interpret common nouns as types in a 'many-sorted' setting has its advantage in that it effectively distinguishes *meaningless* and *false* expressions. For example, the verb `talk` can be given the following interpretations:

(2) In Montague grammar: $[\![\texttt{talk}]\!]_M : e \to t$

(3) In type-theoretical semantics: $[\![\texttt{talk}]\!]_T : [\![\texttt{human}]\!] \to \mathit{Prop}$

Now consider the following sentence (4), which is given interpretation (5) in the Montague grammar and (6) in the type-theoretical semantics:

(4) A table talks.

(5) $\exists x : e.\ [\![\texttt{table}]\!]_M(x)\ \&\ [\![\texttt{talk}]\!]_M(x)$

(6) $\exists t : [\![\texttt{table}]\!]_T .\ [\![\texttt{talk}]\!]_T(t)$

where $[\![\texttt{table}]\!]_M : e \to t$ is a predicate while $[\![\texttt{table}]\!]_T$ is a type. Now, the term (5) is well-typed (and false), while the term (6) is simply not well-typed, i.e., meaningless. We contend that, in this respect, the type-theoretical semantics captures the meanings in a better way: the sentence (4) is usually regarded as meaningless (unless in some fictional world), as in the type-theoretical semantics. □

Coercive Subtyping: Basic Applications As explained above, in a type-theoretical semantics, common nouns are interpreted as types and, in particular, modified common nouns as Σ-types. We have, for instance,

$$\begin{array}{rcl} [\![\texttt{John}]\!] & : & [\![\texttt{man}]\!] \\ [\![\texttt{W\&P}]\!] & : & [\![\texttt{heavy book}]\!] = \Sigma([\![\texttt{book}]\!], [\![\texttt{heavy}]\!]) \end{array}$$

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

(7) John reads a book.

(8) Somebody reads 'War and Peace'.

(9) John reads 'War and Peace'.

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

(7') $\exists b : [\![\texttt{book}]\!] .\ [\![\texttt{read}]\!]([\![\texttt{John}]\!], b)$.

(8') $\exists h : [\![\texttt{human}]\!] . [\![\texttt{read}]\!](h, [\![\texttt{W\&P}]\!])$.

(9') $[\![\texttt{read}]\!]([\![\texttt{John}]\!], [\![\texttt{W\&P}]\!])$.

But none of the above three terms is well-typed. $[\![\texttt{read}]\!]$ requires its first argument to be of type $[\![\texttt{human}]\!]$ and its second of type $[\![\texttt{book}]\!]$, but $[\![\texttt{John}]\!]$ is of type $[\![\texttt{man}]\!]$ and $[\![\texttt{W\&P}]\!]$ is of type $[\![\texttt{heavy book}]\!]$. 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 $[\![\texttt{man}]\!]$ is a subtype of $[\![\texttt{human}]\!]$:

$$[\![\texttt{man}]\!] < [\![\texttt{human}]\!] .$$

For the second case, we have

$$[\![\texttt{heavy book}]\!] < [\![\texttt{book}]\!],$$

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,

$$[\![\texttt{human}]\!] \to [\![\texttt{book}]\!] \to \mathit{Prop} < [\![\texttt{man}]\!] \to [\![\texttt{heavy book}]\!] \to \mathit{Prop}.$$

With these subtyping relations, the terms in (7'-9') can now be well-typed and they interpret the sentences in (7-9), respectively.

Remark The above problem is discussed in (Ranta 1994) (pp. 62-64), where it is called the problem of 'multiple categorization of verbs' (in our case, the verb 'read') and three possible solutions are considered, but none of them is completely satisfactory. One of them is closest to ours where explicit first projections are employed; it is one step short: using π_1 as an implicit coercion, we have managed to capture the phenomena as intended. □

As another example, let's consider the dot-types as studied in (Pustejovsky 1995).[8] 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: $[\![\texttt{book}]\!] <$ PHY • INFO. By contravariance,

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

Therefore, for example, for $[\![\texttt{burn}]\!] : \text{PHY} \to \mathit{Prop}$ and $[\![\texttt{boring}]\!] : \text{INFO} \to \mathit{Prop}$, the following phrase (10) can be interpreted by term (10'), as intended:[9]

(10) `burn a boring book`

(10') $\exists b : \Sigma([\![\texttt{book}]\!], [\![\texttt{boring}]\!]).\ [\![\texttt{burn}]\!](b)$

