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 → simple type theory → 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 <u>traditional</u> 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 <u>is</u>) 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 → MTT (representational, model-theoretic)
    - MTT as meaning-carrying language with its <u>types</u> representing collections (or "sets") and <u>signatures</u> representing situations
  - MTT → 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 → Set-theoretic representations
- Traditional proof-theoretic semantics of logics:
  Logics → Inferences
- Formal semantics in Modern Type Theories:
  NL → MTT-representations → 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
  - Z. Luo. Formal Semantics in Modern Type Theories: Is It Model-theoretic, Proof-theoretic, or Both? Invited talk at Logical Aspects of Computational Linguistics 2014.

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

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

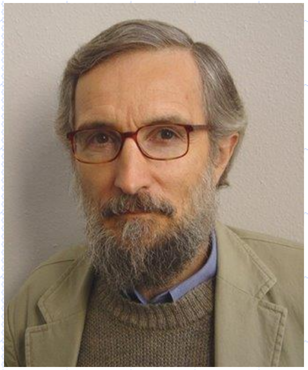
- Montague Grammar (MG)
  - Richard Montague (1930 – 1971)
  - In early 1970s: Lewis, Cresswell, Parsons, ...
  - Later developments: Dowty, Partee, …
- Other formal semantics
  - "Dynamic semantics/logic" (cf, anaphora)
  - Discourse Representation Theory (Kemp 1981, Heim 1982)
  - Situation semantics (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

- What typing is not:
  - "a : A" is not a logical formula.
    - 7 : Nat
    - Different from a logical formula is_nat(7)
  - "a : A" is different from the set-theoretic membership relation "a$\in$S" (the latter is a logical formula in FOL).
- What typing is related to (in linguistic semantics):
  - Meaningfulness (ill-typed ➔ meaningless)
  - Semantic/category errors (eg, "A table talks.")
  - Type presuppositions (Asher 2011)

Simple v.s. modern type theories

- Church's simple type theory
  - As in Montague semantics
  - Base types ("single-sorted"): e and t
  - Composite types: $e\to t$, $(e\to t)\to t$, …
  - Formulas in HOL (eg, membership of sets)
    - Eg, $s : e\to t$ is a set of entities ($a\in s$ 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).

Decidability of type-checking: an example difference

- "a : A" is decidable in STT (Church/Montague) or MTTs.
  - In contrast, the set membership "a $\in$ S" in set theory is not decidable.
- This decidability is essential for embedded logics in TTs.
  - HOL in STT and propositions-as-types logics for MTTs
  - Eg, we must be able to effectively apply HOL-rules in STT
  - Eg, in a propositions-as-types logic, we must be able to effectively check whether a is a proof of A (ie, a : A).
  - Working logics are necessary for formal semantics.

- MTTs have proof-theoretic meaning theories, while set theory does not:
  - MTTs are proof-theoretically specified by natural deduction rules (cf, Martin-Löf's meaning theory).
  - Meanings of MTT-judgements are given by means of their inferential roles – proof-theoretic semantics.

I.2. MTTs (1) – Types

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

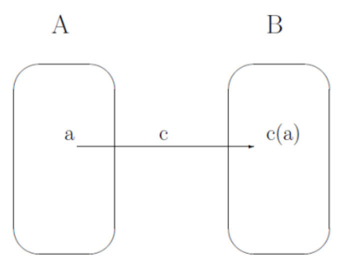
formula	type	example
$A \supset B$	$A \to B$	If …, then …
$\forall x{:}A.B(x)$	$\Pi x{:}A.B(x)$	Every man is handsome.

- Inductive and dependent types
  - $\Sigma(A,B)$ (intuitively, $\{ (a,b) \mid a : A \;\&\; b : B(a) \}$)
    - [handsome man] = Σ([man], [handsome])
  - $\Pi x{:}A.B(x)$ (intuitively, $\{ f : A \to \cup_{a \in A} B(a) \mid a : A \;\&\; b : B(a) \}$)
  - A+B, AxB, 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, $A \bullet B$, …

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≤B if there is a (unique) coercion c from A to B.

