Type-Theoretical Semantics with Coercive Subtyping

Zhaohui Luo

July 2011

These notes are for Lectures 4 and 5 of the course on Lexical Semantics at the 2011 ESSLLI summer school in Ljubljana, Slovenia. The first three lectures of the course will be given by Prof Nicholas Asher and the lectures 4 and 5 by myself.

1 Summary on Type-Theoretical Semantics

Type-theoretical semantics. By *type-theoretical semantics*, we mean the formal semantics in *modern type theories*. It is a formal semantics in the style of the Montague semantics [Mon74], but in modern type theories with dependent types and inductive types, among others, rather than in Church's simple type theory [Chu40] as employed in the Montague semantics.[1] 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.

Examples of modern type theories include

- Martin-Löf's predicative type theory [ML84, NPS90], and
- the impredicative type theory ECC/UTT [Luo94].

In an *impredicative* type theory, there is a type $Prop$ of all logical propositions.[2]

Coercive subtyping. *Coercive subtyping* [Luo97, Luo99] is an adequate theory of subtyping for modern type theories.[3] The basic idea is that subtyping is provided by means of an abbreviation mechanism and, surprisingly, this simple idea provides powerful mechanisms for various forms of subtyping.

In computer science, modern type theories have been implemented in the so-called *proof assistants* (computer systems that help develop proofs of either mathematical theorems or correctness of programs) such as Agda [Agd08] and Coq [Coq07], and used in applications to formalisation of mathematics and verification of programs. The coercion mechanism

[1]Using the terminology *modern type theories*, we are trying to distinguish them from Church's simple type theory [Chu40] (or Montague's IL [Mon74]), the basic knowledge of which is tacitly assumed in this note.

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

[3]In contrast, the more traditional notion of subtyping, *subsumptive subtyping*, is inadequate for modern type theories. In subsumptive subtyping, the following *subsumption rule* is adopted:

$$\frac{a : A \quad A \leq B}{a : B}$$

This rule is incompatible with the notion of canonical object, a central notion in modern type theories. In particular, it violates the property of canonicity, an important one that says intuitively that every object of a type is equal to a canonical object of that type.

has been implemented in several proof assistants, including Coq [Coq07, Saï97], Lego [LP92, Bai99], Matita [Mat08] and Plastic [CL01].

Remarks on type-theoretical semantics and beyond.

- Type-theoretical semantics is very much in the spirit of *'typing as presuppositions'* as discussed by Asher in [Ash11].
- Type-theoretical semantics provides a promising alternative to the traditional Montague semantics with nice solutions, some of which are discussed here, to some of the difficult problems such as copredication as faced in the Montagovian setting.
- It would be interesting to see how far one can go to provide a type-theoretical semantics.

In the following, we shall give a brief introduction to type-theoretical semantics in modern type theories with coercive subtyping. I shall try to be intuitive and less technical so that the students who study linguistics or other less formal subjects can benefit.

2 Type-theoretical Semantics: Basics

We shall introduce the basics of type-theoretical semantics in modern type theories, in comparison with that in the Montague semantics.

Montague semantics in simple type theory. The Montague semantics is based on Church's simple type theory [Chu40], which is a *single-sorted* logic.[4] Here are typical examples in the Montague semantics, where e is the type of all entities and t the type of truth values:

- A sentence (S) is interpreted as a proposition of type t.

  (1) $[\![A\ man\ walks]\!] : t$.

- A common noun (CN) can be interpreted as a function of type $e \rightarrow t$ (a subset of entities).

  (2) $man : CN$
  (3) $[\![man]\!] : e \rightarrow t$

- A verb (IV) can be interpreted as a function of type $e \rightarrow t$ (a subset of entities).