Remark In Montague grammar (and its extensions), common nouns are interpreted as functional subsets of type $e_0 \to t$, where e_0 is a subtype of the type e of entities. For instance, $[\![\texttt{book}]\!] : \text{PHY} \bullet \text{INFO} \to t$ and $[\![\texttt{heavy}]\!] : (\text{PHY} \to t) \to (\text{PHY} \to t)$. In such a situation, in order to interpret, e.g., '`a heavy book`', one would have to apply $[\![\texttt{heavy}]\!]$ to $[\![\texttt{book}]\!]$ by requiring, for example, PHY • INFO $\to t$ to be a subtype of PHY $\to t$, which is not the case – type clashes would happen, leading to unnatural and complicated treatments (Asher 2008). □

Copredication. The final example we shall use to illustrate the use of coercive subtyping is the interpretation of copredication. Copredication has been studied by Asher (2009), where the following example is considered:

(11) 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:

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

8 Dot-types will formally be studied in Section 4. Here, we study the example informally.
9 Thanks for an anonymous referee for the suggestion of the word 'boring'.

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

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

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

In other words, $[\![\texttt{pick up}]\!]$ and $[\![\texttt{master}]\!]$ can both be used in contexts where terms of type $[\![\texttt{human}]\!] \to [\![\texttt{book}]\!] \to \mathit{Prop}$ are required and, therefore, the interpretation of the sentence (11) can proceed straightforwardly as intended.

Remark In a Montagovian setting, the interpretations of such sentences with copredication can become rather sophisticated (see, for example, (Asher & Pustejovsky 2005)). This is because, in Montague grammar, common nouns are interpreted as functional subsets. Such an interpretation seems incompatible with the subtyping relationships involving PHY and INFO. In a type-theoretical semantics 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

In the type theory with coercive subtyping, several useful constructions can be defined and used to model various linguistic phenomena. In this section, we study how the so-called dot-types may be represented.

Dot-types, or sometimes called dot objects or complex types, are proposed by Pustejovsky in his Generative Lexicon Theory (Pustejovsky 1995). Although the meaning of a dot-type is intuitively clear, its proper formal account seems surprisingly difficult and tricky.[10] Researchers have made several proposals to model dot-types including, for example, (Asher & Pustejovsky 2005) and (Cooper 2007). 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

10 See, for example, (Asher 2008) 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.

help of coercive subtyping[11] which, we believe, gives an adequate formal account of dot-types and can hence be used in a type-theoretical semantics to interpret, for instance, copredication (as in Section 3) and logical polysemy as studied in (Pustejovsky 1995).

In Section 3, we have already introduced the dot-types PHY • INFO informally with examples. An important feature of the dot-type 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.* (Pustejovsky 2005)

Put in another way, 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.

Definition 4.1 (components) *Let $T : Type$ be a type in the empty context. Then, $\mathscr{C}(T)$, the set of components of T, is defined as:*

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

where $\text{SUP}(T) = \{T' \mid T \leq T'\}$. □

The rules for the dot-types are given in Figure 1. Note that, in the formation rule, we require that the constituent types do not share common components: $\mathscr{C}(A) \cap \mathscr{C}(B) = \emptyset$. 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 two important differences between dot-types and product types:

11 The idea of using coercive subtyping to model dot-types was considered in (Luo & Callaghan 1998) where, however, the product types and the associated projection coercions were proposed; this is known to be incoherent from Y. Luo's PhD thesis (Luo 2005).

Formation Rule

$$\frac{A : Type \quad B : Type \quad \mathscr{C}(A) \cap \mathscr{C}(B) = \emptyset}{A \bullet B : Type}$$

Introduction Rule

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

Elimination Rules

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

Computation Rules

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

Projections as Coercions

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

Coercion Propagation

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

where $d_1[c_1](\langle a, b \rangle) = \langle c_1(a), b \rangle$.

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

where $d_2[c_2](\langle a, b \rangle) = \langle a, c_2(b) \rangle$.

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

where $d[c_1, c_2](\langle a, b \rangle) = \langle c_1(a), c_2(b) \rangle$.

Figure 1 The rules for dot-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 4.2 below). For product types, however, only one of them can be coercions for, otherwise, coherence would fail (Luo 2005).

According to the rules in Figure 1, $A \bullet B$ 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 Section 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 1.

