Formal Semantics in Modern Type Theories: Is It Model-theoretic, Proof-theoretic, or Both?*

Zhaohui Luo**

Department of Computer Science
Royal Holloway, University of London
zhaohui.luo@hotmail.co.uk

Abstract. In this talk, we contend that, for NLs, the divide between model-theoretic semantics and proof-theoretic semantics has not been well-understood. In particular, the formal semantics based on modern type theories (MTTs) may be seen as *both* model-theoretic and proof-theoretic. To be more precise, it may be seen both ways in the sense that the NL semantics can first be *represented* in an MTT in a model-theoretic way and then the semantic representations can be understood *inferentially* in a proof-theoretic way. Considered in this way, MTTs arguably have unique advantages when employed for formal semantics.

1 Introduction

In logic, there are two well-developed traditions of semantic theories: model-theoretic semantics and proof-theoretic semantics. The former has been developed in Tarski's tradition while the latter by the logicians Gentzen and Prawitz and the philosopher Dummett among others (see, for example, [17]).

In formal semantics of NLs, the model-theoretic tradition has been dominant since the seminal work by Montague, a student of Tarski, who has used Church's simple type theory as an intermediate logical language for model-theoretic semantics [27]. The proof-theoretic semantics for NLs, however, has not been well developed, although there is recent work by Francez and colleagues in that direction (see, for example, [13,14]).[1]

We contend that, for formal semantics of NLs, the divide between model-theoretic semantics and proof-theoretic semantics has not been well-understood

* This paper is associated with the invited talk of mine in Logical Aspects of Computational Linguistics 2014. Its published version in LACL proceedings (LNCS 8535) contains some typos which have been corrected here.

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

[1] In philosophy, different kinds of philosophical semantics have been proposed, including the conceptual role semantics such as semantic inferentialism that has been advocated by Brandom [4,5] and others. Note that such philosophical semantic studies have rather different assumptions and more ambitious objectives as compared to formal semantics, although they have interesting influences on the latter.

and may have been over-exaggerated. In fact, the formal semantics in modern type theories (MTT-semantics, for short) [32,22] may be seen as *both* model-theoretic and proof-theoretic. To be more precise, it may be seen both ways in the following sense: the NL semantics can first be *represented* in an MTT in a model-theoretic way and then the semantic representations can be understood *inferentially* in a proof-theoretic way.

The MTT-semantics is in the tradition of the Montague semantics [27] from which, however, it has a key difference which may be summarised as follows:

- In the Montague semantics, the meanings are given in the set-theoretical models of the simple type theory, which acts as the intermediate logical language for model-theoretic semantics.
- In the MTT-semantics, the meanings are given in the language MTT itself, not in the 'models' of the MTT.

Put in another way, in an MTT-semantics, the MTT is not acting as an intermediate language, but rather as the meaning-carrying language itself.[2] On the one hand, the MTT-semantics may be seen as model-theoretic because, in such semantics, an MTT is employed as a *representational* language and it can do so because of its rich representational structure[3] as well as its internal logic. On the other hand, the MTT-semantics may be seen as proof-theoretic because the meanings of MTT-judgements can be understood by means of their *inferential roles* according to the meaning theories such as that developed by Martin-Löf for his type theory [25,28].

Besides this main theme, some related technical developments are to be reported. We describe a method of representing situations (incomplete possible worlds), which may be infinite or involve more sophisticated phenomena, by means of finite contexts in type theory with manifest entries (c.f., the study of manifest fields in [20]) and/or subtyping entries (c.f., the study of coercion contexts and local coercions in [22,23]) as well as the usual membership entries. The extension with such contextual entries has nice meta-theoretic properties.

Extending a brief argument in §1.2.3 of [18], in agreement with Dummett [12], I shall argue that a proof-theoretic semantics of MTTs should take *both* aspects of inferential use (verification and consequential application) seriously and that this applies to the understanding of judgements and type constructors in MTTs.

In this talk, after giving a summary of the MTT-semantics,[4] I shall elaborate the above theme and point out that, if we accept such views, MTTs may have some unique advantages when employed for formal semantics of NLs.

[2] It could be possible for one to consider an MTT as an intermediate language for model-theoretic semantics, although this view is not usually taken and it might be difficult as well. See §4 for a brief discussion in this respect.