  (4) $walk : IV$
  (5) $[\![walk]\!] : e \rightarrow t$

  Then,

  (6) $[\![John\ walks]\!] = [\![walk]\!]([\![John]\!])$, where $[\![John]\!] : e$.
  (7) $[\![A\ man\ walks]\!] = \exists m : e.\ [\![man]\!](m)\ \&\ [\![walk]\!](m)$.

[4] By 'single-sorted' here, we mean that there is a type e of all entities. Strictly speaking, there is another 'sort'/type t of truth values in Church's simple type theory.

- An adjective (Adj) can be interpreted as a function of type $(e \to t) \to (e \to t)$ (from subsets to subsets).

  (8) $handsome : Adj$

  (9) $[\![handsome]\!] : (e \to t) \to (e \to t)$

  Then,

  (10) $[\![handsome\ man]\!] = [\![handsome]\!]([\![man]\!])$.

Type-theoretical semantics in modern type theories. 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 *propositional types*, *inductive types* and other more advanced type constructions such as *type universes*. We shall introduce them below and explain how they may be used in type-theoretical semantics.

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 type-theoretical semantics [Ran94] (compare them with the above interpretation examples in the Montague semantics):

- A sentence (S) is interpreted as a proposition of type $Prop$, where $Prop$ is the type of logical propositions.[5]

  (11) $[\![A\ man\ walks]\!] : Prop.$

- A common noun (CN) can be interpreted as a type.

  (12) $man,\ human : CN$

  (13) $[\![man]\!],\ [\![human]\!] : Type$

- A verb (IV) can be interpreted as a predicate over the type D that interprets the domain of the verb (ie, a function of type $D \to Prop$).

  (14) $walk : IV$

  (15) $[\![walk]\!] : [\![animated]\!] \to Prop$

  Then, with the subtyping relation $[\![man]\!] \leq [\![animated]\!]$,[6]

  (16) $[\![John\ walks]\!] = [\![walk]\!]([\![John]\!])$, where $[\![John]\!] : [\![man]\!]$.

  (17) $[\![A\ man\ walks]\!] = \exists m : [\![man]\!]\ .\ [\![walk]\!](m)$.

- An adjective (Adj) can be interpreted as a predicate over the type that interprets the domain of the adjective.

  (18) $handsome : Adj$

  (19) $[\![handsome]\!] : [\![man]\!] \to Prop$

  Modified CNs can be interpreted by means of Σ-types (see §3.2 beflow). For example,

  (20) $[\![handsome\ man]\!] = \Sigma([\![man]\!], [\![handsome]\!]) : Type.$

[5] In this note, we assume that the impredicative universe $Prop$ exist in the type theory (eg, we use the impredicative type theory UTT).

[6] Subtyping is needed for the well-typedness of, eg, $[\![walk]\!]([\![John]\!])$. See S4 for more details.

3 Some Features in Modern Type Theories

We shall briefly introduce some features of a modern type theory and their uses in type-theoretical semantics.

3.1 Embedded logic

Propositions as types. A modern type theory has an *embedded logic* (or *internal logic*) based on the propositions-as-types principle [CF58, How80]. For example, there is a correspondence between the logical implication ($P \supset Q$) and the function type ($P \to Q$), and the universal quantifier ($\forall x{:}A.P(x)$) to the dependent Π-type $\Pi(A, P)$.

Type of logical propositions. In a so-called impredicative type theory, there is a type *Prop* of logical propositions, which is a totality and one can quantify over it to form other propositions such as $\forall P{:}Prop.P$ (and this process is regarded as 'circular' by predicativists [Fef05] or 'impredicative', in the technical jargon).

We use *Prop* in linguistic interpretations. As mentioned above, an assertive sentence is interpreted as a proposition of type *Prop* and a verb or an adjective as a predicate of type $A \to Prop$, where A is the domain whose objects the verb or adjective can be meaningfully applied to.

3.2 Dependent types

Modern type theories contain *dependent types*. Here are some examples.

Σ-types. Here are some basic laws governing Σ-types.

