MTT-semantics is both model-theoretic and proof-theoretic

Zhaohui Luo
Royal Holloway
University of London
Model-theoretic & Proof-theoretic Semantics

- Model-theoretic (traditional):
  - Denotations as central (cf, Tarski, ...)
  - Montague: NL $\rightarrow$ simple type theory $\rightarrow$ set theory

- Proof-theoretic (logics):
  - Inferential roles as central (Gentzen, Prawitz, Dummett, Brendom, ...)
  - E.g., logical operators given meaning via inference rules

- MTT-semantics:
  - Semantics in style of Montague semantics
  - But, in Modern Type Theories
Example argument for **traditional** set-theoretic sem.

- Or, an argument against non-set-theoretic semantics

**“Meanings are out in the world”**

- Portner’s 2005 book on “What is Meaning” – typical view
- Assumption that set theory represents (or even is) the world

Comments:

- This is an illusion! Set theory is just a theory in FOL, not “the world”.
- A good/reasonable formal system can be as good as set theory.
Claim:

Formal semantics in Modern Type Theories (MTT-semantics) is both model-theoretic and proof-theoretic.

- NL $\rightarrow$ MTT (representational, model-theoretic)
  - MTT as meaning-carrying language with its types representing collections (or “sets”) and signatures representing situations

- MTT $\rightarrow$ meaning theory (inferential roles, proof-theoretic)
  - MTT-judgements, which are semantic representations, can be understood proof-theoretically by means of their inferential roles
Traditional model-theoretic semantics:
Logics/NL $\rightarrow$ Set-theoretic representations

Traditional proof-theoretic semantics of logics:
Logics $\rightarrow$ Inferences

Formal semantics in Modern Type Theories:
NL $\rightarrow$ MTT-representations $\rightarrow$ Inferences
Why important for MTT-semantics?

- Model-theoretic – powerful semantic tools
  - Much richer typing mechanisms for formal semantics
  - Powerful contextual mechanism to model situations

- Proof-theoretic – practical reasoning on computers
  - Existing proof technology: proof assistants (Coq, Agda, Lego, …)
  - Applications of to NL reasoning

- Leading to both
  - Wide-range modelling as in model-theoretic semantics
  - Effective inference based on proof-theoretic semantics

Remark: new perspective & new possibility not available before!
This talk is based on:

- Collaborative work on MTTs and MTT-semantics with many people including, in recent years, among others:
  - S. Chatzikyriakidis (MTT-semantics)
  - S. Soloviev and T. Xue (coercive subtyping)
  - G. Lungu (signatures)
  - R. Adams, Callaghan, Pollack, ... (MTTs)

- Several papers including
This talk consists of three parts:

I. What is MTT-semantics?
   - Introduction to MTTs and overview of MTT-semantics

II. Model-theoretic characteristics of MTT-semantics
   - Signatures – extended notion of contexts to represent situations

III. Proof-theoretic characteristics of MTT-sem
   - Meaning theory of MTTs – inferential role semantics of MTT-judgements

Kent, June 2016
I. Modern Type Theories & MTT-semantics

- Type-theoretical semantics: general remarks
  - Types v.s. sets
- Modern Type Theories
  - Basics and rich type structure
- MTT-semantics
  - Linguistic semantics: examples
I.1. Type-theoretical semantics

- Montague Grammar (MG)
  - Richard Montague (1930 – 1971)
  - In early 1970s: Lewis, Cresswell, Parsons, ...
  - Later developments: Dowty, Partee, ...

- Other formal semantics
  - “Dynamic semantics/logic” (cf, anaphora)
  - Discourse Representation Theory (Kemp 1981, Heim 1982)
  - Situation semantics (Barwise & Berry 1983)

- Formal semantics in modern type theories (MTTs)
  - Ranta 1994 and recent development (this talk), making it a full-scale alternative to MG, being better, more powerful & with applications to NL reasoning based on proof technology (Coq, ...).