[3] To see that MTTs provide rich representational structures, one may compare the notion of type with that of set and the notion of an MTT context with that of situations or incomplete possible worlds. These will be elaborated below in §2.

[4] The summary is omitted in the current paper (but see §2.1). For its references, see [32,22,21,9] among others.

2 Model-theoretic Characteristics of MTT-semantics

A modern type theory (MTT) has a rich type structure, much richer than the simple arrow types as found in the Montagovian setting.[5] The richness of type structures in an MTT allows it to be used as a *representational* language for formal semantics. Besides logical propositions (i.e., types that are understood as propositions), it contains many *types* which are employed for various useful representational purposes in constructing a formal semantics. These types include those interpreting (to mention a few examples) logical operations, common nouns, adjectival modifications, adverbial modifications, coordination, and also more advanced features such as subtyping, copredication and linguistic coercions.

In an MTT, the notion of *judgement* is the most basic. A typical form of judgement is:

$$\Gamma \vdash a : A,$$

which intuitively says that 'object a is of type A in context Γ', where a context, in its traditional form,[6] is a finite sequence of variable-type pairs of the form

$$x_1 : A_1,\ x_2 : A[x_1],\ ...,\ x_n : A_n[x_1, ..., x_{n-1}],$$

by which we assume that x_i be of type $A_i[x_1, ..., x_{i-1}]$ in which the 'previous' $x_1, ..., x_{i\_1}$ may occur $(i = 1, ..., n)$.

Employing MTTs for formal semantics, the following two points are worth being made explicit:

1. The types in MTTs can be used to represent collections of objects (or just called sets, informally) in a model-theoretic sense, although they are syntactic entities in MTTs.
2. The contexts in MTTs can be used to represent situations or (incomplete) possible worlds.

In this section, we shall illustrate these two points. First, in §2.1, we briefly describe how various type constructors in MTTs can be used in representing linguistic features in formal semantics. We then, in §2.2, give a simple example to explain how MTT contexts can be used to represent situations and, in §2.3, show how more sophisticated situations like infinite ones can be represented in finite contexts by means of manifest entries (c.f., manifest fields in [20]) and

[5] Technically, MTTs may be classified into predicative type theories such as Martin-Löfs type theory [25,28] and impredicative type theories such as the calculus of constructions (CC) [11] and the unifying theory of dependent types (UTT) [18]. In computer science, MTTs have been implemented in the proof assistants such as Coq [10] which has been used to implement the MTT-semantics and conduct experiments in Natural Language Inference [9]. In this paper, the type theory UTT [18] is used when concrete examples are given.

[6] We shall later in the paper extend the notion of context with manifest entries and subtyping entries.

subtyping entries (c.f., the study of coercion contexts and local coercions in [22,23]).[7]

2.1 Types as Sets: Examples

The rich type structure in MTTs provides adequate and useful tools for semanticists to construct formal semantics. Here, we only mention several examples without going into their details, because most of them have been explained in details in other places, and the appropriate references are given in each case.

1. Dependent sum types (Σ-types $\Sigma(A, B)$ which have product types $A \times B$ as special cases).
   - For example, Σ-types have been used to interpret adjectival modifications when the adjectives are intersective and subsective [32,8]. Note that subtyping is essential for such Σ-type interpretations to be adequate [22].[8]
2. Dependent product types (Π-types $\Pi(A, B)$, which have arrow-types $A \to B$ as special cases). These are basic dependent types that, together with universes (see below), provide polymorphism among other things.
   - For instance, verb modifying adverbs are typed by means of dependent Π-types (together with the universe CN of common nouns) [22,6].
3. Disjoint union types ($A + B$).
   - For instance, disjoint union types have been proposed to give interpretations of privative adjectives [8].
4. Universes. These are types of types, very useful since they make it possible to treat various collections of types as internal totalities. Typical examples of universes in MTT-semantics include, among others,
   - *Prop*, the universe of logical propositions, as found in impredicative type theories such as CC and UTT;
   - CN, the universe of (the interpretations of) common nouns [22]; and
   - *LType*, the universe of 'linguistic types', introduced in studying coordination [7].
5. Dot-types ($A \bullet B$). These are special types introduced to study the linguistic phenomena of copredication [22]. It is always worth mentioning that coercive subtyping is essentially employed in the formulation of the dot-types.