  Eg. Man ≤ Human; ∑(Man, handsome) ≤ Man; …



- Adequacy for MTTs (Luo, Soloviev & Xue 2012)
  - Coercive subtyping is adequate for MTTs
  - Note: traditional subsumptive subtyping is not.
- Subtyping essential for MTT-semantics
  - [walk] : Human→Prop, [Paul] = p : [handsome man]
  - [Paul walks] = [walk](p) : Prop

    because p : [handsome man] ≤ Man ≤ 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 <u>types</u> (not predicates of type $e \to t$)
  - 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

Category	Semantic Type
S	Prop
CNs (book, man, …)	types (each CN is interpreted as a type: [book]. [man], …)
IV	A→Prop (A is the “meaningful domain” of a verb)
Adj	A→Prop (A is the “meaningful domain” of an adjective)

MTT-semantics: examples

- Sentences as propositions: [A man walks] : Prop
- Common nouns as types: [man], [handsome man], [table] : Type
- Verbs as predicates: [shout] : [human]$\rightarrow$Prop
  - [A man shouts] = $\exists$m:[man]. [shout](m) : Prop
  - Only well-typed because [man] $\leq$ [human] – subtyping is crucial.
- Adjectives as predicates: [handsome] : [man]$\rightarrow$Prop
  - Modified CNs as Σ-types: [handsome man] = Σ([man], [handsome])
  - Subtyping is crucial: [handsome man] $\leq$ [man]
- Adverbs as polymorphic functions:
  - [quickly] : ΠA:CN. (A$\rightarrow$Prop)$\rightarrow$(A$\rightarrow$Prop), where CN is universe of CNs

MTT-sem: more examples of linguistic features

- Anaphora analysis
  - MTTs provide alternative mechanisms for proper treatments via Σ-types [Sundholm 1989] (cf, DRTs, dynamic logic, …)
- Linguistic coercions
  - Coercive subtyping provides a promising mechanism (Asher & Luo 2012)
- Copredication
  - Cf, [Pustejovsky 1995, Asher 2011, Retoré et al 2010]
  - Dot-types [Luo 2009, Xue & Luo 2012, Chatzikyriakidis & Luo 2015]
- Generalised quantifiers (Sundholm 1989, Lungu & Luo 2014)
  - [every] : Π A:CN. (A→Prop)→Prop
  - [Every man walks] = [every]([man], [walk])
- Event semantics (Luo 2016)
  - Event types as dependent types 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, Γ is a context and Σ 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: $\Sigma_1 \equiv D : Type,$
    $John : D,\ Paul : D,\ George : D,\ Ringo : D,\ Brian : D,\ Bob : D$
  - Assignment: $\Sigma_2 \equiv B : D \to Prop,\ b_J : B(John),\ ...,\ b_B : \neg B(Brian),\ b'_B : \neg B(Bob),$
    $G : D \to Prop,\ g_J : G(John),\ ...,\ g_G : \neg G(Ringo),\ ...$
  - 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".

Manifest entries

- More sophisticated situations
  - E.g., infinite domains
  - Traditional contexts with only membership entries are not enough
- In signatures, we can have a manifest entry:

$$x \sim a : A$$

  where a : A.
- Informally, it assumes x that behaves the same as a.
- Formally, it is an abbreviation of a emmbership entry and a subtyping entry (omitted).

