# MTT-semantics for NLs and Its Formalisation

Zhaohui Luo
Royal Holloway
University of London

# This talk consists of two parts:

I. MTT-semantics

- Formal semantics of NLs in dependent/modern type theories
- Overview: what (basics), why (advantages) & how (advanced)

II. Focuses in this talk:

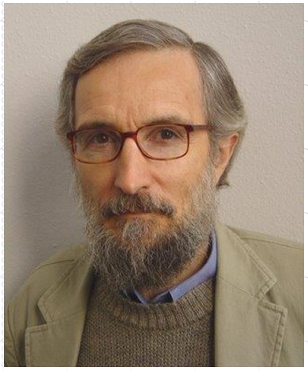
- MTT-semantics is both model-theoretic and proof-theoretic
  - MTT-semantics and reasoning in Coq
- Selected case studies: some theories under development
  - dependent event types, negated sentences and conditionals, CNs as setoids, …

# I. MTT-semantics

- Montague Grammar (MG)
  - R. Montague (1930–1971): MG in Church's simple type theory
  - Dominating in linguistic semantics since 1970s
- Development of formal semantics in last decades:
  - Discourse Representation Theory & Dynamic Logic (anaphora …)
  - Situation Semantics (situations – incomplete/partial worlds …)
- MTT-semantics: formal semantics in modern type theories
  - Ranta (1994) & recent development ➔ full-scale alternative to MG
  - Advantages: both model/proof-theoretic, proof tech. support, …
- Two books:
  - Chatzikyriakidis and Luo (eds.) Modern Perspectives in Type Theoretical Semantics. Springer, 2017. (Collection on rich typing in NL semantics)
  - Chatzikyriakidis and Luo. Formal Semantics in Modern Type Theories. ISTE/Wiley, to appear. (Monograph on MTT-semantics)

# Simple v.s. Modern Type Theories

- Church's simple type theory (1940)
  - As in Montague semantics
  - Types ("single-sorted"): e, t, $e \rightarrow t$, …
  - HOL (e.g., membership of `sets')
    - E.g., 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, $\Sigma$(Man, handsome), Phy•Info, …
  - Examples of MTTs:
    - Martin-Löf's TT (predicative; non-standard FOL)
    - $CIC_p$ (Coq) & UTT (Luo 1994) (impredicative; HOL)



# MTT-semantics compared with MG

| | Example | Montague semantics | Semantics in MTTs |
|---|---|---|---|
| CN | man, human | $[\![man]\!],\ [\![human]\!] : e \rightarrow t$ | $[\![man]\!],\ [\![human]\!] : Type$ |
| IV | talk | $[\![talk]\!] : e \rightarrow t$ | $[\![talk]\!] : [\![human]\!] \rightarrow Prop$ |
| Adj | handsome | $[\![handsome]\!] : (e \rightarrow t) \rightarrow (e \rightarrow t)$ | $[\![handsome]\!] : [\![man]\!] \rightarrow Prop$ |
| MCN | handsome man | $[\![handsome]\!]([\![man]\!])$ | $\sum[m : [\![man]\!],\ h : [\![handsome]\!](m)]$ |
| S | A man talks | $\exists m : e.\ [\![man]\!](m) \& [\![talk]\!](m)$ | $\exists m : [\![man]\!].\ [\![talk]\!](m)$ |

A key difference: CNs as types rather than predicates

(*) John is a man.

- MTT-sem:  j : Man where Man : Type
- Montague: man(j)  where man : e→t

(#) The table talks.  ("selection restriction": meaningfulness v.s. truth)

- talk(t)?
  - Untypable/meaningless in MTT-sem (talk:Human→Prop & t:Table)
  - Well-typed/false in Montague (t:e and [talk] : e→t)

# Modelling Adjective Modifications

| classification | characterisation | example | MTT-semantics |
|---|---|---|---|
| intersective | Adj(N) ➔ N & Adj | handsome man | $\Sigma x$:Man.handsome(x) |
| subsective | Adj(N) ➔ N (not Adj) | large mouse | large : $\Pi A$:CN. A→Prop<br>large(mouse) : Mouse→Prop |
| privative | Adj(N) ➔ ¬N | fake gun | $G = G_R + G_F$<br>with $G_R \leq_{inl} G$, $G_F \leq_{inr} G$ |
| non-committal | Adj(N) ➔ nothing | alleged criminal | $\Sigma h$:Human.$B_h$(criminal) |

Cf, [Chatzikyriakidis & Luo 2013]

# Note on Subtyping

- Subtyping essential for MTT-semantics
  - Could a "handsome man" talk?
  - Paul talks ➔ talk(p)?

    where talk:Human→Prop and p:[handsome man]
  - talk(p) : Prop

    because p : [handsome man] ≤ Man ≤ Human
- Remarks
  - Subtyping is crucial for MTT-semantics
  - Coercive subtyping is adequate for MTTs and we use it in MTT-semantics.

# Advanced features in MTT-semantics: examples

- 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]
- Dependent event types
  - Event semantics [Davidson 67] and its neo-Davidsonian turn [Parsons 90] (better treatment of adverbial modification, among others)
  - Dependent event types [Luo & Soloviev 2017] (refined event types such as Evt(h), solving the "event quantification problem", among others)

# II. MTT-semantics is both model/proof-theoretic

- 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:
  - Opens a new perspective in semantics
  - Both model-theoretic and proof-theoretic – in what sense?
  - What does this imply?

- 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
  - Z. Luo. Formal Semantics in Modern Type Theories: Is It Model-theoretic, Proof-theoretic, or Both? Invited talk at LACL14.

