



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, $\forall X:\text{Prop}.X : \text{Prop}$.

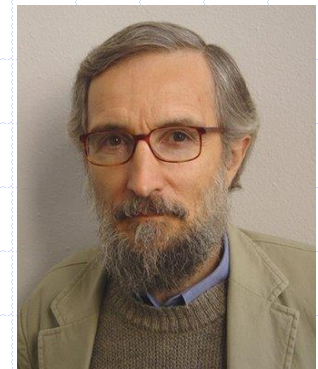
❖ Church's simple type theory (1940)

- ❖ Formal system based on λ -calculus
- ❖ Higher-order logic with simple types ($e, t, e \rightarrow t, \dots$)



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 \rightarrow Vect(n) with $n:\text{Nat}$ (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

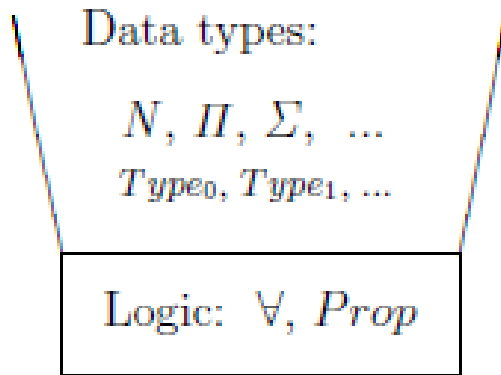
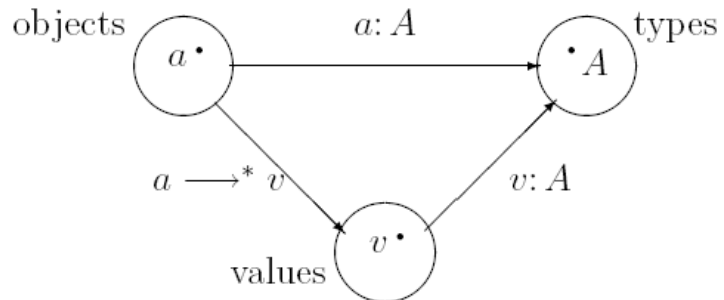


Fig. 1. The type structure in UTT.



Example: $A = \text{Nat}$, $a = 3+4$, $v = 7$.

- ❖ 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 Martin-Lof's $\mathbb{T}\mathbb{T}$, Σ also plays the role of existential quantifier (strong version of Curry-Howard).

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

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

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

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

$$(\text{proj}_1) \quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \pi_1(a, b) = a : A}$$

$$(\text{proj}_2) \quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B \text{ type}}{\Gamma \vdash \pi_2(a, b) = b : [a/x]B}$$

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)



The Kepler conjecture

First proposed by Johannes Kepler in 1611, it states that the most efficient way to stack cannonballs or equal-sized spheres is in a pyramid. A University of Pittsburgh mathematician has proven the 400-year-old conjecture.



Source: Thomas C. Hales - Post Gazette

Remark: effectiveness of applications

- ❖ More effective (much more) when built-in entities are used directly.
- ❖ 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:

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

❖ Insertion sort:

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

$$isort(nil) = nil$$

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

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

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

$$insert(n, cons(m, l))$$

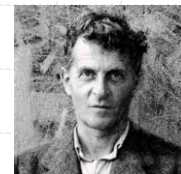
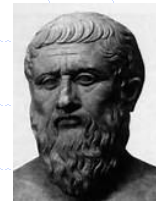
$$= \underline{\text{if}} \ n \leq_2 \ m \ \underline{\text{then}} \ cons(n, cons(m, l)) \ \underline{\text{else}} \ cons(m, insert(n, l))$$