Manifest entries: examples

$$\Sigma_1 \equiv D : Type,$$
$$John : D,\ Paul : D,\ George : D,\ Ringo : D,\ Brian : D,\ Bob : D$$
$$\Sigma_2 \equiv B : D \to Prop,\ b_J : B(John),\ ...,\ b_B : \neg B(Brian),\ b'_B : \neg B(Bob),$$
$$G : D \to Prop,\ g_J : G(John),\ ...,\ g_G : \neg G(Ringo),\ ...$$

➔➔➔

$$D \sim a_D : Type,\ B \sim a_B : D \to Prop,\ G \sim a_G : D \to Prop,$$

where

$$a_D = \{John,\ Paul,\ George,\ Ringo,\ Brian,\ Bob\}$$
$$a_B : D \to Prop, \text{ the predicate `was a Beatle'},$$
$$a_G : D \to Prop, \text{ the predicate `played guitar'},$$

with a_D being a finite type and a_B and a_G inductively defined.
(Note: Formally, "Type" should be a type universe.)

- Infinity:
  - Infinite domain D represented by infinite type Inf

    $D \sim Inf : Type$
  - Infinite predicate with domain D:

    $f \sim \text{f-defn} : D \rightarrow Prop$

    with f-defn being inductively defined.
- "Animals in a snake exhibition":

  $Animal_1 \sim Snake : CN$

Subtyping entries in signatures

- Subtyping entries in a signature:

  $$c : A \leq B$$

  where c is a functional operation from A to B.
- Eg, we may have

  $$D \sim \{ John, \ldots \} : Type, \quad c : D \leq Human$$
- Note that, formally, for signatures,
  - we only need "coercion contexts" but do not need "local coercions" [Luo 2009, Luo & Part 2013];
  - this is meta-theoretically much simpler (Lungu & Luo 2016)

Remarks

- Using contexts to represent situations: historical notes
  - Ranta 1994 (even earlier?)
  - Further references [Bodini 2000, Cooper 2009, Dapoigny/Barlatier 2010]
  - Remark: contexts introduce variables ➔ signatures are proper ways to represent situations as they introduce constants.
- Preserving TT's meta-theoretic properties is important!
  - Using the traditional notion of contexts is (of course) OK.
  - Our signatures with membership/manifest/subtyping entries are OK as well (meta-theory done by G. Lungu).
  - Extensions/changes need be careful: e.g., one may ask: are we preserving logical consistency under the propositions-as-types principle?

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

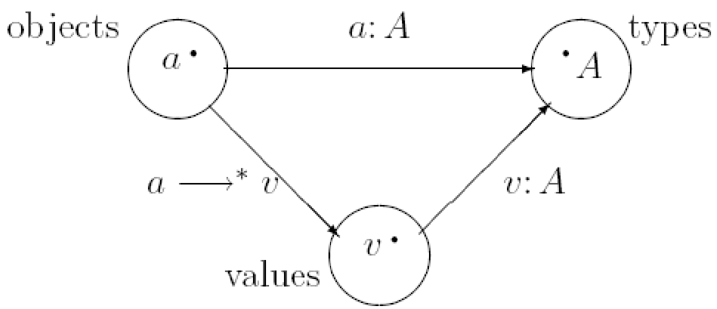
- Two aspects of use of judgements
  - How to prove a judgement?
  - What consequences can be proved from a judgement?
- Type constructors
  - They are specified by rules including, introduction rules & elimination rule.
  - Eg, for Σ-types

$$(\Sigma\text{-I}) \quad \frac{\Gamma \vdash_\Sigma a : A \quad \Gamma \vdash_\Sigma b : B(a) \quad \ldots}{\Gamma \vdash_\Sigma p(a,b) : \Sigma(A,B)}$$

$$(\Sigma\text{-E}) \quad \frac{\Gamma \vdash_\Sigma a : A \quad \Gamma \vdash_\Sigma b : B(a) \quad \Gamma \vdash_\Sigma C : (\Sigma(A,B))Type}{\Gamma \vdash_\Sigma \mathcal{E}_\Sigma(C,\ p(a,b)) : C(p(a,b))}$$

Verificationist meaning theory

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



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 $Id_A(a,b)$ (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 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 → MTT (model-theoretic)
    - Hence wide coverage of linguistic features
  - MTT → 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, …)