RHUL project [http://www.cs.rhul.ac.uk/home/zhaohui/lexsem.html](http://www.cs.rhul.ac.uk/home/zhaohui/lexsem.html)
Simple v.s. modern type theories

- **Church’s simple type theory**
  - As in Montague semantics
  - Base types ("single-sorted"): e and t
  - Composite types: e→t, (e→t)→t, ...
  - Formulas in HOL (eg, membership of sets)
    - Eg, s : e→t is a set of entities (a∈s iff s(a))

- **Modern type theories**
  - Many types of entities – "many-sorted"
    - Table, Man, Human, Phy, ... are types
  - Different MTTs have different embedded logics:
    - Martin-Löf’s type theory (1984): (non-standard) first-order logic
    - Impredicative UTT (Luo 1994): higher-order logic
Types v.s. Sets

- Both types and sets represent “collections of objects”
  - So, both may be used to represent collections in formal semantics (“model-theoretic”).
  - But, their similarity stops here.
  - MTT-types are “manageable”.
  - Some set-theoretical operations in set theory are destructive – they destroy salient MTT-properties.
    - Eg, intersection/union operations, a resulting theory is usually undecidable (see below).
I.2. MTTs (1) – Types

- Propositional types
  (Curry-Howard’s propositions-as-types principle)

<table>
<thead>
<tr>
<th>formula</th>
<th>type</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>A \implies B</td>
<td>A \to B</td>
<td>If ..., then ...</td>
</tr>
<tr>
<td>\forall x:A. B(x)</td>
<td>\prod x:A. B(x)</td>
<td>Every man is handsome.</td>
</tr>
</tbody>
</table>

- Inductive and dependent types
  - \Sigma(A,B) (intuitively, \{(a,b) \mid a:A \& b:B(a)\})
    - [handsome man] = \Sigma([man], [handsome])
  - \Pi x:A. B(x) (intuitively, \{f : A \to \bigcup_{a \in A} B(a) \mid a : A \& b : B(a)\})
  - A+B, A\times B, Vect(A), ...

- Universes
  - A universe is a type of (some other) types.
  - Eg, CN – a universe of the types that interpret CNs

- Other types: Phy, Table, A•B, ...
MTTs (2): Coercive Subtyping

- History: studied from two decades ago (Luo 1997) for proof development in type theory based proof assistants
- Basic idea: subtyping as abbreviation
  - $A \leq B$ if there is a (unique) coercion $c$ from $A$ to $B$. 
    Eg. $\text{Man} \leq \text{Human}; \Sigma(\text{Man}, \text{handsome}) \leq \text{Man}; \ldots$
- Adequacy for MTTs (Luo, Soloviev & Xue 2012)
  - Coercive subtyping is adequate for MTTs
  - Note: traditional subsumptive subtyping is not.
- Subtyping essential for MTT-semantics
  - $[\text{walk}] : \text{Human} \rightarrow \text{Prop}, [\text{Paul}] = p : [\text{handsome man}]$
  - $[\text{Paul walks}] = [\text{walk}](p) : \text{Prop}$
    because $p : [\text{handsome man}] \leq \text{Man} \leq \text{Human}$
MTTs (3): examples

- **Predicative type theories**
  - Martin-Löf’s type theory
  - Extensional and intensional equalities in TTs

- **Impredicative type theories**
  - Prop
    - Impredicative universe of logical propositions (cf, t in simple TT)
    - Internal totality (a type, and can hence form types, eg Table → Prop, Man → Prop, ∀X: Prop.X,
  - F/F^ω (Girard), CC (Coquand & Huet)
  - ECC/UTT (Luo, implemented in Lego/Plastic)
  - CIC_p (Coq-team, implemented in Coq/Matita)
MTTs (4): Technology and Applications

- Proof technology based on type theories
  - Proof assistants – ALF/Agda, Coq, Lego/Plastic, NuPRL, ...