Besides the above, I should also emphasise that *subtyping* is crucial for an MTT to be a viable language for formal semantics. Furthermore, also very importantly, subtyping is needed when considering many linguistic features such as

[7] It is worth remarking that, formally, the studies in [20] and [22,23] are technically more complex in both cases: we studied manifest fields (not just manifest contextual entries) in [20] and local coercions (not just subtyping contexts) in [22,23]. In this paper, when we only consider such entries in signatures (see the brief formal treatment of the extension at the end of §2.3), it is formally simpler.

[8] See §3.3 of [32] for a very interesting discussion of the problem of 'multiple categorisation of verbs'. A satisfactory solution is an adequate subtyping mechanism for MTTs, provided by the framework of coercive subtyping [19,24].

copredication [1,31]. However, introducing subtyping into a Montagovian semantics has proven difficult, if not impossible. Coercive subtyping in MTTs solves this problem – it not only provides a satisfactory solution to the copredication problem [22] but a useful way to formalise various linguistic coercions [2].

2.2 Situations Represented as Contexts: a Simple Example

Here, we present a small example to show how an MTT can be used as a representational language. Situations,[9] or incomplete possible worlds, can be represented in MTTs as *contexts*. This was first studied by Ranta in [32]. Our example is simple; for instance, the domain of the situation in the example is finite. This will serve as a basis for our discussions on how more sophisticated situations such as those involving infinity or other special circumstances should be represented in §2.3, where we will show how manifest entries and subtyping entries can be used for such purposes.

Example 1. The example, taken from Chapter 10 of [33], is about an (imagined) situation in the Cavern Club at Liverpool in 1962 where the Beatles were rehearsing for a performance. This situation can be represented as follows.

1. The domain of the situation consists of several peoples including the Beatles (John, Paul, George and Ringo), their manager (Brian) and a fan (Bob). In type theory, this can be represented be means of the following context Γ_1:

$$\begin{aligned}\Gamma_1 \equiv\ & D : Type,\\ & John : D,\ Paul : D,\ George : D,\ Ringo : D,\ Brian : D,\ Bob : D\end{aligned}$$

2. The assignment function which, for example, assigns predicate symbols such as B and G to the propositional functions expressing 'was a Beatle' and 'played guitar', respectively. We can introduce the following in our context to represent such an assignment function:

$$\begin{aligned}\Gamma_2 \equiv\ & B : D \to Prop,\ b_J : B(John),\ ...,\ b_B : \neg B(Brian),\ b'_B : \neg B(Bob),\\ & G : D \to Prop,\ g_J : G(John),\ ...,\ g_G : \neg G(Ringo),\ ...\end{aligned}$$

   There are other predicate symbols as well: for example, there may be M being assigned the propositional function expressing 'was a manager', etc. We omit them here.

Eventually, we obtain a context $\Gamma \equiv \Gamma_1,\ \Gamma_2,\ ...,\ \Gamma_n$ that represents the situation. We shall then have, for instance,[10]

$$\Gamma \vdash G(John)\ \ true\ \ and\ \ \Gamma \vdash \neg B(Bob)\ \ true.$$

where $G(John)$ and $B(Bob)$ are the semantic interpretations of `John played Guitar` and `Bob was a Beatle`, respectively.

[9] Here, of course, we use the notion of situation informally, not in the formal sense in Situation Semantics [3].

[10] A judgement $\Gamma \vdash A\ true$ means that $\Gamma \vdash a : A$ for some a.

As we see from the above example, contexts in an MTT can be employed to represent situations model-theoretically and, in this endeavour, type theory is used as a meaning-carrying language (like set theory) rather than a language to be further interpreted.

2.3 Representing More Sophisticated Situations: Manifest and Subtyping Entries

The above example of situation is extremely simple: in particular, its domain D is finite. In general, one usually considers more sophisticated situations in which, for example, the domain is infinite or there are some other more sophisticated phenomena. In such cases, only using the traditional notion of context, which has only membership entries of the form $x : A$, is not enough. Here we consider two extensions to the notion of context:

1. *Manifest entries* of the form $x \sim a : A$, where $a : A$.[11]
2. *Subtyping entries* of the form $c : A < B$, where c is a functional operation from A to B.

They will provide extra powers in representing situations as contexts.

