Modern Type Theories and Their Applications in Formal Semantics

Zhaohui Luo

Royal Holloway, Univ. of London

This talk – three parts

I. Modern Type Theories: brief introduction

- Basics of MTTs (and meta-theory)
- Applications (verification, formalisation and semantics)

II. MTT-semantics (NL semantics in MTTs)

- Montague semantics v.s. MTT-semantics
- Adjectival modification: a case study

III. Donkey anaphora with both strong/weak sums

Part I. Modern Type Theories

Historical development of type theory

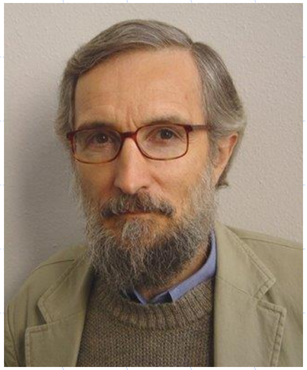
- Russell's ramified type theory (1925)
  - Paradoxes in naïve set theory
  - Zermelo: axiomatic set theory
  - Russell: ramified type theory ("axiom of reducibility")

- Ramsey (1926)
  - Logical v.s. semantic paradoxes
  - Impredicativity is circular, but not vicious.
    For example, $\forall X{:}Prop.X : Prop$.

- Church's simple type theory (1940)
  - Formal system based on λ-calculus
  - Higher-order logic with simple types (e, t, $e \rightarrow t$, …)

Modern Type Theories

- Martin-Löf has introduced/employed
  - Dependent/inductive types, type universes
  - Judgements with contexts, definitional equality
  - Curry-Howard principle of propositions-as-types



- Dependent types: "types segmented by indexes"
  - List ➔ Vect(n) with n:Nat (lists of length n)
- Examples of MTTs:
  - Predicative TTs:
    - Martin-Löf's intensional type theory MLTT [1973, …]
(non-standard FOL – strong sum Σ as existential quantifier; Agda)
  - Impredicative TTs:
    - CC [Coquand & Huet 1988] and CIC_p (HOL; Coq/Lean)
    - UTT [Luo 1990, 1994] (HOL; Lego/Plastic)

UTT = MLTT + CC

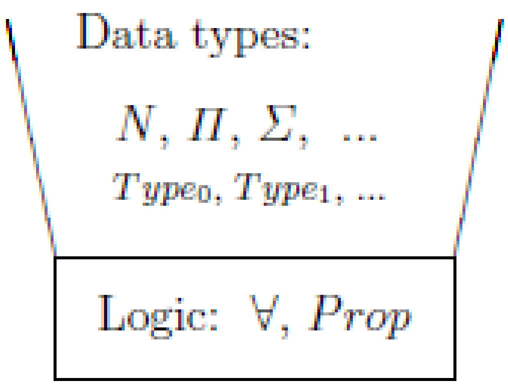
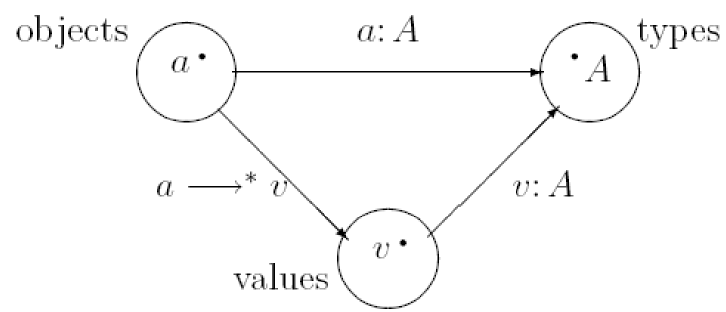


Fig. 1. The type structure in UTT.



Example: A = Nat, a = 3+4, v = 7.

- UTT has nice meta-theoretic properties
  - Goguen's PhD thesis on "Typed Operational Semantics" (1994)
  - Strong normalisation, which implies, e.g., consistency etc.

Σ-types – strong sum (example of dependent types)

- Informally (borrowing set-theoretical notations, formal rules next slide),

$$\Sigma x{:}A.B[x] = \{ (a,b) \mid a : A \text{ and } b : B[a] \}$$

- Uses include:
  - Representations of collections of structured data (types for "subsets": $\Sigma x{:}A.P[x]$ for A's such that P[x] holds).
  - In Matin-Lof's TT, Σ also plays the role of existential quantifier (strong version of Curry-Howard).

$$(\Sigma) \qquad \frac{\Gamma \vdash A\ type \quad \Gamma,\ x : A \vdash B\ type}{\Gamma \vdash \Sigma x : A.B\ type}$$

$$(pair) \qquad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B\ type}{\Gamma \vdash (a, b) : \Sigma x : A.B}$$

$$(\pi_1) \qquad \frac{\Gamma \vdash p : \Sigma x : A.B}{\Gamma \vdash \pi_1(p) : A}$$

