# Modern Type Theoretical Semantics: Reasoning Using Proof-Assistants

Stergios Chatzikyriakidis
Centre for Linguistic Theory and Studies in Probability, University of Gothenburg

August 27, 2015

# Structure of the talk

- Intro to Modern Type Theoretical Semantics
  - MTT semantics for NL semantics
    - Some test cases: Modification
  - Inference Using Proof-Assistant Technology
    - Coq as an NL reasoner
  - Future work

# A brief intro to Modern Type Theories (MTTs)

- Type Theories within the tradition of Martin Löf
  - In linguistics, this work has been initiated by pioneering work of Ranta (1994)
- Here, we use one such MTT, UTT, first applied by Luo (2010) to the study of linguistic semantics
  - Two characteristics that are promising in using MTTs as an alternative formal semantics language:
    - ★ Consistent internal logic according to the propositions-as-types principle
    - ★ Rich type structures

# Intro to MTTs-Type Many Sortedness and Rich Typing

- Many-sortedness of types
  - Use of many types to interpret CNs, *man* and *table*
  - CNs are interpreted as Types rather than as predicates $(e \rightarrow t)$
- Use of Dependent Types $\Pi$ and $\Sigma$
  - When $A$ is a type and $P$ is a predicate over $A$, $\Pi x{:}A.P(x)$ is the dependent function type that stands for the universally quantified proposition $\forall x{:}A.P(x)$
  - $\Pi$ for polymorphic typing: $\Pi A{:}CN.(A \rightarrow Prop) \rightarrow (A \rightarrow Prop)$
  - $A$ is a type and $B$ is an $A$-indexed family of types, then $\Sigma x{:}A.B(x)$, is a type, consisting of pairs $(a, b)$ such that $a$ is of type $A$ and $b$ is of type $B(a)$.
  - Adjectival modification as involving $\Sigma$ types (Ranta, 1994; Luo, 2010): $heavybook = \Sigma x : book.heavy(x)$

# Intro to MTTs-Subtyping

- Coercive subtyping
  - Can be seen as an abbreviation mechanism
    - ⋆ $A$ is a (proper) subtype of $B$ ($A < B$) if there is a unique implicit coercion $c$ from type $A$ to type $B$
    - ⋆ An object $a$ of type $A$ can be used in any context $\mathfrak{C}_B[\_]$ that expects an object of type $B$: $\mathfrak{C}_B[a]$ is legal (well-typed) and equal to $\mathfrak{C}_B[c(a)]$.
    - ⋆ For example assuming $man < human$, $John : man$ and $shout : human \rightarrow Prop$, then $shout(John)$ is well-typed.

# Intro to MTTs-Universes

- Universes
  - A universe is a collection of (the names of) types into a type (Martin Löf, 1984).
  - Universes can help semantic representations. For example, one may use the universe $\mathrm{CN} : \mathit{Type}$ of all common noun interpretations and, for each type $A$ that interprets a common noun, there is a name $\overline{A}$ in $\mathrm{CN}$. For example,

    $$\overline{man} : \mathrm{CN} \quad \text{and} \quad T_{\mathrm{CN}}(\overline{man}) = man.$$

    In practice, we do not distinguish a type in $\mathrm{CN}$ and its name by omitting the overlines and the operator $T_{\mathrm{CN}}$ by simply writing, for instance, $man : CN$.

# Modification

- Adjectival modification as involving $\Sigma$ types, in line with Ranta (1994)
  - Intersective adjectives as simple predicate types and subsective as polymorphic types over the CN universe:
    - ★ $[\![black]\!] : Object \rightarrow Prop$
    - ★ $[\![small]\!] : \Pi A{:}CN.A \rightarrow Prop$ (the $A$ argument is implicit)
    - ★ For *black man*, we have: $\Sigma m{:}\,[\![man]\!]\,.\,[\![black]\!](m) < [\![man]\!]$ (via $\pi_1$)
    - ★ $< \Sigma m{:}\,[\![human]\!]\,.\,[\![black]\!](m)$ (via subtyping propagation)
    - ★ $< [\![human]\!]$ (via $\pi_1$)
  - For *small man*:
    - ★ $\Sigma m{:}\,[\![man]\!]\,.\,[\![small]\!]\,[\![man]\!](m) < [\![man]\!]$ (via $\pi_1$)
    - ★ BUT NOT:
      $\Sigma m{:}\,[\![man]\!]\,.\,[\![small]\!]\,[\![man]\!](m) < \Sigma m{:}\,[\![animal]\!]\,.\,[\![small]\!]\,[\![man]\!](m)$
    - ★ Many instances of small: $small([\![man]\!])$ is of type $[\![man]\!] \rightarrow Prop$, $small([\![animal]\!])$ is of type $[\![animal]\!] \rightarrow Prop$