Manifest Entries in Contexts. A *manifest entry* in a context is of the form

$$x \sim a : A, \tag{1}$$

Informally, it assumes that x behave exactly the same as a of type A. Put in another way, in any place that we could use an object of type A, we could use x which actually plays the role of a. Formally, the above manifest entry (1) is an abbreviation of the following entry:

$$x : \mathbf{1}_A(a),$$

where $\mathbf{1}_A(a)$ is the inductive unit type parameterised by $A : Type$ and $a : A$, whose only object is $*_A(a)$. We also assume that $\mathbf{1}_A(a) <_{\xi_{A,a}} A$, where $\xi_{A,a}(x) = a$ for every $x : A$.[12] We have the following derivable formation rule for manifest contextual entries:

$$\frac{\Gamma \vdash a : A}{\Gamma,\ x \sim a : A\ valid} \quad (x \notin FV(\Gamma))$$

Using manifest entries, we can not only simplify contextual representations (the following Example 2) but represent infinite or other sophisticated situations by means of the finite contexts in type theory.

[11] Manifest entries are different from the definitional entries of the form $x = a : A$ in proof assistants – the former is just a notational abbreviation in the framework of coercive subtyping (see below), while the latter involves real extensions whose metatheory has proved to be not easy. For a study of the latter for Pure Type Systems, see [34].

[12] We omit the formal definitions of $\mathbf{1}_A(a)$ and the formal details for the coercion ξ here; see [20].

Example 2. With manifest entries, the situation in Example 1 can be represented as the following context:

$$D \sim a_D : Type,\ B \sim a_B : D \to Prop,\ G \sim a_G : D \to Prop,\ \ldots\ \ldots \qquad (2)$$

where

- $a_D = \{John,\ Paul,\ George,\ Ringo,\ Brian,\ Bob\}$ is the finite type (not the finite set) consisting of $John$ etc.
- $a_B : D \to Prop$, the predicate 'was a Beatle', is an inductively defined function such that $a_B(John) = a_B(Paul) = a_B(George) = a_B(Ringo) = True$ and $a_B(Brian) = a_B(Bob) = False$.
- $a_G : D \to Prop$, the predicate 'played guitar', is an inductively defined function such that $a_G(John) = a_G(Paul) = a_G(George) = True$ and $a_G(Ringo) = a_G(Brian) = a_G(Bob) = False$.

In other words, Γ_1 in Example 1 is now expressed by the first entry of (2) and Γ_2 in Example 1 by the second and third entries of (2).

Also, the domain in the situation in Example 1 is finite and, therefore, we can use the finitely many membership entries as in Example 1 to represent it. However, sometimes we have to deal with infinite domains. For instance, it is not difficult to imagine a situation where domain D consists of infinitely many things that we may represent by the natural numbers and some predicates are represented as propositional functions such as f over D. In such a case, we can use manifest entries like

$$D \sim Nat : Type,\ f \sim \text{f-defn} : D \to Prop$$

to introduce the infinite domain D and the infinite propositional function f, where f-defn can be defined inductively (in this case by inductive definition over the inductive type Nat).

Subtyping Entries in Contexts. Another useful form of contextual entries is the subtyping entries, for coercive subtyping, which are called coercion contexts in [22,23]. Formally, a subtyping entry is of the following form:

$$c : A < B,$$

where A and B are types and c is a functional operation from A to B. In a context with such an entry, A is a subtype of B via. coercion c.

Subtyping plays an essential role in MTT-semantics. It is not difficult to see that subtyping contextual entries are very useful in representing situations as contexts. As a simple example, one may introduce a subtyping entry to assume that everything in the domain D in Example 2 is a human: $D < Human$.

Manifest entries and subtyping entries are useful for some sophisticated circumstances as well. For instance,[13] one may imagine a situation (e.g., in the snake

[13] Thanks to Nicholas Asher for this example (in a private communication).

exhibition of a zoo) where every animal is a snake and such a situation may be represented by a context with the manifest entry $Animal_1 \sim Snake : Type$ or a subtyping entry expressing that $Animal_1$ is a subtype of $Snake$, where $Animal_1$ stands for the type of animals in the particular situation. One may regard such entries as unusual, although they describe the situation faithfully.