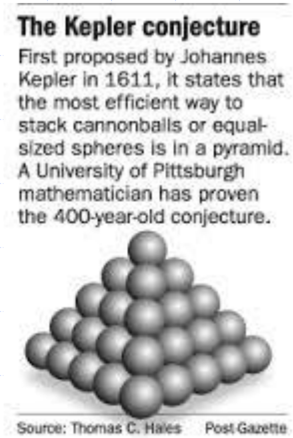
$$(\pi_2) \qquad \frac{\Gamma \vdash p : \Sigma x : A.B}{\Gamma \vdash \pi_2(p) : [\pi_1(p)/x]B}$$

$$(proj_1) \qquad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B\ type}{\Gamma \vdash \pi_1(a, b) = a : A}$$

$$(proj_2) \qquad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B\ type}{\Gamma \vdash \pi_2(a, b) = b : [a/x]B}$$

MTT-based technology and applications

- Proof technology based on type theories
  - Proof assistants
    - MTT-based: ALF/Agda, Coq, Lego, NuPRL, Plastic, …
    - HOL-based: Isabelle, HOL, …
- Applications of proof assistants
  - Math: formalisation of mathematics – eg,
    - 4-colour theorem (Coq), Kepler conjecture (Isabelle)
    - Homotopy type theory [HoTT 2013] (Coq/Agda)
  - Computer Science:
    - program verification and advanced programming
  - Computational Linguistics
    - NL reasoning based on MTT-sem (Coq)



Remark: effectiveness of applications

- More effective (much more) when built-in entities are used <u>directly</u>.
- Application examples:
  - Formalisation of mathematics
    - HoTT-based proof development (e.g., HITs for quotients) [HoTT 2013]
    - In contrast with, e.g., setoids and related formalisation/proofs.
  - Program verification
    - Built-in functions as FP programs (and their verification)
    - In contrast with, e.g., "deep embedding + semantics" (cumbersome …)
  - Linguistic semantics
    - CNs-as-types in MTT-semantics (see below)
    - In contrast with, e.g., CNs-as-predicates in Montague semantics.

Example: "built-in" sorting program

- Lists:

$$\frac{}{List[Nat]\ type} \qquad \frac{}{nil : List[Nat]} \qquad \frac{n : Nat \quad l : List[Nat]}{cons(n,l) : List[Nat]}$$

$$\frac{\Gamma \vdash c : C(nil) \quad \Gamma, x : Nat, y : List[Nat], z : C(y) \vdash f(x,y,z) : C(cons(x,y))}{\Gamma \vdash \mathcal{E}_L(c,f,l) : C(l)}$$

- Insertion sort:

$$isort : List[Nat] \to List[Nat]$$

$$isort(nil) = nil$$

$$isort(cons(n,l)) = insert(n, isort(l))$$

$$insert : Nat \to List[Nat] \to List[Nat]$$

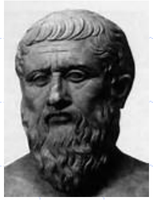
$$insert(n, nil) = cons(n, nil)$$

$$insert(n, cons(m,l))$$
$$= \underline{\text{if}}\ n \leq_2 m\ \underline{\text{then}}\ cons(n, cons(m,l))\ \underline{\text{else}}\ cons(m, insert(n,l))$$

Part II. MTT-semantics

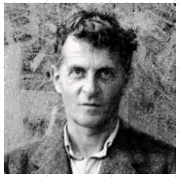
Natural Language Semantics

- Semantics – study of meaning (communicate = convey meaning)
- Various kinds of theories of meaning
  - Meaning is reference ("referential theory")
    - Word meanings are things (abstract/concrete) in the world.
    - c.f., Plato, …



  - Meaning is concept ("internalist theory")
    - Word meanings are ideas in the mind.
    - c.f., Aristotle, …, Chomsky.

  - Meaning is use ("use theory")
    - Word meanings are understood by their uses.
    - c.f., Wittgenstein, …, Dummett, Brandom.



Type-Theoretical Semantics

- Montague Semantics (Montague 1930–1971)
  - Dominating in linguistic semantics since 1970s
  - Set-theoretic, using simple type theory as intermediate
- MTT-semantics: formal semantics in modern type theories
  - Ranta (1994): formal semantics in Martin-Löf's type theory
  - Recent development on MTT-semantics ➔ full-scale alternative to Montague semantics
    - Z. Luo. Formal Semantics in Modern Type Theories with Coercive Subtyping. Linguistics and Philosophy, 35(6). 2012.
    - S. Chatzikyriakidis and Z. Luo. Formal Semantics in Modern Type Theories. Wiley/ISTE, 2020. (Monograph on MTT-semantics)
  - Research context on rich typing in NL (many researchers …)
    - S. Chatzikyriakidis and Z. Luo (eds.) Modern Perspectives in Type Theoretical Semantics. Springer, 2017.