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 4.2 (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 operator 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 $\textsc{Phy}$. If $\textsc{Phy} \bullet (\textsc{Phy} \bullet \textsc{Info})$ were a dot-type, there would be the following two coercions p_1 and $p_1 \circ p_2$:

$$\begin{array}{lll} \textsc{Phy} \bullet (\textsc{Phy} \bullet \textsc{Info}) & <_{p_1} & \textsc{Phy} \\ \textsc{Phy} \bullet (\textsc{Phy} \bullet \textsc{Info}) & <_{p_1 \circ p_2} & \textsc{Phy} \end{array}$$

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

Finally, it is worth remarking that one may consider a formal treatment of dot-types by means of the so-called (non-dependent) record types, which are informally labelled product types (Pollack 2002; Luo 2009b,a). Then, the condition that the constituent types do not share common components can be achieved by requiring that the (top-level) labels of the constituent types be distinct. Furthermore, the record projections can be taken as coercions (see (Luo 2009b) for details, where the coherence of these coercions is discussed in a different context.)

5 Coercions in Local Contexts

In this section, we briefly discuss *coercion contexts*, which are useful for representing special phenomena such as reference transfers in local contexts.

In the use of natural language, many usages are only meaningful in special situations or local contexts in which, for instance, the meanings of some words change (cf., reference transfers as discussed by Jackendoff (1997)). Consider the following utterance by a waiter in a café:

(12) The ham sandwich shouts.

Let us assume that the act of shouting require that the argument be human, i.e., $[\![\texttt{shout}]\!] : [\![\texttt{human}]\!] \to \mathit{Prop}$. But `ham sandwich` clearly is not human and there is nothing about the semantics which suggests the required subtyping relation: $[\![\texttt{ham sandwich}]\!] < [\![\texttt{human}]\!]$. This suggests that, in certain local contexts, it is acceptable to distinguish entities on the basis of some salient property. Here, it is the property of having ordered a ham sandwich, in the environment of a café etc. This is clearly dependent on the extralinguistic context of an utterance and, in particular, there are contexts where this kind of coercion is invalid.

In the framework of coercive subtyping, coercions have only been considered as global in the sense that they are applicable in all valid contexts (see, for example, (Luo 1999) for formal details). As suggested by the above example, to apply coercive subtyping in formal semantics, a notion of coercion context is called for in the type-theoretic framework, with which coercions can be introduced, and only valid, in a specific context.

A *coercion context* is a context whose entries may be of the form $A <_c B$ as well as the usual entries of the form $x : A$. Formally, this involves the following rules:

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

where $(A)B$ is the functional kind from A to B in the logical framework (see Chapter 9 of (Luo 1994) 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, we may introduce $[\![\texttt{ham sandwich}]\!] < [\![\texttt{human}]\!]$ in a particular local context and then the above sentence (12) can be interpreted satisfactorily.

Remark (coherent context) Please note that the 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. □

Coercions may be introduced into terms (expressions in the type theory), by means of 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 forms of judgement (e.g., J may be $k : K$) and the key word **coercion** distributes through J. For example, the judgements **coercion** $A <_c B$ **in** $(a : A)$ and $(\textbf{coercion}\ A <_c B\ \textbf{in}\ a)\ :\ (\textbf{coercion}\ A <_c B\ \textbf{in}\ A)$ are identified.

In this paper, we content ourselves with the above informal introduction to coercion contexts and claim that the notion supports the intended applications of subtyping in local contexts. A formal development is left for a future paper.

6 Conclusion

In this paper, we have studied the application of modern type theories to formal semantics and shown that the theory of coercive subtyping plays a useful role in enriching modern type theories to become powerful formal languages for type-theoretical semantics. It will be interesting to conduct a full-scale study of type-theoretical semantics, starting with, for example, an evaluation of the proposals made by Ranta (1994). One may compare carefully the semantic interpretations in the Montagovian setting and those possible in the type-theoretical semantics and this is expected to lead to the further interesting development in type-theoretical semantics.

As to concrete proposals for linguistic interpretations, we should mention that there have been various proposals for the formalisation of structured lexical entries as studied by Pustejovsky (1995). In particular, Cooper (2005) has proposed to use dependent record *kinds* in Martin-Löf's type theory to represent lexical entries. Note that these record kinds are *not* types, as we have talked about in this paper – in the type theory they are at the same level as other kinds like *Type*, the kind of all types. This would prohibit them from, for example, being combined with other types to form new types. Dependent record types are studied in, for example, (Pollack 2002; Luo 2009b,a). Cooper's work has shown that record types can be employed to represent lexical entries and it is very interesting to see how this may be combined with other type constructs (eg, dot-types and their record-type formalisation as mentioned at the end of Section 4) in type-theoretical semantics.