Formal Treatment of the Contextual Extensions. The above extension of the notion of contexts with manifest and subtyping entries can formally be considered by introducing the notion of *signature*. Informally, signatures can be used to represent situations in formal semantics. Formally, we consider the following forms of judgements:

$$\Gamma \vdash_\Sigma J \tag{3}$$

where Γ is a context in the traditional sense (a sequence of membership entries of the form $x : A$), J is one of $a : A$, $a = b : A$, A *type* or $A = B$ *type*, and Σ is a signature, a sequence of entries of one of the following forms:

1. $x : A$ (usual membership entries)
2. $x \sim a : A$ (manifest entries)
3. $c : A < B$ (subtyping entries)

What is new here is signatures, which could formally be regarded just as initial contextual segments whose entries will never be abstracted to the right of the $\vdash$-sign. In other words, the judgement (3) could be treated as $\Sigma,\ \Gamma \vdash J$ except that the entries in Σ cannot be abstracted.[14]

Since manifest entries of the form $x \sim a : A$ are just abbreviations of membership entries of the form $x : \mathbf{1}_A(a)$ (together with coercion $\xi_{A,a}$), it is straightforward to see that the extended type theory preserves the nice properties of the original type system as long as the signatures concerned are coherent[15]. For instance, if Σ is coherent and $\Gamma \vdash_\Sigma a : A$, then all of the terms a, A, and those in Γ and Σ are strongly normalising (and, as a consequence among many others, the embedded logic is logically consistent).

3 Proof-theoretic Characteristics of MTT-semantics

Besides model-theoretic semantics, there is another kind of formal semantics: *proof-theoretic semantics*. Proof-theoretic semantics was pioneered by Gentzen [15], developed by Prawitz [29,30] and Martin-Löf [25,26] (among other logicians) to study meaning theories of logical systems, and studied by Dummett [12] and Brandom [4,5] (among other philosophers) to study general philosophical theories of meaning of NLs. Proof-theoretic semantics is a form of inferential semantics in that it not only takes inference seriously but regards it as the central

[14] This is similar to the treatment of signatures in Edinburgh LF [16].

[15] If a well-formed signature Σ is *coherent* then $\Gamma \vdash_\Sigma A <_c B$ and $\Gamma \vdash_\Sigma A <_{c'} B$ imply $\Gamma \vdash_\Sigma c = c' : (A)B$. We do not get into the formal details here; see [19].

concept of meaning theories and such a direct link to inference is regarded as a key advantage of proof-theoretic semantics.[16]

3.1 Proof-Theoretic Semantics of MTTs

The MTT-semantics does not only have model-theoretic characteristics, as studied in the above section, but proof-theoretic characteristics. This is because that the meanings of judgements, which are the basic sentences in MTTs, can be understood in a proof-theoretic meaning theory. Martin-Löf has carried out a whole programme of proof-theoretic semantics, studying and developing it for his type theory [25,26].[17] Therefore, once MTT-semantics has been constructed, the statements in the semantics (i.e., judgement in the MTT) can be understood proof-theoretically.

Studying proof-theoretic meaning theories, people have considered two aspects of use: verification and consequential application. For MTTs, the first aspect is to consider how a judgement can be correctly asserted while the other (second) aspect is to consider what consequences it has to accept that a judgement is correct. The verificationist thinks that the first aspect of verification is the (possibly only) central conceptual focus in meaning theories while the pragmatist thinks that the second aspect of consequential application is instead central. For example, under the verificationist view on meaning explanations for type theory, one may consider such a meaning explanation of the judgement $a : A$ (I omit the context Γ here): it can correctly asserted if a computes to a canonical object of type A.

Personally, I take the view that *both* aspects of use are essential for a meaning theory of MTTs. This point of view is in agreement with Dummett [12] (if I read him correctly) and was briefly discussed in §1.2.3 of [18].[18] More concretely, this view says that the meanings of the type constructors are given by both of their introduction and elimination rules. Accordingly, the meanings of MTT-constructs and, in particular, the MTT-judgements, are given proof-theoretically (or inferentially, in more general terms). When the type constructors and judgements are employed in formal semantics, such proof-theoretic meanings can be taken as the basis of the semantics.

[16] There are philosophical arguments that conceptual role semantics (eg, proof-theoretic semantics) supports the view in cognitive science that something is meaningful just because that it plays a certain role (eg, inferential role) in a person's psychology. I ignore this because here proof-theoretic semantics is considered as an approach to formal semantics.