# Adjectival Modification/More Advanced Issues

- Privative adjectives like *fake*
  - We follow Partee (2007) and argue that privative adjectives are actually subsective adjectives which operate on CNs with extended denotations
    - For exaple, the denotation of *fur* is expanded to include both *real* and *fake* furs:

      (1) I don't care whether that fur is fake fur or real fur.

      (2) I don't care whether that fur is fake or real.

    - $G = G_R + G_F$ with $inl(r){:}G_R$ and $inl(f){:}G_F$
    - Injections as coercions: $G_R <_{inl} G \texttt{and} G_F <_{inr} G$ and we define:
      $real\_gun(inl(r)) = True \quad and \quad real\_gun(inr(f)) = False$;
      $fake\_gun(inl(r)) = False \quad and \quad fake\_gun(inr(f)) = True$.
- Non-committal adjectives like *alleged*
  - Use of TT contexts representing beliefs (Ranta 1994)
    $[\![alleged\ N]\!] = \Sigma p{:}Human.\ B(p, A_N)$

# Adjectival Modification/More Advanced Issues

- Dealing with additional parameters: grades, temporal arguments
  - Use of indexed types
    - Basically, CNs with indexed arguments
    - For example, in order to reason about *height* in George is 1.60 tall, one needs to be able to refer to a *height* parameter
    - We define type $[\![Human]\!] : Height \to Prop$
    - $Human_i$ ($i{:}Height$) stands for humans indexed by $i$.
    - Gradable adjectives are defined as taking indexed CN arguments (e.g. $[\![short]\!] : Human_i \to Prop$)
  - Different degree parameters are needed (e.g. *height,size,width* or even abstract ones like *idiocy* (for example in he is a huge idiot))
    - Introduce a universe of degrees ($D$) that will contain all degree types
    - All types in the universe are totally ordered, anti-symmetric, reflexive and dense

# Adjectival Modification/An example: *tall*

- We first use the auxiliary object *TALL* and then define *tall* to be its first projection, $\pi_1$
  - $SHORT : \Pi i : Height.\Sigma p : Human(i) \rightarrow Prop.\forall h_1{:}Human(i).$ $p(h_1) \leftrightarrow i < n$
  - $[\![short]\!](i) = \pi_1(SHORT(i)) : Human(i) \rightarrow Prop$
- $n$ is a contextual parameter, the standard value provided by the context
- $[\![STND]\!] = \lambda A{:}\mathrm{CN}.\lambda i{:}D.\lambda P{:}A_i \rightarrow Prop.\exists n_1{:}Nat.n_1 = n \wedge i <> n$
  - *short* basically returns the first component of the pair $SHORT(i)$ of type $Human_i \rightarrow Prop$
  - The inference *John is tall* $\Rightarrow$ *John is taller than the standard value* follows from the second component of $SHORT(i)$
    - Assuming that $tall(John_i)$ is $p(h_1)$, $i < n$ follows
    - Similar account for comparatives: Instead of a relation between an $i$ and the standard, we have a relation between $i$ and $j$ provided by two arguments $Human_i$ and $Human_j$

# Adjectival Modification/Multidimensional Adjectives

- Quantification across different dimensions
  - E.g. to be considered *healthy* one has to be healthy w.r.t a number of dimensions (blood pressure, cholesterol etc.)
    - Involves universal quantification over dimensions
  - The antonyms of these type of multidimensional adjectives existentially quantify over dimensions
    - For one to be sick, only one dimension is needed
- We formulate this idea by Sassoon (2008) as follows:
  - We define an inductive type *health*
    - $Inductive\ [\![Health]\!] : D := Heart — Blood\_pressure — Cholesterol$
  - Then we define:
    - $[\![healthy]\!] = \lambda x{:}Human.\forall h{:}Health.Healthy(h)(x)$
    - $[\![sick]\!] = \lambda x{:}Human.\neg(\forall h{:}Health.Healthy(h)(x))$