- Applications of proof assistants
  - Math: formalisation of mathematics (eg, 4-colour Theorem in Coq)
  - CS: program verification and advanced programming
  - Computational Linguistics
    - E.g., MTT-sem based NL reasoning in Coq (Chatzikyriakidis & Luo 2014)
I.3. MTT-semantics

- Formal semantics in modern TTs
  - Formal semantics in the Montagovian style
  - But, in modern type theories (not in simple TT)
- Key differences from the Montague semantics:
  - CNs interpreted as types (not predicates of type \( e \rightarrow t \))
  - Rich type structure provides fruitful mechanisms for various linguistic features (CNs, Adj/Adv modifications, coordination, copredication, linguistic coercions, events, ...)
- Some work on MTT-semantics
  - Ranta (1994): basics of MTT-semantics
  - A lot of recent developments ... ...
## MTT-semantics

<table>
<thead>
<tr>
<th>Category</th>
<th>Semantic Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Prop</td>
</tr>
<tr>
<td>CNs (book, man, ...)</td>
<td>types (each CN is interpreted as a type: [book]. [man], ...)</td>
</tr>
<tr>
<td>IV</td>
<td>A → Prop (A is the “meaningful domain” of a verb)</td>
</tr>
<tr>
<td>Adj</td>
<td>A → Prop (A is the “meaningful domain” of an adjective)</td>
</tr>
</tbody>
</table>
MTT-semantics: examples

- Sentences as propositions: \([\text{A man walks}] : \text{Prop}\)
- Common nouns as types: \([\text{man}], [\text{handsome man}], [\text{table}] : \text{Type}\)
- Verbs as predicates: \([\text{shout}] : [\text{human}] \to \text{Prop}\)
  - \([\text{A man shouts}] = \exists m: [\text{man}]. [\text{shout}(m) : \text{Prop}]\)
  - Only well-typed because \([\text{man}] \leq [\text{human}]\) – subtyping is crucial.
- Adjectives as predicates: \([\text{handsome}] : [\text{man}] \to \text{Prop}\)
  - Modified CNs as \(\Sigma\)-types: \([\text{handsome man}] = \Sigma([\text{man}], [\text{handsome}])\)
  - Subtyping is crucial: \([\text{handsome man}] \leq [\text{man}]\)
- Adverbs as polymorphic functions:
  - \([\text{quickly}] : \prod A: \text{CN}. (A \to \text{Prop}) \to (A \to \text{Prop}), where \text{CN is universe of CNs}\)
Modelling Adjectives: Case Study

- **Intersective adjectives (eg, handsome)**
  - $\text{Adj}(N) \Rightarrow N \& \text{Adj}(N) \Rightarrow \text{Adj}$
  - $[\text{handsome man}] = \sum([\text{man}], [\text{handsome}])$

- **Subsective, but non-intersective, adjectives (eg, large)**
  - $\text{Adj}(N) \Rightarrow N$ (but not the 2\textsuperscript{nd} above)
  - $[\text{large}] : \Pi A:CN. (A \rightarrow \text{Prop})$
  - $[\text{large mouse}] = \sum([\text{mouse}], [\text{large}][\text{mouse}])$

- **Privative adjectives (eg, fake)**
  - $\text{Adj}(N) \Rightarrow \neg N$
  - $G = G_R + G_F$ – type of all guns
  - Declare $\text{inl}$ and $\text{inr}$ both as coercions: $G_R <_{\text{inl}} G$ and $G_F <_{\text{inr}} G$

- **Non-committal adjectives (eg, alleged)**
  - $\text{Adj}(N) \Rightarrow ?$
  - Employ “belief contexts” ...
MTT-sem: more examples of linguistic features

- **Anaphora analysis**
  - MTTs provide alternative mechanisms for proper treatments via $\Sigma$-types [Sundholm 1989] (cf, DRTs, dynamic logic, ...)