[17] But see Conclusion for the remarks on future work on meaning theories on MTTs in general and some existing problems (for example, about meaning explanations of hypothetical judgements).

[18] A reason for taking this view is that an MTT is much more sophisticated than first-order logical operators. However, I cannot detail its arguments in this paper.

$$(eI) \qquad \frac{\Gamma,\ \mathsf{j}\ \mathsf{isa}\ X \vdash S[\mathsf{j}]}{\Gamma \vdash S[\mathsf{every}\ X]} \quad (j\ \textit{fresh})$$

$$(eE) \qquad \frac{\Gamma \vdash S[\mathsf{every}\ X] \quad \Gamma \vdash \mathsf{j}\ \mathsf{isa}\ X \quad \Gamma,\ S[\mathsf{j}] \vdash S'}{\Gamma \vdash S'}$$

$$(sI) \qquad \frac{\Gamma \vdash \mathsf{j}\ \mathsf{isa}\ X \quad \Gamma \vdash S[\mathsf{j}]}{\Gamma \vdash S[\mathsf{some}\ X]}$$

$$(sE) \qquad \frac{\Gamma \vdash S[\mathsf{some}\ X] \quad \Gamma,\ \mathsf{j}\ \mathsf{isa}\ X,\ S[\mathsf{j}] \vdash S'}{\Gamma \vdash S'} \quad (j\ \textit{fresh})$$

Fig. 1. The PTS-rules for a basic language in [13].

3.2 Francez's Proof-Theoretic Semantics

In the literature, there have been some, but not many, researches on proof-theoretic semantics of NLs. The most notable is the recent work by Francez and his colleagues in a series of papers (see [13,14] among others). What I shall do here is to embed the basic rules of Francez's proof-theoretic semantics in [13] into the MTT-semantics and show that they are actually derivable or admissible.[19]

For instance, consider the rules of Figure 1 in [13], repeated here in our Figure 1 (we omit the structural rules and its treatment of scope here, which is irrelevant to our point and can be restored if needed). Employing the following 'translation' $[\![\_]\!]$, it is easy to see that the rules in Figure 1 are all derivable or admissible in the MTT semantics:

- The syntactic isa corresponds to the colon sign (:) in type theory. Thus, for example, $[\![\mathsf{j}\ \mathsf{isa}\ X]\!] = j : [\![X]\!]$.
- $[\![\mathsf{every}\ X]\!] = \forall([\![X]\!])$ of type $([\![X]\!])Prop$. For instance, the sentence (4) is interpreted as (5) after translation.

  (4) Every man walks.

  (5) $\forall([\![man]\!], [\![walk]\!])$

- $[\![\mathsf{some}\ X]\!] = \exists([\![X]\!])$ of type $([\![X]\!])Prop$. (Example omitted.)

For instance, the rule (eI) becomes

$$(eI') \qquad \frac{[\![\Gamma]\!],\ j : [\![X]\!] \vdash [\![S[j]]\!]}{[\![\Gamma]\!] \vdash [\![S[\forall(X)]]\!]}$$

which is admissible according to $\forall$-introduction. (Note: the condition that j is fresh is now embodied in the well-formedness of $[\![\Gamma]\!],\ j : [\![X]\!]$.)

[19] I cannot make serious comparisons here and further work is needed to compare these two approaches in a more exact way.

4 Concluding Remarks

In this talk, I have argued that the MTT-semantics may be seen as both model-theoretic and proof-theoretic, with situations represented as contexts and judgements understood inferentially.

There are several interesting directions for future work, either proof theoretically or model theoretically. Proof theoretically, it is important to develop further the meaning theories of modern type theories in general. For instance, it is interesting to study and obtain a better meaning explanation of an impredicative universe like that found in UTT. Even just for predicative type theories like that of Martin-Löf, people may argue that it is still unclear how hypothetical judgements should be explained satisfactorily. Further studies are called for.

As remarked in Footnote 2, one might consider model theory of MTTs, thinking of which as intermediate languages to be further interpreted when used for formal semantics. However, it is unclear whether such an approach can be successful, partly because that there is no such a general notion of models for MTTs as yet, although some initial work on models of generalised algebraic theories have been done. Substantial work is needed to get this clarified and developed.

