Coercive subtyping and lexical semantics*
(Extended Abstract)

Zhaohui Luo and Paul Callaghan
Department of Computer Science, University of Durham
{Zhaohui.Luo, P.C.Callaghan}@durham.ac.uk

1 Introduction

This paper investigates the use of constructive type theory in lexical semantics. Our intention is to explore how a rich language of types with subtyping can be used to express lexical knowledge, both as an application of type theory and as an alternative to current approaches. In particular, we show that coercive subtyping [Luo97, Luo98a], provides a formal framework with useful mechanisms for lexical semantics.

Coercive subtyping extends constructive type theories (eg, Martin-Löf's intensional type theory [NPS90] and the type theory UTT [Luo94]) with a simple abbreviational mechanism. It provides elegant and flexible means of representing inheritance and overloading. In our earlier paper on the structure of Mathematical Vernacular [LC98], coercive subtyping is used to represent the inheritance relationships between mathematical concepts and to model some common abbreviations used in mathematical language.

Building on some of the ideas in [LC98], we consider in this paper how coercive subtyping can be applied in modelling of word meanings in natural language and their semantic compositions. We show that the abbreviational mechanisms of coercive subtyping provide interesting and useful tools to represent and analyse different modes of semantic composition that involve type shifting, reference transfer, and multiple sense selection. Relating the use of coercive subtyping to Pustejovsky's work on the Generative Lexicon Theory [Pus95] and Jackendoff's work on Enriched Composition [Jac97], we discuss how some forms of logical polysemy and enriched composition may be analysed in this framework.

The motivation of this work is twofold. Firstly, it is our hope that the use of coercive subtyping will provide lexical semantic studies with useful tools based on a solid proof-theoretic (or type-theoretic) foundation. Secondly, we believe that this will shed light on how lexical semantics of full mathematical vernacular should be formally understood.

2 Coercive subtyping in constructive type theories

We give a brief and informal introduction to coercive subtyping. For more detailed presentation and explanation, see [Luo97, Luo98a], and for the related proof-theoretic results, see [JLS97, SL98].

The basic idea and the coercion mechanism The basic idea of coercive subtyping is to consider subtyping as an abbreviational mechanism. A is a subtype of B, notation $A \leq B$, if either $A = B$ (computational equality) or A is a proper subtype of B such that there is a unique implicit coercion c from A to B (in this latter case, we use the judgement form $A <_c B$).

The key proof-theoretic mechanism to realise coercive subtyping is the following *coercive application rule* and *coercive definition rule*, where $(x{:}K)K'[x]$ is a dependent functional kind in a typed logical framework for specifying type theories (see [Luo94]):

$$\frac{\Gamma \vdash f : (x{:}K)K'[x] \quad \Gamma \vdash k_0 : K_0 \quad \Gamma \vdash K_0 <_c K}{\Gamma \vdash f(k_0) : K'[c(k_0)]} \qquad \frac{\Gamma \vdash f : (x{:}K)K'[x] \quad \Gamma \vdash k_0 : K_0 \quad \Gamma \vdash K_0 <_c K}{\Gamma \vdash f(k_0) = f(c(k_0)) : K'[c(k_0)]}$$

In general, if $A <_c B$, these rules allow one to use an object a of type A in a context $\mathcal{C}_B[\_]$ where an object of type B is expected: $\mathcal{C}_B[a]$ is well-typed (by the first rule), and furthermore, we have that $\mathcal{C}_B[a]$ is

*Logical Aspects of Computational Linguistics (LACL'98), Grenoble, Dec 1998. This work is supported partly by UK EPSRC project GR/K79130 (see http://www.dur.ac.uk/~dcs7ttg/sir.html) and partly by the Durham Mathematical Vernacular project funded by the Leverhulme Trust (see http://www.dur.ac.uk/~dcs7ttg/mv.html).

computationally equal to $\mathcal{C}_B[c(a)]$ (by the second rule). This gives the direct (computational) meaning of the abbreviation introduced by the coercion mechanism, and conforms with and coherently extends Martin-Löf's meaning theory for constructive type theories.