- **Linguistic coercions**
  - Coercive subtyping provides a promising mechanism (Asher & Luo 2012)

- **Copredication**

- **Generalised quantifiers** (Sundholm 1989, Lungu & Luo 2014)
  - \([\text{every}] : \Pi A:CN. (A \rightarrow \text{Prop}) \rightarrow \text{Prop}\)
  - \([\text{Every man walks}] = [\text{every}][[\text{man}], [\text{walk}]]\)

- **Event semantics** (Luo 2016)
  - Event types as dependent types $\text{Evt}(h)$ (rather than just Event)
MTT-semantics: implementation and reasoning

- MTT-based proof assistants (see earlier)
- Implementation of MTT-semantics in Coq
  - UTT v.s. CIC$_p$
  - They are implemented in Lego/Plastic and Coq, respectively.
  - They are essentially the same.
- Coq supports a helpful form of coercions
- Reasoning about NL examples (Chatzikyriakidis & Luo 2014)
- Experiments about new theories
  - Theory of predicational forms (Chatzikyriakidis & Luo 2016a)
  - CNs with identity criteria (Chatzikyriakidis & Luo 2016b)
II. MTT-sem: Model-theoretic Characteristics

- In MTT-semantics, MTT is a representational language.
- MTT-semantics is model-theoretic
  - Types represent collections – see earlier slides on using rich types in MTTs to give semantics.
  - Signatures represent situations (or incomplete possible worlds).
- Types and signatures/contexts are embodied in judgements:
  \[ \Gamma \vdash_{\Sigma} a : A \]
  where \( A \) is a type, \( \Gamma \) is a context and \( \Sigma \) is a signature.
- Contexts are of the form \( \Gamma \equiv x_1 : A_1, \ldots, x_n : A_n \)
- Signatures, similar to contexts, are finite sequences of entries, but
  - their entries are introducing constants (not variables; i.e., cannot be abstracted – c.f, Edinburgh LF (Harper, Honsell & Plotkin 1993)), and
  - besides membership entries, allows more advanced ones such as manifest entries and subtyping entries (see later).
Situations represented as signatures

- **Beatles’ rehearsal: simple example**
  - **Domain:** \( \Sigma_1 \equiv D : Type, \)
    
    \[
    \begin{align*}
    John : D, & \quad Paul : D, \\
    George : D, & \quad Ringo : D, \\
    Brian : D, & \quad Bob : D
    \end{align*}
    \]
  
  - **Assignment:** \( \Sigma_2 \equiv B : D \rightarrow Prop, b_J : B(John), ..., b_B : \neg B(Brian), b'_B : \neg B(Bob), \)
    
    \[
    \begin{align*}
    G : D \rightarrow Prop, & \quad g_J : G(John), ..., g_G : \neg G(Ringo), ...
    \end{align*}
    \]

  - **Signature representing the situation of Beatles’ rehearsal:**
    
    \( \Sigma \equiv \Sigma_1, \Sigma_2, ..., \Sigma_n \)

  - **We have, for example,**
    
    \( \Gamma \vdash_{\Sigma} G(John) \) true and \( \Gamma \vdash_{\Sigma} \neg B(Bob) \) true.

    “John played guitar” and “Bob was not a Beatle”.

Kent, June 2016
Subtyping Entries in Signatures

- **Subtyping entries** in a signature, where \( \kappa : (A)B : \)
  \[
  A \leq_{\kappa} B
  \]
- Eg, Man \( \leq_{\kappa} \) Human (\( \kappa \) depends on how Man is defined.)
- Eg, Vect(A,m) \( \leq_{\kappa(m)} \) List(A), parameterised by m : Nat, where \( \kappa(m) \) maps \( \langle n_1, \ldots, n_m \rangle \) to \( [n_1, \ldots, n_m] \).

- Note that, formally, for signatures with subtyping entries:
  - We do not need “local coercions” [Luo 2009] (no need to abstract subtyping entries to the right!)
  - This is meta-theoretically simpler (cf, [Luo & Part 2013])
Remark on coherence

- With subtyping entries, we don’t just need validity, but should also consider coherence, of signatures.
- Intuitively, from a coherent signature, one cannot derive two different coercions between the same types, in an appropriate subsystem of $T_s$, where the following coercive definition rule is removed:

\[
f : (x:A)B \quad a_0 : A_0 \quad A_0 \leq_{\kappa} A
\]

\[
f(a_0) = f(\kappa(a_0)) : \left[\kappa(a_0)/x\right]B
\]