References

1. N. Asher. *Lexical Meaning in Context: a Web of Words*. Cambridge University Press, 2012.
2. N. Asher and Z. Luo. Formalisation of coercions in lexical semantics. *Sinn und Bedeutung 17, Paris*, 2012.
3. J. Barwise and J. Perry. *Situations and Attitudes*. MIT Press, 1983.
4. R. Brandom. *Making It Explicit: Reasoning, Representing, and Discursive Commitment*. Harvard University Press, 1994.
5. R. Brandom. *Articulating Reasons: an Introduction to Inferentialism*. Harvard University Press, 2000.
6. S. Chatzikyriakidis. Adverbs in a modern type theory. In *LACL 2014*, 2014. (this volumn).
7. S. Chatzikyriakidis and Z. Luo. An account of natural language coordination in type theory with coercive subtyping. In Y. Parmentier and D. Duchier, editors, *Proc. of Constraint Solving and Language Processing (CSLP12). LNCS 8114*, pages 31–51, Orleans, 2012.
8. S. Chatzikyriakidis and Z. Luo. Adjectives in a modern type-theoretical setting. In G. Morrill and J.M Nederhof, editors, *Proceedings of Formal Grammar 2013. LNCS 8036*, pages 159–174, 2013.
9. S. Chatzikyriakidis and Z. Luo. Natural language reasoning using proof-assistant technology: Rich typing and beyond. *EACL Workshop on Type Theory and Natural Language Semantics, Goteborg*, 2014.
10. The Coq Development Team. *The Coq Proof Assistant Reference Manual (Version 8.1), INRIA*, 2007.
11. Th. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76(2/3), 1988.
12. M. Dummett. *The Logical Basis of Metaphysics*. Duckworth, 1991.

13. N. Francez and R. Dyckhoff. Proof-theoretic semantics for a natural language fragment. *Linguistics and Philosophy*, 33(6), 2011.
14. N. Francez, R. Dyckhoff, and G. Ben-Avi. Proof-theoretic semantics for subsentential phrases. *Studia Logica*, 94, 2010.
15. G. Gentzen. Untersuchungen über das logische schliessen. *Mathematische Zeitschrift*, 39, 1934.
16. R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, 1993.
17. R. Kahle and P. Schroeder-Heister, editors. *Proof-Theoretic Semantics*. Special Issue of Synthese, 148(3), 2006.
18. Z. Luo. *Computation and Reasoning: A Type Theory for Computer Science*. Oxford University Press, 1994.
19. Z. Luo. Coercive subtyping. *Journal of Logic and Computation*, 9(1):105–130, 1999.
20. Z. Luo. Manifest fields and module mechanisms in intensional type theory. In S. Berardi, F. Damiani, and U. de'Liguoro, editors, *Types for Proofs and Programs, Proc. of Inter. Conf. of TYPES'08, LNCS 5497.*, 2009.
21. Z. Luo. Common nouns as types. In D. Bechet and A. Dikovsky, editors, *Logical Aspects of Computational Linguistics (LACL'2012). LNCS 7351*, 2012.
22. Z. Luo. Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy*, 35(6):491–513, 2012.
23. Z. Luo and F. Part. Subtyping in type theory: Coercion contexts and local coercions (extended abstract). *TYPES 2013, Toulouse*, 2013.
24. Z. Luo, S. Soloviev, and T. Xue. Coercive subtyping: theory and implementation. *Information and Computation*, 223:18–42, 2012.
25. P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.
26. P. Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1), 1996.
27. R. Montague. *Formal Philosophy*. Yale University Press, 1974. Collected papers edited by R. Thomason.
28. B. Nordström, K. Petersson, and J. Smith. *Programming in Martin-Löf's Type Theory: An Introduction*. Oxford University Press, 1990.
29. D. Prawitz. Towards a foundation of a general proof theory. In P. Suppes *et al.*, editor, *Logic, Methodology, and Phylosophy of Science IV*, 1973.
30. D. Prawitz. On the idea of a general proof theory. *Synthese*, 27, 1974.
31. J. Pustejovsky. *The Generative Lexicon*. MIT, 1995.
32. A. Ranta. *Type-Theoretical Grammar*. Oxford University Press, 1994.
33. J. Saeed. *Semantics*. Wiley-Blackwell, 1997.
34. P. Severi and E. Poll. Pure type systems with definitions. *Proc. of LFCS'94, LNCS 813*, 1994.