Generality and flexibility The power of coercive subtyping comes from its uniform generalisation of the traditional notions of subtyping and inheritance, including the inclusion-based subtyping (eg, between the types of men and humans), represented as injective coercions, and that of inheritance-based subtyping (eg, between record types), represented as projective coercions. In fact, any function from A to B can be specified as a coercion, as long as the coherence property holds for the overall system (see below). With the rich type structure in constructive type theory (eg, dependent function types, types of dependent pairs or Σ-types, and other inductive types), the simple idea of coercive subtyping provides surprisingly powerful mechanisms in a uniform way. Different forms of coercion mechanisms have been implemented in proof systems such as Lego [LP92] and Coq [Coq96], and have been successfully applied to large proof developments [Bai98] among other applications.

Coherence and conservativity Given the generality of possible coercions, one must set conditions so that the resulting type theory with coercive subtyping has nice proof-theoretic properties. A key condition is that of *coherence*, which says that there can be no computationally distinct coercions between any two types (ie, if $A <_c B$ and $A <_{c'} B$, then c and c' are computationally equal). An important meta-theoretic result about coercive subtyping is that, for coherent subtyping relations, the extension with coercive subtyping is a conservative extension of the original type theory (see [SL98] for a proof of this for subtyping between parameterised inductive types.)

Remark Although developed independently from completely different formal frameworks and motivations, coercive subtyping shares some basic ideas (cf, [BCGS91]) with, for example, Pustejovsky's use of coercions in his Generative Lexicon Theory [Pus95]. However, the techniques and representational mechanisms provided by coercive subtyping are much more general and flexible. As a theory of subtyping and inheritance for type theories with rich type structure and solid proof-theoretic foundation, it is extremely interesting to see how it can be applied in lexical semantics in relation to existing work.

3 Using coercive subtyping in lexical semantics

We assume that lexical semantics is described in a typed language. Semantic typing gives restrictions on (semantic) well-formedness of phrases and sentences. For example, the semantics of a class-denoting common noun (eg, `car`) is a type (eg, Car) and that of an adjective is a predicate over a domain type. Under the standard typing discipline (without subtyping), for example, the semantic composition of an adjective with a common noun requires that the domain of the adjective is the same as the type associated with the common noun. With the coercive subtyping mechanisms, when appropriate coercions are specified, semantic composition gives flexibility and produces the expected meaning constructions. In this section, we sketch how coercive subtyping may be used in modelling some phenomena of lexical semantics and explain the ideas with fairly standard examples.

3.1 Homonymy and multiple sense selection

We consider in this section semantic modelling of homonymy as an example to see how the coercion mechanism supports multiple sense selection (or in general, ad hoc polymorphism).

Homonymy – the simple cases First, consider homonymies whose different meanings have distinct semantic types (eg, `run`). In general, assume that there are n words in a (finite) type $W = \{w_1, ..., w_n\}$ and each word w_i has k_i unrelated meanings m_{ij} of types A_{ij} ($j = 1, ..., k_i$ and $k_i \geq 1$), and there is no type A such that $A_{ij_1} \leq A$ and $A_{ij_2} \leq A$ for any $j_1 \neq j_2$. For instance, one of the words may be `run` with two meanings $\texttt{run}_1$ of type $Person \to S$ and $\texttt{run}_2$ of type $Person \times Institution \to S$, where S is the type of propositions.

Now, let $Unit : (W)\mathbf{Type}$ be the family of unit types, where each $Unit(w_i)$ only contains (the name of) w_i as its object. Define $c(i,j) : (Unit(w_i))A_{ij}$ such that $c(i,j)(w_i) = m_{ij}$. Then the family of coercions c (a parameterised coercion) gives a description for the above sense selection model.