MTT-semantics: basic categories

Category	**Semantic Type**
S	Prop (the type of all propositions)
CNs (book, man, …)	types (each common noun is interpreted as a type)
IV	$A \rightarrow Prop$ (A is the "meaningful domain" of a verb)
Adj	$A \rightarrow Prop$ (A is the "meaningful domain" of an adjective)
Adv	$\Pi A{:}CN.(A \rightarrow Prop) \rightarrow (A \rightarrow Prop)$ (polymorphic on CNs)

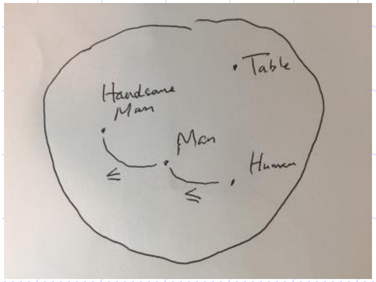
In MTT-semantics, common nouns (CNs) are types rather than predicates as in Montague semantics.

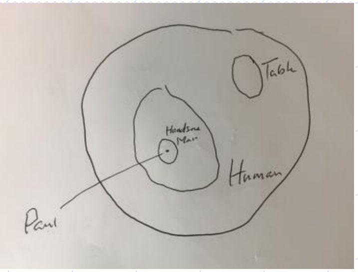
Modelling Adjectival Modification: Case Study

Classical classification	Example	Characterisation	MTT-semantics
intersective	handsome man	Adj(N) ➔ N & Adj	$\Sigma x{:}Man.handsome(x)$
subsective	large mouse	Adj(N) ➔ N (Adj depends on N)	$large : \Pi A{:}CN.\ A \rightarrow Prop$ $large(mouse) : Mouse \rightarrow Prop$
privative	fake gun	Adj(N) ➔ $\neg$N	$G = G_R + G_F$ with $G_R \leq_{inl} G$, $G_F \leq_{inr} G$
non-committal	alleged criminal	Adj(N) ➔ nothing	$H_{h,Adj} : Prop \rightarrow Prop$

- [Chatzikyriakidis & Luo: FG13, JoLLI17 & MTT-sem book 2020]

Note on Subtyping in MTT-semantics





- Simple example

  A human talks. Paul is a handsome man.

  Does Paul talk?

  Semantically, can we type talk(p)?

  (talk : Human→Prop & p : Σ(Man,handsome))

  Yes, because p : Σ(Man,handsome) ≤ Man ≤ Human.

- Subtyping is crucial for MTT-semantics
  - Coercive subtyping [Luo 1999, Luo, Soloviev & Xue 2012] is adequate for MTTs and we use it in MTT-semantics.
  - Note: Traditional subsumptive subtyping is inadequate for MTTs (eg, canonicity fails with subsumption.)

Advanced features in MTT-semantics: examples

- Copredication
  - Linguistic phenomenon studied by many (Pustejovsky, Asher, Cooper, Retoré, …)
  - Dot-types in MTTs [Luo 2009, Xue & Luo 2012, Chatzikyriakidis & Luo 2018]
  - Linguistic feature difficult, if not impossible, to find satisfactory treatment in a Montagovian framework.
- Several developments
  - Linguistic coercions via coercive subtyping [Asher & Luo (S&B12)]
  - Dependent event types [Luo & Soloviev (WoLLIC17)]
  - Propositional forms of judgemental interpretations [Xue et al (NLCS18)]
  - CNs as setoids [Chatzikyriakidis & Luo (Oslo 2018)]
  - MTT-sem in $MLTT_h$ (extension of MLTT with HoTT's logic) [Luo (LACompLing 2018)]

Part III. Donkey Anaphora with Both Strong and Weak Sums

Donkey anaphora

- Examples (Geach 1962, …)
  (*) Every farmer who owns <u>a donkey</u> beats <u>it</u>.
  (#) Every person who buys <u>a TV</u> and has <u>a credit card</u> uses <u>it</u> to pay for <u>it</u>.
- Strong/weak readings (Chierchia 1990):
  - Strong reading of (*):
    Every farmer who owns a donkey beats every donkey s/he owns.
  - Weak reading of (*):
    Every farmer who owns a donkey beats some donkeys s/he owns.

Original problem and use of dependent types

- Every farmer who owns a donkey beats it.
- In traditional logics:
  - $(\#)\ \forall x.\ [farmer(x)\ \&\ \exists y.(donkey(y)\ \&\ own(x,y))] \Rightarrow beat(x,y)$
    where $\exists$ is a "weak sum" and the last y is outside its scope.
- Using dependent types (Mönnich 85, Sundholm 86)
  - $\forall z : F_\Sigma.\ beat(\pi_1(z), \pi_1(\pi_2(z)))$ with $F_\Sigma = \Sigma x{:}F\ \Sigma y{:}D.\ own(x,y)$
    where Σ is the "strong sum" with two projections π_1 and π_2
  - Note: the interpretation only conforms to the strong reading.
- Σ plays a double role:
  - subset constructor (1^{st} Σ) and existential quantifier (2^{nd} Σ).
  - But this is problematic ➔ counting problem.