- If A is a type and B is an A-indexed family of types, then $\Sigma(A, B)$, or sometimes written as $\Sigma x{:}A.B(x)$, is a type.
- $\Sigma(A, B)$ consists of pairs (a, b) such that a is of type A and b is of type $B(a)$.
- When $B(x)$ is a constant type (i.e., always the same type no matter what x is), the Σ-type degenerates into product type $A \times B$ of non-dependent pairs.
- Σ-types (and product types) are associated projection operations π_1 and π_2 so that $\pi_1(a, b) = a$ and $\pi_2(a, b) = b$, for every (a, b) of type $\Sigma(A, B)$ or $A \times B$.

In a type-theoretical semantics, modified common nouns are interpreted as Σ-types. For instance, in the above Example (20), $\Sigma([\![man]\!], [\![handsome]\!])$ is the type of handsome men (or more precisely, of those men together with proofs that they are handsome).

Other dependent types. There are other dependent types. An example is the Π-types mentioned above: when A is a type and P is a predicate over A, $\Pi(A, P)$ or $\Pi x{:}A.P(x)$ is the dependent function type that, in the embedded logic, stands for the universally quantified proposition $\forall x{:}A.P(x)$. Π-types degenerates to the function type $A \to B$ in the non-dependent case.

3.3 Type Universes

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. In type-theoretical semantics, universes can be introduced to help semantic interpretations. We explain this by an example.

Type universe of CN interpretations. For instance, one may consider the universe $\textsc{cn} : Type$ of all common noun interpretations and, for each type A that interprets a common noun, there is a name $\overline{A}$ in $\textsc{cn}$. For example,

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

Interpretation of adverbs. The universe $\textsc{cn}$ can be used to give semantic interpretations to adverbs [Luo11]. An adverb modifies a verb (an adjective) to result in a verb (adjective) phrase.[7] Since, in a type-theoretical semantics, 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 'quickly' in 'John walked quickly' and 'simply' in 'That idea is simply ridiculous' would be interpreted as having the following ('polymorphic') type:

(21) $[\![quickly]\!],\ [\![simply]\!] : \Pi A : \textsc{cn}.\ (A \to Prop) \to (A \to Prop)$

For instance, the following phrase (22) can be interpreted as (23), which is of type $[\![animated]\!] \to Prop$:

(22) walk quickly

(23) $[\![quickly]\!]([\![animated]\!], [\![walk]\!])$

4 Coercive Subtyping

Coercive subtyping [Luo97, Luo99] is an adequate theory of subtyping for modern type theories. A theory of subtyping is crucial for type-theoretical semantics. We describe below how coercive subtyping may play an important role in this endeavor.

4.1 The basic idea of coercive subtyping

The basic idea of coercive subtyping is to consider subtyping as an abbreviation mechanism: A is a (proper) subtype of B ($A \leq B$) 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 $\mathbb{C}_B[\_]$ that expects an object of type B: $\mathbb{C}_B[a]$ is legal (well-typed) and equal to $\mathbb{C}_B[c(a)]$. (See Figure 1.)

For instance, one may introduce $[\![man]\!] \leq [\![human]\!]$. Then, if we assume that $[\![John]\!] : [\![man]\!]$ and $[\![shout]\!] : [\![human]\!] \to Prop$, the interpretation (25) of (24) is well-typed:

(24) John shouts.

(25) $[\![shout]\!]([\![John]\!])$

according to the rule of coercive subtyping, because $[\![man]\!] \leq [\![human]\!]$.

[7] There are other adverbs. For example, an adverb may modify sentences to result in new sentences and, similarly to Montague semantics, such adverbs are interpreted as functions from *Prop* to *Prop*.