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 | $A \rightarrow \text{Prop}$ (A is the "meaningful domain" of a verb) |
| Adj | $A \rightarrow \text{Prop}$ (A is the "meaningful domain" of an adjective) |
| Adv | $\prod A:\text{CN}.(A \rightarrow \text{Prop}) \rightarrow (A \rightarrow \text{Prop})$ (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 | $\text{Adj}(N) \rightarrow N \ \& \ \text{Adj}$ | $\sum x:\text{Man.handsome}(x)$ |
| subsective | large mouse | $\text{Adj}(N) \rightarrow N$ (Adj depends on N) | $\text{large} : \Pi A:\text{CN}. A \rightarrow \text{Prop}$ $\text{large}(\text{mouse}) : \text{Mouse} \rightarrow \text{Prop}$ |
| privative | fake gun | $\text{Adj}(N) \rightarrow \neg N$ | $G = G_R + G_F$ with $G_R \leq_{\text{inl}} G, G_F \leq_{\text{inr}} G$ |
| non-committal | alleged criminal | $\text{Adj}(N) \rightarrow \text{nothing}$ | $H_{h,\text{Adj}} : \text{Prop} \rightarrow \text{Prop}$ |

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

Note on Subtyping in MTT-semantics

❖ Simple example

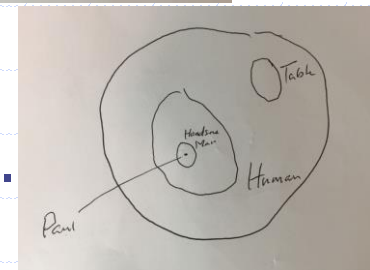
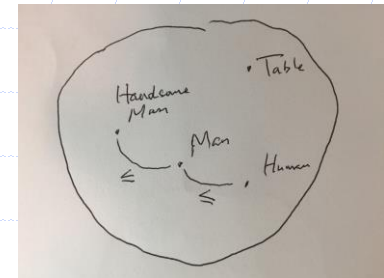
A human talks. Paul is a handsome man.

Does Paul talk?

Semantically, can we type $\text{talk}(p)$?

$(\text{talk} : \text{Human} \rightarrow \text{Prop} \ \& \ p : \Sigma(\text{Man}, \text{handsome}))$

Yes, because $p : \Sigma(\text{Man}, \text{handsome}) \leq \text{Man} \leq \text{Human}$.



❖ 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.)

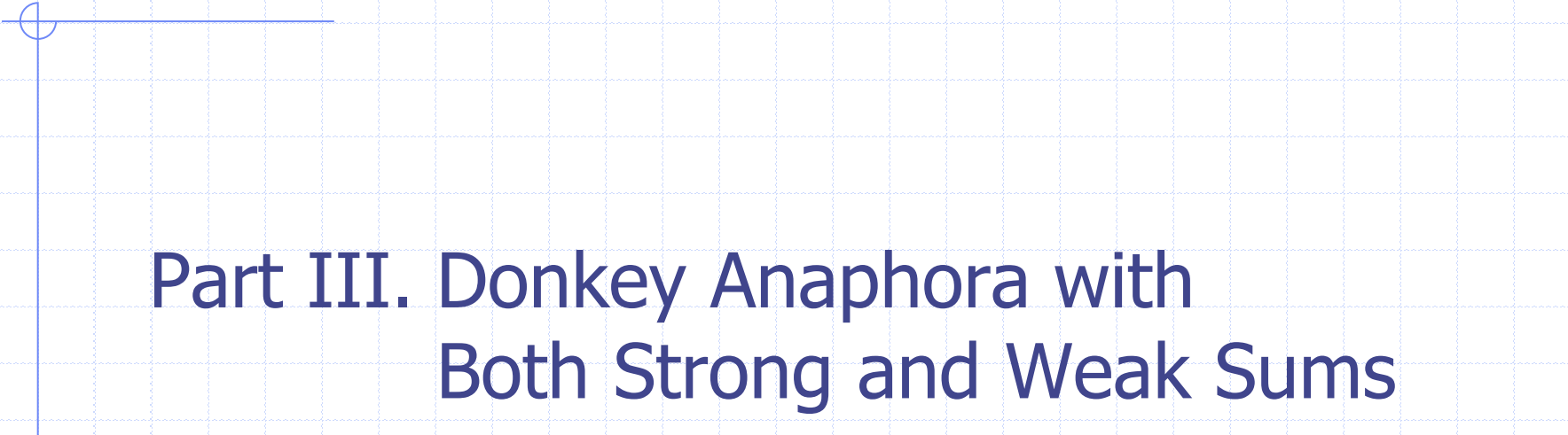
Advanced features in MTT-semantics: examples

❖ 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 a donkey beats it.