There have been a lot of work to study formal or semi-formal accounts of the Generative Lexicon Theory (Pustejovsky 1995) (see (Bassac, Mery & Retoré 2009) for an interesting and rather comprehensive summary of the work so far). It is unclear what, if any, formalism may be employed to capture the essential aspects of

the GL theory. It would be interesting to see how far one may go to model the GL theory in a type-theoretical semantics.

Acknowledgement The author wants to thank Nicholas Asher and the anonymous referees for their comments on the abstract of the paper.

References

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

Altenkirch, T., C. McBride & W. Swierstra. 2007. Observational Equality, Now! *Proceedings of the 2007 Workshop on Programming Languages Meets Program Verification (PLPV 2007)* .

Asher, N. 2008. A type driven theory of predication with complex types. *Fundamenta Infor.* 84(2).

Asher, N. 2009. *Lexical Meaning in Context: A Web of Words*. (Draft book to be published by CUP).

Asher, N. & J. Pustejovsky. 2005. Word meaning and commonsense metaphysics.

Bailey, A. 1999. *The machine-checked literate formalisation of algebra in type theory*: University of Manchester dissertation.

Bassac, C., B. Mery & C. Retoré. 2009. Towards a Type-theoretical account of lexical semantics. *J of Logic, Language and Information* 19(2).

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

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

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

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

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

Coquand, Th. & G. Huet. 1988. The calculus of constructions. *Information and Computation* 76(2/3).

Curry, H.B. & R. Feys. 1958. *Combinatory logic*, vol. 1. North Holland Publishing Company.

Howard, W. A. 1980. The formulae-as-types notion of construction. In J. Hindley & J. Seldin (eds.), *To H. B. Curry: Essays on Combinatory Logic*, Academic Press.

Jackendoff, R. 1997. *The Architecture of the Language Faculty*. MIT.
Luo, Y. 2005. *Coherence and transitivity in coercive subtyping*: University of Durham dissertation.
Luo, Z. 1994. *Computation and Reasoning: A Type Theory for Computer Science*. Oxford University Press.
Luo, Z. 1997. Coercive subtyping in type theory. *CSL'96, LNCS'1258* .
Luo, Z. 1999. Coercive subtyping. *J of Logic and Computation* 9(1).
Luo, Z. 2009a. Dependent record types revisited. *Proc. of the 1st Inter. Workshop on Modules and Libraries for Proof Assistants (MLPA'09), Montreal.* ACM Inter. Conf. Proceeding Series, Vol 429.
Luo, Z. 2009b. Manifest fields and module mechanisms in intensional type theory. *Types for Proofs and Programs (TYPES'08), LNCS 5497.* .
Luo, Z. & P. Callaghan. 1998. Coercive subtyping and lexical semantics (extended abstract). *Logical Aspects of Computational Linguistics (LACL'98)* .
Luo, Z. & R. Pollack. 1992. LEGO Proof Development System: User's Manual. LFCS Report ECS-LFCS-92-211. Dept of Computer Science, Univ of Edinburgh.
Martin-Löf, P. 1984. *Intuitionistic type theory*. Bibliopolis.
Matita 2008. 2008. The Matita proof assistant. `http://matita.cs.unibo.it/`
Montague, R. 1974. *Formal philosophy*. Yale University Press. (Collection edited by R. Thomason).
Nordström, B., K. Petersson & J. Smith. 1990. *Programming in Martin-Löf's type theory*. Oxford University Press.
Pollack, R. 2002. Dependently typed records in type theory. *Formal Aspects of Computing* 13(3-5).
Pustejovsky, J. 1995. *The Generative Lexicon*. MIT.
Pustejovsky, J. 2005. A survey of dot objects. Manuscript.
Ranta, A. 1994. *Type-theoretical grammar*. Oxford University Press.
Saïbi, A. 1997. Typing algorithm in type theory with inheritance. *POPL'97* .
Soloviev, S. & Z. Luo. 2002. Coercion completion and conservativity in coercive subtyping. *Annals of Pure and Applied Logic* 113(1-3).

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