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

Zhaohui Luo
Dept of Computer Science
Royal Holloway, Univ of London

- Model-theoretic (traditional):
  - NL → set-theoretical models
  - E.g., Montague: NL → simple type theory → set theory
- Proof-theoretic:
  - NL → inferential roles
  - E.g., logical operators given meaning via inference rules
- MTT-semantics:
  - Semantics in style of Montague semantics
  - But, in Modern Type Theories

- Claim:

  Formal semantics in Modern Type Theories

  is both model-theoretic and proof-theoretic.

  - NL → MTT (representational, model-theoretic)
    - MTT as meaning-carrying language with its types representing collections (or "sets") and signatures 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 (c.f., Martin-Löf's meaning theory)

This talk

- What is MTT-semantics?
  - Introduction and overview
- Model-theoretic characteristics of MTT-sem
  - Signatures – extended notion of contexts to represent situations
- Proof-theoretic characteristics of MTT-sem
  - Meaning theory of MTTs – inferential role semantics of MTT-judgements

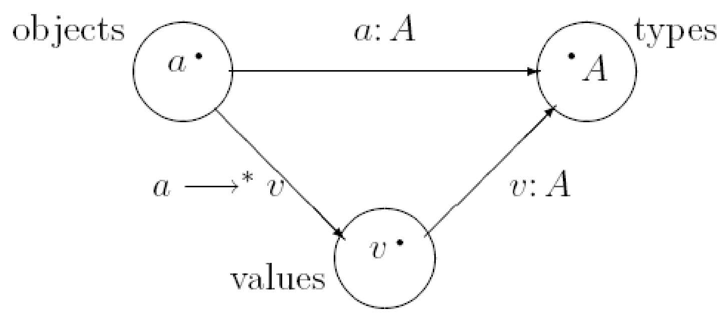
I. Modern Type Theories & MTT-semantics

- Church's simple type theory (Montague semantics)
  - Base types ("single-sorted"): e and t
  - Composite types: $e \rightarrow t$, $(e \rightarrow t) \rightarrow t$, …
  - Formulas in HOL (eg, membership of sets)
    - Eg, $s : e \rightarrow t$ 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 all types (of certain entities).
  - Different MTTs have different embedded logics; e.g.,
    - Martin-Löf's type theory (1984): (non-standard) first-order logic
    - Impredicative UTT (Luo 1994): higher-order logic

Types v.s. Sets

- Types are "collections of objects"
  - May be thought of as "manageable sets"
  - Model-theoretic
- Modern type theories have meaning theories:
  - Proof-theoretic
  - Meanings given by means of inferential roles
- Some typical differences
  - Typing is decidable: "a:A" is decidable (in intensional TTs), while the set membership "a$\in$S" is not.
  - Type theories can have an embedded/consistent logic, by propositions-as-types principle, while set theory is only a theory in FOL.

MTTs (1) – Canonicity



Examples:

- $A = N$, $a = 3+4$, $v = 7$.
- $A = N \times N$, $a = (\lambda x{:}N.\langle x,x+1\rangle)(2)$, $v = \langle 2,3\rangle$.

- Definition

  Any closed object of an inductive type is computationally equal to a canonical object of that type.

- This is a basis of MTTs.

MTTs (2) – Types

- Propositional types ("props-as-types")

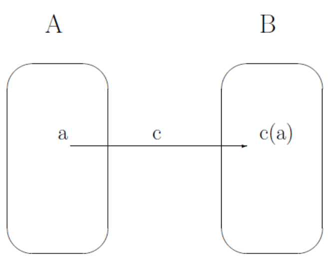
formula	type	example
$A \supset B$	$A \rightarrow B$	If …, then …
$\forall x{:}A.B(x)$	$\Pi x{:}A.B(x)$	Every man is handsome.

- 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, $\{ f : A \rightarrow \cup_{a \in A} B(a) \mid a : A \;\&\; b : B(a) \}$)
  - 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 (3): Coercive Subtyping

- 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 (4): Technology and Applications

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

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, …)
- Some work on MTT-semantics
  - Ranta (1994): basics of MTT-semantics
  - A lot of recent developments … …

MTT-semantics: examples

- Sentences as propositions: [A man walks] : Prop
- Common nouns as types: [man], [human], [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])
  - Coercive subtyping is crucial: [handsome man] $\leq$ [man]
  - Other classes of adjectives (Chatzikyriakidis & Luo 2013)
- Adverbs as polymorphic functions:
  - [quickly] : ΠA:CN. (A$\rightarrow$Prop)$\rightarrow$(A$\rightarrow$Prop), where CN is universe of CNs
  - Cf, [Chatzikyriakidis 2014]

MTT-sem: Some Advanced 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]
- Generalised quantifiers [Sundholm 1989, Lungu & Luo 2014]
  - [every] : Π A:CN. (A→Prop)→Prop
  - [Every man walks] = [every]([man], [walk])