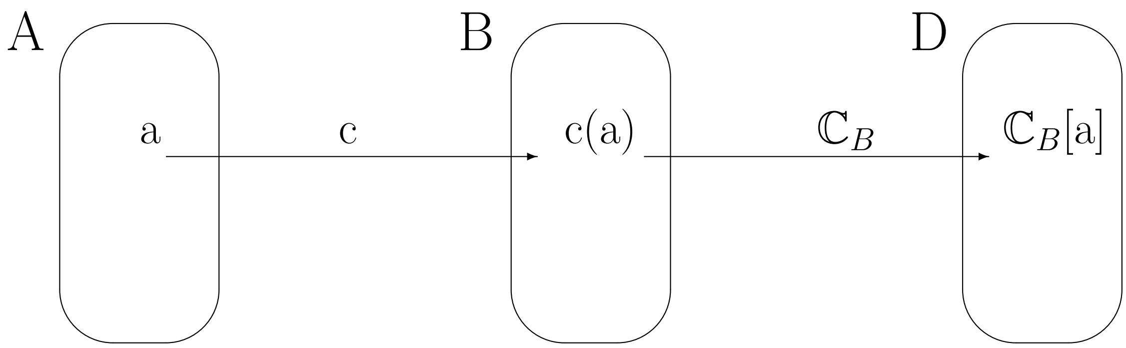

Figure 1: Pictorial explanation of $A \leq_c B$

4.2 Coercive Subtyping in Type-Theoretical Semantics

The usefulness of coercive subtyping in type-theoretical semantics is explained here with concrete examples. For further and formal details, see [Luo10] for dot-types and copredication and [Luo11] for the others.

Basic need for subtyping. Because CNs are interpreted as types in type-theoretical semantics in modern type theories, subtyping is crucially important.[8] Consider the following sentences:

(26) A man is a human.

(27) A handsome man is a man.

(28) Paul walks.

where *Paul* is a handsome man (ie, $[\![Paul]\!] : [\![handsome\ man]\!] = \Sigma([\![man]\!], [\![handsome]\!])$). To interpret the above sentences as intended, we need the following subtyping relationships: (29) is needed for (26), (30) for (27), and both (29) and (30) for (28).

(29) $[\![man]\!] \leq [\![human]\!]$

(30) $\Sigma([\![man]\!], [\![handsome]\!]) \leq_{\pi_1} [\![man]\!]$, where π_1 is the first projection (see §3.2).

Sense enumeration/selection via overloading. The representation of a sense enumeration model for anonymous words and the associated automated selection can be done by overloading (or *ad hoc polymorphism*) [Str00], which can be supported by coercive subtyping. Consider, for example, the anonymous word 'run' as in the following sentences:

(31) John runs quickly.

(32) John runs a bank.

In a type-theoretical semantics, we may have the following two different and homonymous meanings of 'run', corresponding to the above uses:

(33) $[\![run]\!]_1 : [\![human]\!] \to Prop$

(34) $[\![run]\!]_2 : [\![human]\!] \to [\![institution]\!] \to Prop$

[8] Ranta discussed this and called it the problem of multiple categorization of verbs (p62-64 in [Ran94]). Coercive subtyping provides a satisfactory solution to the problem.

$$\begin{tikzcd}
 & {[run]_1 : Human \to Prop} \\
run : \mathbf{1}_{run} \arrow[ur, "c_1"] \arrow[dr, "c_2"'] & \\
 & {[run]_2 : Human \to Institution \to Prop}
\end{tikzcd}$$

Figure 2: Example of sense enumeration/selection via coercions.