(Formal definition omitted.)
Manifest Entries in Signatures

- More sophisticated situations
  - E.g., infinite domains
  - Traditional membership entries are not enough.
- In signatures, we can have a manifest entry:
  \[ c \sim a : A \]

where \( a : A \).
- Informally, it assumes constant \( c \) to behave the same as \( a \).
Manifest entries: examples

\[ \Sigma_1 \equiv D : Type, \]
\[ \quad \text{John} : D, \quad \text{Paul} : D, \quad \text{George} : D, \quad \text{Ringo} : D, \quad \text{Brian} : D, \quad \text{Bob} : D \]
\[ \Sigma_2 \equiv B : D \to Prop, \quad b_J : B(\text{John}), \quad \ldots, \quad b_B : \neg B(\text{Brian}), \quad b'_B : \neg B(\text{Bob}), \]
\[ \quad G : D \to Prop, \quad g_J : G(\text{John}), \quad \ldots, \quad g_G : \neg G(\text{Ringo}), \quad \ldots \]

\[ D \sim a_D : Type, \quad B \sim a_B : D \to Prop, \quad G \sim a_G : D \to Prop, \]

where

\[ a_D = \{ \text{John, Paul, George, Ringo, Brian, Bob} \} \]
\[ a_B : D \to Prop, \text{ the predicate ‘was a Beatle’,} \]
\[ a_G : D \to Prop, \text{ the predicate ‘played guitar’,} \]

with \( a_D \) being a finite type and \( a_B \) and \( a_G \) inductively defined. (Note: Formally, “Type” should be a type universe.)
Representations of infinite situations:

- Infinite domain $D$ represented by infinite type $\text{Inf}$
  
  $D \sim \text{Inf} : U$

- Infinite predicate with domain $D$:
  
  $f \sim f\text{-defn} : D \rightarrow \text{Prop}$

  with $f\text{-defn}$ being inductively defined.
Manifest Entries: Formal Treatment

- A manifest entry abbreviates two special entries.
- \( c \sim a : A \) abbreviates
  \[
  c : 1_A(a), \ 1_A(a) \leq_\xi A
  \]
  - \( 1_A(a) \) is the inductively defined unit type, parameterised by \( A \) and \( a \);
  - \( \xi(x) = a \) for \( x : 1_A(a) \).
- So, in any hole that requires an object of type \( A \), we can use “\( c \)” which, under the above coercion, will be coerced into “\( a \)”, as intended.
- In short, \( c \) stands for \( a \)!
Such manifest entries are intensional.

- Compare (weakly) extensional definitional entries: \( x = a : A \)
- Equivalent to
  \[ x : \text{Singleton}_A(a) \]
  where \( y = a : A \) if \( y:A \) (\( \eta \)-equality).
- But, in signatures, \( c \sim a : A \) is intensional (no \( \eta \)-equality).

Remarks:

- For contextual entries and manifest fields in \( \Sigma \)/record-types: see (Luo 2008).
- Here, we only consider manifest entries in signatures, as we only have subtyping entries in signatures.
Meta-theoretic Results

**Theorem**

Let $T$ be a type theory specified in LF and $T_S$ the extension of $T$ with signatures (with subtyping/manifest entries in signatures). Then, $T_S$ preserves the meta-theoretic properties of $T$ for coherent signatures.

Note: Meta-theoretic properties include Church-Rosser, strong normalisation, consistency, etc. Eg, as a special case of the above:

If $T$ satisfies SN ($\Gamma \vdash a : A \Rightarrow a$ is SN), then for $T_S$, if $\Gamma \vdash_{\Sigma} a : A$ for coherent $\Sigma \Rightarrow a$ is SN.
III. MTT-sem: Proof-theoretic Characteristics

- Proof-theoretic semantics
  - Meaning is use (cf, Wittgenstein, Dummett, Brandom)
    - Conceptual role semantics; inferential semantics
    - Inference over reference/representation
  - Two aspects of use
    - Verification (how to assert a judgement correctly)
    - Consequential application (how to derive consequences from a correct judgement)
Proof-theoretic semantics in logics