This description is adequate. In particular, in any context $\mathcal{C}[\_]$, if the required meaning of w_i is of type A_{ij}, then we have $\mathcal{C}[w_i] = \mathcal{C}[c(i,j)(w_i)] = \mathcal{C}[m_{ij}]$ (computational equality), as expected. For instance, we have by the coercive definition rule, `John runs quickly = John runs`$_1$ `quickly` and `John runs a bank = John runs`$_2$ `a bank`.

Homonymous common nouns Common nouns are given types as their meanings. A homonymous common noun (eg, `bank`) has several unrelated meanings, which cannot be distinguished in contexts solely by typing as above. The disambiguation is dependent on further linguistic information, such as reference to financial matters. This information is arbitrarily distributed (ie, it need not occur in the same sentence), so we need a mechanism for representing underspecification and for removing it when sufficient constraints appear.

First, we represent the meaning of a homonymous common noun as a family of types, each of which represents a particular meaning. For example, `bank` can be represented as a family of types $Bank(i)$, where $Bank(1)$ is the type of banks as financial institutions, $Bank(2)$ that of banks as river sides, etc. In general, if a common noun has different senses as types $W(i)$ indexed by i ranging over finite type I, the type family $W : (I)\mathbf{Type}$ gives the meaning of the common noun.[1]

An underspecified expression such as `a bank` can then be represented as a variable of type $W(m)$ (eg, $b : Bank(m)$), where $m{:}I$ is a meta-variable. This meta-variable represents the initial lack of knowledge as to which exact sense of Bank is intended; further constraints, such as use of b in a context where a financial Bank is required, will result in instantiation of the meta-variable with the appropriate index from I. For instance, assuming that `lend` takes an argument, among others, which must be of type $FInst$ (financial institution) and $Bank(1) < FInst$, then `a bank` in `A bank lends me some money`, can be interpreted as of type $Bank(1)$, with the meta-variable instantiated as 1 in this case.

3.2 Type-theoretic accounts of enriched composition

In natural language, there are many phenomena which cannot be analysed with a simple notion of composition (ie, using a simple typing discipline). For example, interpreting Adjective-Noun constructions as a predicate on the noun clearly produces wrong results in the examples below. Jackendoff introduces a notion of *enriched composition* when analysing such examples [Jac97]. It is related to Pustejovsky's uses of qualia structure and his notions of coercion and dot objects, as in Generative Lexicon Theory [Pus95]. We explain below how the flexible mechanisms of coercive subtyping helps to analyse the examples.

"fast car" Intuitively, the semantics of 'fast' is to indicate that some process is done quickly. We assume 'fast' is a predicate with domain *Process*. Therefore the adjective must act in some way on the semantic representation of the noun to determine which aspect of the noun is a process. We do not specify here how 'car' should be represented; at minimum, *Car* can be understood as a process, hence $Car <_c Process$. Under coercive subtyping, for any car $x{:}Car$ we have the computational equality $fast(x) = fast(c(x))$ ('x travels fast') and hence `fast car = car that travels fast`.

"ham sandwich" (reference transfer) Consider *The ham sandwich shouts*. The act of shouting provides a context where the argument is human, ie $shout{:}Human \rightarrow S$. Clearly, the literal referent of an expression is not the intended referent; a reference transfer is required. But "ham sandwich" clearly is not human and there is nothing about the semantics which suggests the required subtyping relation, $Ham\_sandwich <_c Human$.

In certain contexts, however, 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 context of a café etc. This is clearly dependent on the extralinguistic context of an utterance, so we introduce the notion of such coercions being part of a wider context. In particular, there are contexts where this kind of coercion is invalid. In this aspect, a notion of coercion context is called for in the type-theoretic framework. We shall not elaborate this here.

3.3 Pustejovsky's qualia structures and dot objects

Pustejovsky's "Generative Lexicon Theory" [Pus95] includes a notion of qualia structure to represent the semantics of a word so that logical polysemy can be dealt with via enriched composition. The qualia structures encode aspects of meaning, such as part-whole relationships, how something came in to being, and what the purpose of something is.