When the different meanings can be distinguished by their types (eg, in the case of 'run'), the sense selection model can be represented by means of *overloading* (or *ad hoc polymorphism*) supported by coercive subtyping. For instance, the sense selection model for the two meanings of 'run' is given by the following two coercions c_1 and c_2 (see Figure 2[9]):

$$\begin{aligned}
&c_1 : \mathbf{1}_{run} \to (Human \to Prop) \\
&c_1(run) = [\![run]\!]_1 \\
\\
&c_2 : \mathbf{1}_{run} \to (Human \to Institution \to Prop) \\
&c_2(run) = [\![run]\!]_2
\end{aligned}$$

This has the effect that, for example, in any context $\mathcal{C}_1[run]$ that requires an object of type $[\![human]\!] \to Prop$, we have

$$\mathcal{C}_1[run] = \mathcal{C}_1[c_1(run)] = \mathcal{C}_1[[\![run]\!]_1],$$

and, in any context $\mathcal{C}_2[run]$ that requires an object of type $[\![human]\!] \to [\![institution]\!] \to Prop$, we have

$$\mathcal{C}_2[run] = \mathcal{C}_2[c_2(run)] = \mathcal{C}_2[[\![run]\!]_2].$$

Therefore, through automated insertions of coercions, the sentences (31) and (32) will both be interpreted correctly.

Coercion contexts Word meanings are context-sensitive and, in order to express lexical semantics formally, a formal notion of context that allows declaration of coercions is very useful. Consider reference transfer in the following utterance (cf., [Nun95]):

(35) The ham sandwich shouts.

It is obvious that (35) 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).

A *coercion context* is a context whose entries may be of the form $A \leq_c B$ as well as the usual form $x : A$. For instance, the following context may be used to describe the special circumstances in a café:

(36) ..., $[\![ham\ sandwich]\!] \leq [\![human]\!]$, ...

[9] In Figure 2, $\mathbf{1}_{run}$ is the inductive unit type with 'run' as its only object. See Appendix A of [Luo11] for formal details.

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 (36), the above sentence (35) can be interpreted satisfactorily as intended.

Local coercions. Consider the following phrases that use the homonym 'bank':

(37) the bank of the river

(38) the richest bank in the city

The anonymous word 'bank' cannot be disambiguated by the typing of its semantic interpretations: eg, both of the two interpretations $[\![bank]\!]_1$ and $[\![bank]\!]_2$ are types (or they are of the same 'type' $Type$). Therefore, if we consider two coercions (cf, the above for the word 'run'):

$$\begin{array}{l} c_1 : \mathbf{1}_{bank} \to Type \\ c_1(bank) = [\![bank]\!]_1 \end{array}$$

$$\begin{array}{l} c_2 : \mathbf{1}_{bank} \to Type \\ c_2(bank) = [\![bank]\!]_2 \end{array}$$

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):

(39) $\textbf{coercion}\ \mathbf{1}_{bank} \leq_{c_1} Type\ \textbf{in}\ [\![(37)]\!]$

(40) $\textbf{coercion}\ \mathbf{1}_{bank} \leq_{c_2} Type\ \textbf{in}\ [\![(38)]\!]$

The coercions declared locally are only effective in the expressions in the scope of the keyword **in** and, therefore, the phrases in (37) and (38) are given semantics (39) and (40), respectively. as intended.

Dot-types and copredication. Dot-types are proposed by Pustejovsky in his Generative Lexicon Theory [Pus95]. Researchers have made proposals to model dot-types including, for example, [AP05, Coo11]. There are arguments about whether these do capture, and therefore give successful formal accounts of, dot-types. Here, we present a type-theoretic treatment of dot-types with the help of coercive subtyping, as proposed in [Luo10], 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 etc.

Our proposal is, intuitively:

- If types A and B *do not share components*, $A \bullet B$ is a well-formed type.
- If $A \bullet B$ is well-formed, then it is the type of pairs *both of whose projections are coercions.*

To explain the notion of component, whose formal definition can be found in [Luo10], it may be the best to give examples of types A and B which do share components (and hence cannot form a dot-type $A \bullet B$). Assume that PHY and INFO be the (different) types of physical objects and informational objects, respectively. Then,

- PHY and PHY share the component PHY (therefore, PHY $\bullet$ PHY is not well-formed).