II. MTT-sem: Model-theoretic Characteristics

- In MTT-semantics, MTT is a representational language.
- MTT-semantics is model-theoretic
  - Types represent collections (c.f., sets in set theory) – 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: $\Sigma_2 \equiv B : D \to Prop,\ b_J : B(John),\ \ldots,\ b_B : \neg B(Brian),\ b'_B : \neg B(Bob),$
    $$G : D \to Prop,\ g_J : G(John),\ \ldots,\ g_G : \neg G(Ringo),\ \ldots$$
  - Signature representing the situation of Beatles' rehearsal:
    $$\Sigma \equiv \Sigma_1,\ \Sigma_2,\ \ldots,\ \Sigma_n$$
  - We have, for example,
    $$\Gamma \vdash_\Sigma G(John)\ \ true \quad and \quad \Gamma \vdash_\Sigma \neg B(Bob)\ \ true.$$

    "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 <u>manifest entry</u>:

$$x \sim a : A$$

  where a : A.
- Informally, it assumes x that behaves the same as a.

Manifest entries: formal treatment

- Manifest entries are just abbreviations of special membership entries:
  - x ~ a : A abbreviates x : $1_A(a)$ where $1_A(a)$ is the unit type with only object $*_A(a)$.
  - with the following coercion:

$$\frac{\Gamma \vdash_\Sigma A : Type \quad \Gamma \vdash_\Sigma a : A}{\Gamma \vdash_\Sigma \mathbf{1}_A(a) \leq_{\xi_{A,a}} A : Type}$$

    where $\xi_{A,a}(z) = a$ for every z : $1_A(a)$.
- So, in any hole that requires an object of type A, we can use x which, under the above coercion, will be coerced into a, as intended.

Manifest entries: examples

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

➔➔➔

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

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:

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

    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:

  $$c : A < B$$

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

  $$D \sim \{ \text{John}, \ldots \} : \text{Type},\ c : D < \text{Human}$$
- 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.

Remarks

- Using contexts to represent situations: historical notes
  - Ranta 1994 (even earlier?)
  - Further references [Bodini 2000, Cooper 2009, Dapoigny/Barlatier 2010]
- We introduce "signatures" with new forms of entries: manifest/subtyping entries
  - Manifest/subtyping entries in signatures are simpler than manifest fields (Luo 2009) and local coercions (Luo & Part 2013).
- Preserving TT's meta-theoretic properties is important!
  - Ranta, Bodini, Dapoigny & Barlatier just use the traditional notion of contexts; so OK.
  - Our signatures with membership/manifest/subtyping entries are OK as well.
  - Other extensions/changes need be careful: e.g., one may ask: are we preserving logical consistency under propositions-as-types?

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)
- Proof-theoretic semantics for NLs
  - Not much work so far
    - cf, Francez's work (eg, (Francez & Dyckhoff 2011))
  - 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

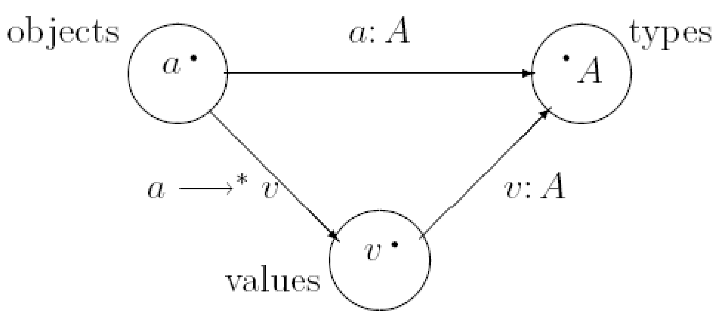
- Two aspects of use of judgements
  - How to prove a judgement?
  - What consequences can be proved from a judgement?
- Type constructors
  - They are specified by rules including, introduction rules & elimination rule.
  - Eg, for Σ-types

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

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

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)
  - MTT → meaning theory (proof-theoretic)
- 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, …)

References

The cited references in the talk refer to either those in the published paper in LACL 2014 proceedings or those listed below.

[Belo 2007] J. Belo. Dependently Sorted Logic. LNCS 4941.

[Bodini 2000] P. Bodini. Formalizing Contexts in Intuitionistic Type Theory. Fundamenta Informaticae 4(2).

[Cartmell 1978] Generalised algebraic theories and contextual categories, Ph.D. thesis, Oxford.

[Chatzikyriakidis 2014] Adverbs in a Modern Type Theory. This volume.

[Dapoigny/Barlatier 2010] Modelling Contexts with Dependent Types. Fundamenta Informaticae 104.

[Lungu & Luo 2014] Monotonicity Reasoning in Formal Semantics Based on Modern Type Theories. This volume.

[Luo 2009] Type-theoretical semantics with coercive subtyping. SALT20.

[Pustejovsky 1995] The Generative Lexicon. MIT.

[Retoré et al 2010] Towards a Type-Theoretical Account of Lexical Semantics. JoLLI 19(2).

[Streicher 1993] T. Streicher. Investigations into Intensional Type Theory. Habilitation Thesis, 1993.

[Sundholm 1989] Constructive Generalized Quantifiers. Synthese 79(1).

[Xue & Luo 2012] Dot-types and their implementation. LACL 2012, LNCS 7351.