[1] One may give a type as the meaning for `bank` by integrating the types in the family W together into the disjoint union type $\cup_{i\in I} W(i)$. One can also declare a dependent coercion [Luo98b] of kind $(x{:}\cup_{i\in I} W(i))W(f(x))$, where $f(intro_i(w)) = i$, which would be useful in analysing phrases involving fully specified expressions such as `Barclays Bank`. However, this union type does not seem to be a logically accurate model of the unresolved homonymy.

Formalisation of qualia structure It is possible to formalise the notion of qualia structure (and the related components) in Pustejovsky's theory by means of Σ-types or record types in type theory, which provide the necessary sharing mechanisms. For instance, the semantics of `car` may be formalised as a record type. Then, the coercion from Car to $Process$ considered above can be defined as a function (rather than just as declared constant), which extracts the relevant process of travelling from the qualia structure of `car`. Similar examples that can be dealt with include `begin the book` etc.

Pustejovsky uses the language of Typed Feature Structures in his treatment of qualia structure etc. Usually, unification is the primary operation on such structures. Further research is needed to see whether unification is essential in a type-theoretic setting and if so, how the corresponding operations can be treated.

Dot objects Pustejovsky has also introduced a notion of dot object in his theory to explain cospecification. For example, `book` has at least two aspects – as information and as physical object; to read a book can be regarded as concerned with both, while understanding a book would select the information aspect. Therefore, one needs a semantic entity that encompasses both these aspects. In his work, Pustejovsky has only sketched (but did not precisely define) what a dot object is, though the motivation and the intended use is clear.

We believe that a more adequate account for the notion of dot object is the following. Consider two types A_1 and A_2. Then the so-called dot type is in fact the product type $A_1 \times A_2$ *together with* the two projections $\pi_i : A_1 \times A_2 \to A_i$ $(i = 1, 2)$ as coercions. More details of this with examples and its use in our setting will be discussed in the full paper.

References

[Bai98] A. Bailey. *The Machine-checked Literate Formalisation of Algebra in Type Theory.* PhD thesis, University of Manchester, 1998.

[BCGS91] V. Breazu-Tannen, T. Coquand, C. Gunter, and A. Scedrov. Inheritance and explicit coercion. *Information and Computation*, 93, 1991.

[Coq96] Coq. *The Coq Proof Assistant Reference Manual (version 6.1).* INRIA-Rocquencourt and CNRS-ENS Lyon, 1996.

[Jac97] R. Jackendoff. *The Architecture of the Language Faculty.* MIT, 1997.

[JLS97] A. Jones, Z. Luo, and S. Soloviev. Some proof-theoretic and algorithmic aspects of coercive subtyping. *Proc. of the Annual Conf on Types and Proofs (TYPES'96)*, 1997.

[LC98] Z. Luo and P. Callaghan. Mathematical vernacular and conceptual well-formedness in mathematical language. *Proc of LACL97.*, 1998. To appear in LNCS series.

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

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

[Luo97] Z. Luo. Coercive subtyping in type theory. *Proc. of CSL'96, the 1996 Annual Conference of the European Association for Computer Science Logic, Utrecht. LNCS 1258*, 1997.

[Luo98a] Z. Luo. Coercive subtyping. *Journal of Logic and Computation*, 1998. To appear.

[Luo98b] Z. Luo. Dependent coercions, 1998.

[NPS90] B. Nordström, K. Petersson, and J. Smith. *Programming in Martin-Löf's Type Theory: An Introduction.* Oxford University Press, 1990.

[Pus95] J. Pustejovsky. *The Generative Lexicon.* MIT, 1995.

[SL98] S. Soloviev and Z. Luo. Coercion completion and conservertivity in coercive subtyping. Talk given at and paper submitted to TYPES'98, 1998.