- $\textsc{Phy}$ and $\textsc{Info}$ do not share components (therefore, $\textsc{Phy} \bullet \textsc{Info}$ is well-formed).
- $\textsc{Phy}$ and $\textsc{Phy} \bullet \textsc{Info}$ share the component $\textsc{Phy}$ (therefore, $\textsc{Phy} \bullet (\textsc{Phy} \bullet \textsc{Info})$ is not well-formed).

Also, we have, $[\![book]\!] \leq \textsc{Phy} \bullet \textsc{Info}$.

This notion of dot-types can be formalised in type theory with coercive subtyping [Luo10]. In particular, if $A \bullet B$ is well-formed, we have

$$A \bullet B \leq_{p_1} A \ \ and \ \ A \bullet B \leq_{p_2} B,$$

where p_1 and p_2 are the projection operators mapping $\langle a, b\rangle$ to a and b, respectively.

Dot-types can be used to give satisfactory treatments of, say, copredication. Consider the following example [Ash11]:

(41) 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 $\textsc{Phy}$ and $\textsc{Info}$ as above, it is natural that these phrases have the following types:

$$\begin{array}{lcl} [\![pick\ up]\!] & : & [\![human]\!] \rightarrow \textsc{Phy} \rightarrow Prop \\ [\![master]\!] & : & [\![human]\!] \rightarrow \textsc{Info} \rightarrow Prop \end{array}$$

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

$$\begin{array}{lcl} [\![pick\ up]\!] & : & [\![human]\!] \rightarrow \textsc{Phy} \rightarrow Prop \\ & \leq & [\![human]\!] \rightarrow \textsc{Phy} \bullet \textsc{Info} \rightarrow Prop \\ & \leq & [\![human]\!] \rightarrow [\![book]\!] \rightarrow Prop \\ & & \\ [\![master]\!] & : & [\![human]\!] \rightarrow \textsc{Info} \rightarrow Prop \\ & \leq & [\![human]\!] \rightarrow \textsc{Phy} \bullet \textsc{Info} \rightarrow Prop \\ & \leq & [\![human]\!] \rightarrow [\![book]\!] \rightarrow Prop \end{array}$$

In other words, $[\![pick\ up]\!]$ and $[\![master]\!]$ can both be used in contexts where terms of type $[\![human]\!] \rightarrow [\![book]\!] \rightarrow Prop$ are required and, therefore, the interpretation of the sentence (41) can proceed straightforwardly as intended.

Remark In a Montagovian setting, the interpretations of such sentences with copredication can become rather sophisticated. This is because, in Montague semantics, CNs are interpreted as functional subsets. Such an interpretation seems incompatible with the subtyping relationships involving $\textsc{Phy}$ and $\textsc{Info}$. In a type-theoretical semantics with coercive subtyping, where common nouns are interpreted as types, the interpretation of sentences with copredication is quite straightforward. □

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

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

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

[CF58] H.B. Curry and R. Feys. *Combinatory Logic*, volume 1. North Holland, 1958.

[Chu40] A. Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5(1), 1940.

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

[Coo11] R. 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.

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

[How80] W. 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.

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

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

[Luo97] Z. Luo. Coercive subtyping in type theory. *CSL'96, LNCS'1258*, 1997.

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

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

[Luo11] Z. Luo. Contextual analysis of word meanings in type-theoretical semantics. *Logical Aspects of Computational Linguistics (LACL'2011). LNAI 6736*, 2011.

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

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

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

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

[Nun95] Geoffrey Nunberg. Transfers of meaning. *J of Semantics*, 12(2), 1995.

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

[Ran94] A. Ranta. *Type-Theoretical Grammar*. Oxford University Press, 1994.

[Saï97] A. Saïbi. Typing algorithm in type theory with inheritance. *POPL'97*, 1997.

[Str00] C. Strachey. Fundamental concepts in programming languages. *Higher-Order and Symbolic Computation*, 13(1-2), 2000. (Paper based on 1967 lectures.).