Problem of counting (Sundholm 89, Tanaka 15)

- Cardinality of finite types
  - |A| = n if A ≅ Fin(n) (i.e., it has exactly n objects.)
  - For example, |Σx:A.Fin(2)| = 2 x |A| (if A is finite.)
- Consider the donkey sentence with "most":
  - Most farmers who own a donkey beat it.
  - $Most_S\ z : F_\Sigma.\ beat(\pi_1(z), \pi_1(\pi_2(z)))$ with $F_\Sigma = \Sigma x{:}F\ \Sigma y{:}D.\ own(x, y)$
- But, this is inadequate – failing to "count" correctly:
  - $|F_\Sigma|$ = the number of (x,y,p) ≠ #(donkey-owning farmers)
  - E.g., 10 farmers, 1 owns 20 donkeys and beats all of them, and the other 9 own 1 donkey each and do not beat them.
  - The above sentence with "most" could be true – incorrect.
  - C.f., the "proportion problem" in using DRT to do this.

Why and …?

- "Double role" by Σ in $F_{\Sigma}=\Sigma x{:}Farmer\Sigma y{:}Donkey.own(x,y)$
  - First Σ: representing the collection of farmers such that …
  - Second Σ: representing the existential quantifier (!)
- But, unlike traditional ∃, Σ is strong:
  - $|\Sigma x{:}A.B|$ is the number of pairs (a,b), not just the number of a's such that B is true. So, the 2nd Σ is problematic.
- Can we somehow replace the 2nd Σ by ∃?
  - Yes, although not directly (c.f., the original scope problem), by considering different readings of donkey sentences <u>AND IF</u> we have both Σ and ∃ in the type theory.

    ➔ UTT (it has both Σ and ∃)
  - Note: ∃ in simple TT and Σ in Martin-Löf's TT, but <u>not both</u>.

Logic in UTT and proof irrelevance

- Formulas/propositions: ∀x:A.P, ∃x:A.P, P⇒Q, …
- Proof irrelevance:
  - Every two proofs of the same proposition are the same.
  - In UTT, this can be enforced by the following rule:

$$\frac{P : Prop \quad p : P \quad q : P}{p = q : P}$$

  - Note: Proof irrelevance would not be directly possible for, e.g., Martin-Löf's type theory (we'd need to consider $MLTT_h$ …)
- As a consequence, we have, for example:
  - |P| ≤ 1, if P : Prop (e.g., |∃x:A.R| ≤ 1)
  - |Σx:A.Q| ≤ |A|, if A is a finite type and Q : A→Prop

Donkey sentences in UTT

- Most farmers who own a donkey beat it.
  - Most farmers who own a donkey beat <u>every</u> donkey they own.
  - Most farmers who own a donkey beat <u>some</u> donkeys they own.
- "Most" in UTT
  - Definition similar to (Sundholm 89), but with ∃ as existential quantifier, instead of Σ.
- Interpretations

$$F_{\exists} = \Sigma x{:}F.\ \exists y{:}D.own(x,y)$$
$$Most\ z : F_{\exists}.\ \forall y' : \Sigma y{:}D.own(\pi_1(z),y).\ beat(\pi_1(z),\pi_1(y'))$$
$$Most\ z : F_{\exists}.\ \exists y' : \Sigma y{:}D.own(\pi_1(z),y).\ beat(\pi_1(z),\pi_1(y'))$$

Combining strong and weak sums

- How to add Σ to an impredicative type theory with ∃-propositions?

$$\Sigma \xrightarrow{\ ?\ } \boxed{\exists}$$

- Three possibilities:
  - UTT (seen before): Σ-types + ∃-propositions
  - "Large" Σ-propositions
    - ➔ logical inconsistency

$$\frac{A\ type \quad P : A \to Prop}{\Sigma x{:}A.P(x) : Prop}$$

  - "Small" Σ-propositions
    - ➔ weak ∃ becoming strong

$$\frac{A : Prop \quad P : A \to Prop}{\Sigma x{:}A.P(x) : Prop}$$

Conclusion: Only the UTT's approach is OK.

- How to add ∃ to a type theory with Σ-types?

$$\exists \xrightarrow{\quad ? \quad} \boxed{\Sigma}$$

- Not clear how to do this without changing the existing type theory.
- We can, for example, extend Martin-Löf's type theory with HoTT's "h-logic" to become $MLTT_h$ [Luo 2019].