- Two aspects of use via introduction/elimination rules, respectively.
- Gentzen (1930s) and studied by Prawitz, Dummett, ... (1970s)
- Meaning theory for Martin-Löf’s type theory (Martin-Löf 1984)
- Further developed by philosopher Brendon (1994, 2000)

Proof-theoretic semantics for NLs

- Not much work so far
  - cf, Francez’s work (Francez & Dyckhoff 2011) under the name, but different ...
  - Traditional divide of MTS & PTS might have a misleading effect.
- MTT-semantics opens up new possibility – a meta/representational language (MTT) has a nice proof-theoretic semantics itself.
Meaning Explanations in MTTs

- Two aspects of use of judgements
  - How to prove a judgement?
  - What consequences can be proved from a judgement?

- Type constructors
  - They are specified by rules including, introduction rules & elimination rule.
  - Eg, for $\Sigma$-types

\[
\begin{align*}
(\Sigma-I) & \quad \frac{\Gamma \vdash_{\Sigma} a : A \quad \Gamma \vdash_{\Sigma} b : B(a) \quad \ldots}{\Gamma \vdash_{\Sigma} p(a, b) : \Sigma(A, B)} \\
(\Sigma-E) & \quad \frac{\Gamma \vdash_{\Sigma} a : A \quad \Gamma \vdash_{\Sigma} b : B(a) \quad \Gamma \vdash_{\Sigma} C : (\Sigma(A, B))Type}{\Gamma \vdash_{\Sigma} \epsilon_{\Sigma}(C, p(a, b)) : C(p(a, b))}
\end{align*}
\]
Verificationist meaning theory

- Verification (introduction rule) as central
- In type theory, meaning explanation via canonicity (cf, Martin-Löf); recall the following picture:

  ![Diagram](image)

  cf, strong normalisation property.
Pragmatist meaning theory

- Consequential application (elimination rule) as central
- This is possible for some logical systems
  - For example, operator \&.
- For dependent types, impossible.
  - One can only formulate the elimination rules based on the introduction operators!
Another view: both essential

- **Both** aspects (verification & consequential application) are essential to determine meanings.
  - Dummett
    - Harmony & stability (Dummett 1991), for simple systems.
  - For MTTs, discussions on this in (Luo 1994).
  - For a type constructor in MTTs, both introduction and elimination rules together determine its meaning.

- Argument for this view:
  - MTTs are much more complicated – a single aspect is insufficient.
  - Pragmatist view:
    - impossible for dependent types (see previous page)
  - Verificationist view:
    - Example of insufficiency – identity types
Identity type $\text{Id}_A(a,b)$ (eg, in Martin-Löf’s TT)

- Its meaning cannot be completely determined by its introduction rule (Refl), for reflexivity, alone.
- The derived elimination rule, so-called J-rule, is deficient in proving, eg, uniqueness of identity proofs, which can only be possible when we introduce the so-called K-rule [Streicher 1993].
- So, the meaning of $\text{Id}_A$ is given by either one of the following:
  - (Refl) + (J)
  - (Refl) + (J) + (K)

ie, elimination rule(s) as well as the introduction rule.
Concluding Remarks

❖ Summary
  ❖ NL $\rightarrow$ MTT (model-theoretic)
    ❖ Hence wide coverage of linguistic features
  ❖ MTT $\rightarrow$ meaning theory (proof-theoretic)
    ❖ Hence effective reasoning in NLs (eg, in Coq)

❖ Future work
  ❖ Proof-theoretic meaning theory
    ❖ E.g. impredicativity (c.f., Dybjer’s recent work in on “testing-based meaning theory”)
    ❖ Meaning explanations of hypothetical judgements
  ❖ General model theory for MTTs? But ...
    ❖ Generalised algebraic theories [Cartmell 1978, Belo 2007]
    ❖ Logic-enriched Type Theories (LTTs; c.f., Aczel, Palmgren, ...)