(#) Every person who buys a TV and has a credit card uses it to pay for it.

❖ 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 (1st Σ) and existential quantifier (2nd Σ).
 - ❖ But this is problematic \rightarrow counting problem.

Problem of counting (Sundholm 89, Tanaka 15)

❖ Cardinality of finite types

- ❖ $|A| = n$ if $A \cong \text{Fin}(n)$ (i.e., it has exactly n objects.)
- ❖ For example, $|\Sigma x:A. \text{Fin}(2)| = 2 \times |A|$ (if A is finite.)

❖ Consider the donkey sentence with “most”:

- ❖ Most farmers who own a donkey beat it.
- ❖ $\text{Most}_S z : F_\Sigma. \text{beat}(\pi_1(z), \pi_1(\pi_2(z)))$ with $F_\Sigma = \Sigma x:F \Sigma y:D. \text{own}(x, y)$

❖ But, this is inadequate – failing to “count” correctly:

- ❖ $|F_\Sigma|$ = the number of $(x, y, p) \neq \#(\text{donkey-owning farmers})$
- ❖ 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 $F_{\Sigma} = \Sigma x: \text{Farmer} \Sigma y: \text{Donkey.own}(x,y)$
 - ❖ First Σ : representing the collection of farmers such that ...
 - ❖ Second Σ : representing the existential quantifier (!)
- ❖ But, unlike traditional \exists , 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 \exists ?
 - ❖ Yes, although not directly (c.f., the original scope problem), by considering different readings of donkey sentences AND IF we have both Σ and \exists in the type theory.
 - ➔ UTT (it has both Σ and \exists)
 - ❖ Note: \exists in simple Π and Σ in Martin-Löf's Π , but not both.

Logic in UTT and proof irrelevance

❖ Formulas/propositions: $\forall x:A.P$, $\exists x:A.P$, $P \Rightarrow Q$, ...

❖ 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:
 - ❖ $|P| \leq 1$, if $P : Prop$ (e.g., $|\exists x:A.R| \leq 1$)
 - ❖ $|\Sigma x:A.Q| \leq |A|$, if A is a finite type and $Q : A \rightarrow Prop$

Donkey sentences in UTT

❖ Most farmers who own a donkey beat it.

- ❖ Most farmers who own a donkey beat every donkey they own.
- ❖ Most farmers who own a donkey beat some donkeys they own.

❖ “Most” in UTT

- ❖ Definition similar to (Sundholm 89), but with \exists as existential quantifier, instead of Σ .

❖ Interpretations

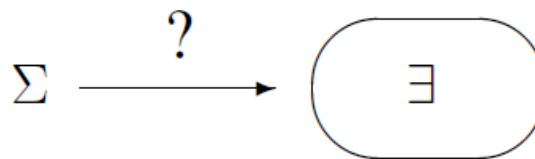
$$F_{\exists} = \Sigma x:F. \exists y:D.own(x, y)$$

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

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

Combining strong and weak sums

- ❖ How to add Σ to an impredicative type theory with \exists -propositions?



- ❖ Three possibilities:

- ❖ UTT (seen before): Σ -types + \exists -propositions

- ❖ “Large” Σ -propositions
→ logical inconsistency

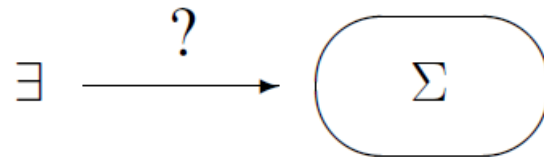
- ❖ “Small” Σ -propositions
→ weak \exists becoming strong

$$\frac{A \text{ type } P : A \rightarrow Prop}{\Sigma x:A.P(x) : Prop}$$

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

Conclusion: Only the UTT’s approach is OK.

- ❖ How to add \exists to a type theory with Σ -types?



- ❖ 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 .
 - ❖ Note: MLTT_h is a proper extension of MLTT .
 - ❖ [Luo 2019] for details.



Thank you!