- 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!*

# MTT-semantics being model-theoretic

- Types and signatures
  - Collection by types (cf, set-theoretical semantics)
  - Situations (incomplete worlds) by signatures (cf, situation theory)
- Signatures $\Sigma$ in $\Gamma \vdash_{\Sigma} a : A$
  - Cf, notion of signature in Edinburgh LF [Harper et al 1987]
- New forms of signature entries (besides c:A)

$$\ldots,\ c{:}A,\ \ldots,\ A \leq_c B,\ \ldots,\ c \sim a : A,\ \ldots$$

  - Subtyping entries (cf, Lungu's forthcoming PhD thesis)
  - Manifest entries (c is a of type A; cf, TYPES08 paper)
- Representational power:
  - Various situations (e.g., linguistic coercions and infinite situations)
  - Wide coverage (a major "advantage" of model-theoretic semantics)

# Proof-theoretic semantics: historical remarks

- Philosophy
  - Meaning is use (Wittgenstein)
  - Inference over presentation (Dummett, Brandom)
- PTS for logics
  - Gentzen, Prawitz, Dummett, Martin-Löf, …
  - Eg, meaning theory for TTs (Martin-Löf & others, eg, Dybier)
- PTS for NLs
  - Not much work so far
    - Francez's early work with Dyckhoff (but problem with scaling up etc.)
  - Traditional divide of MTS & PTS has a misleading effect.

# MTT-semantics being proof-theoretic

- MTT-semantics opens up new possibility!
- MTTs are representational languages with proof-theoretic semantics themselves.
  - This was not available before.
- MTT-based proof technology
  - MTTs can be implemented.
  - Reasoning based on MTT-semantics can be carried out in proof assistants like Coq.

- Coq formalisation
  - Pretty straightforward and simple, whence MTT-semantic treatment is given.
  - But this is a nice application of proof technology to NL reasoning (a not-so-straightforward business in the past …)
- Some Coq codes can be found in:
  - Z. Luo. Contextual analysis of word meanings in type-theoretical semantics. Logical Aspects in Computational Linguistics. 2011.
  - S. Chatzikyriakidis & Z. Luo. NL Inference in Coq. 23(4), JoLLI. 2014.
  - S. Chatzikyriakidis & Z. Luo. Proof assistants for NL semantics. LACL16.
  - S. Chatzikyriakidis & Z. Luo. On the Interpretation of Common Nouns: Types v.s. Predicates. In CL (eds). Modern Perspectives in TTS. 2017.

  with new codes for recent developments.

# Several recent developments

- Dependent types in event semantics
  - Davidson’s event semantics and its new-Davidsonian turn
  - Dependent event types [Luo & Soloviev 2017]
- Negative sentences & conditionals
  - Judgemental v.s. predicational forms
    - “Tables don’t talk.”
  - NOT for negatives/conditionals
    - NOT in [Chatzikyriakidis & Luo 2017], today a different NOT2
    - $\forall$x:Table. NOT2(Human,talk,Table,x), where talk : Human→Prop.
    - To be justified by heterogeneous/John Major equality (in progress)
- CNs as setoids
  - Reasoning in more sophisticated situations (work in progress)

*Note: Coq experiments done for all of the above.*

# Example: NOT for negative/conditional sentences

```
(* NOT2(A,p,B,b) ~ "b does not p" *)
Parameter NOT2 : forall A:CN, (A->Prop) -> forall B:CN, B->Prop.

(* "Tables don't talk." *)
Variables Human Table : CN.  Variable talk : Human -> Prop.
Definition tableNOT2talk  := forall t:Table, NOT2 Human talk Table t.

(* p_A & NOT2(A,p_A,B,b) ~ "b is not A" *)
Definition pr (A:Type)(a:A) := True.

(* "Mary is not a man." *)
Variables Man Woman : CN.  Variable m : Woman.
Definition mNOT2Man : Prop := NOT2 Man (pr Man) Woman m.
```

```
(* Laws for NOT2 *)

(* L1: if x:A, ¬NOT2(A,p_A,A,x) is true *)
Definition L1 := forall (A:CN)(x:A), not (NOT2 A (pr A) A x).

(* L2: if A<B, c is not B => c is not A *)
Variables A B C : CN.
Variable cAB : A->B. Coercion cAB : A >-> B.
Definition L2 := forall c:C, NOT2 B (pr B) C c -> NOT2 A (pr A) C c.
```

# NOT: Justified by Heterogeneous Equality

```
(* Justification of NOT2 by JMeq *)
Require Import Coq.Logic.JMeq.

(* NOT2 defined by means of JMeq  *)
Definition NOT2 (A:CN)(p:A->Prop)(B:CN)(b : B) := forall x:A, JMeq x b->not (p x).

(* p_A *)
Definition pr (A:Type)(a:A) := True.
```

# Two laws for NOT

```
(* L1: if x:A, ¬NOT2(p_A(x)) is true *)
Definition L1 := forall (A:CN)(x:A), not (NOT2 A (pr A) A x).
Lemma l1 : L1.
unfold L1. unfold NOT2. unfold not. intros. apply (H x). auto. unfold pr. auto. Qed.

(* L2: if A<B, c is not B => c is not A *)
Variables A B C : CN.
Variable cAB : A->B. Coercion cAB : A >-> B.
Axiom subsumptionAB : forall x:A, JMeq x (cAB x).

Definition L2 := forall c:C, NOT2 B (pr B) C c -> NOT2 A (pr A) C c.
Lemma l2 : L2.
unfold L2. unfold NOT2. intros c cNOTB. intros a JMac. apply (cNOTB (cAB a)).
apply JMeq_sym. apply (JMeq_trans (JMeq_sym JMac) (subsumptionAB a)). Qed.
```

# Remark

- We do not employ heterogeneous equality directly, for it "overgenerates" in the sense that it would allow meaningless sentences to have eligible semantics.