# Adverbial Modification

- Typing issues: How are we going to type adverbs in a many sorted TT?
  - Two basic types
    - Sentence adverbs: $Prop \to Prop$
    - VP-adverbs: $\Pi A{:}CN.(A \to Prop) \to (A \to Prop)$
    - Polymorphic type: Depends on the choice of $A$
    - Given that we are talking about predicates, depends on the choice of the argument
    - $[\![walk]\!] : Animal \to Prop \Rightarrow [\![ADV]\!]\,[\![walk]\!] : (Animal \to Prop)$
    - $[\![drive]\!] : Human \to Prop \Rightarrow [\![ADV]\!]\,[\![drive]\!] : (Human \to Prop)$

# Adverbial Modification: Veridicality

- Veridical Adverbials when applied to their argument, imply their argument
  - John opened the door quickly $\Rightarrow$ John opened the door
  - Fortunately, John is an idiot $\Rightarrow$ John is an idiot
- Non-veridical adverbs do not have this property
  - John allegedly opened the door $\not\Rightarrow$ John opened the door

# Adverbial Modification: Veridicality

- We can use a similar organizational strategy as in the case with adjectives
  - Define an auxiliary object first, define the adverb as its first projection
    - ⋆ $[\![VER_{Prop}]\!] : \Pi v : Prop.\ \Sigma p : Prop.p \supset v$
    - ⋆ $[\![ADV_{ver-Prop}]\!] = \lambda v : Prop.\ \pi_1(VER_{Prop}(v))$
  - An adverb like *fortunately* will be defined:
  - $[\![fortunately]\!] = \lambda v : Prop.\ \pi_1(VER_{Prop}(v))$
- Consider the following: Fortunately, John went $\Longrightarrow$ John went
  - The second component of $(VER_{Prop}(v))$ is a proof of $[\![fortunately]\!](v) \Rightarrow v$
  - Taking $v$ to be $[\![John\ went]\!]$, the inference follows

# Adverbial Modification: Intensional/domain adverbials

- Use of TT contexts in this case as well
  - $[\![allegedly]\!] = \lambda P : Prop.\ \exists p : [\![human]\!], B_p(P)$
  - Someone has alleged that $P$ ($P$ is an agent's belief context (Chatzikyriakidis 2014; Chatzikyriakidis and Luo 2015))
- Introduction of intenTional contexts: Contexts including the intentions (rather than the beliefs) of an agent. We can use this idea for adverbs like intentionally:
  - $[\![Intentionally]\!] = \lambda x : [\![human]\!] . \lambda P : [\![human]\!] \rightarrow Prop.\ I_x(P(x)) \wedge \Gamma(P(x))$
- Domain adverbs, e.g. *botanically, mathematically*
  - $[\![botanically]\!] = \lambda P : Prop. \Gamma_B P$
- Intensionality without possible worlds

# The Coq proof-assistant

- An ideal tool for formal verification
  - Powerful and expressive logical language
  - Consistent embedded logic
  - Built-in proof tactics that help in the development of proofs
  - Equipped with libraries for efficient arithmetics in $N$, $Z$ and
  - Built-in automated tactics that can help in the automation of all or part of the proof process
  - Allows the definition of new proof-tactics by the user
    - ⋆ The user can develop automated tactics by using this feature

# MTT semantics in Coq

- Encoding MTT semantics based on theoretical work using Type Theory with Coercive Subtyping in Coq
  - Coq is a natural toolkit to perform such a task
    - ★ The type theory implemented in Coq is quite close to Type Theory with Coercive Subtyping
    - ★ Thus, the TT does not need to be implemented!
    - ★ What we need, is a way to encode the various assumptions as regards linguistic semantics and then reason about them

# Reasoning with NL

- As soon as NL categories are defined, Coq can be used to reason about them
  - In effect, we can view a valid NLI as a theorem
    - ⋆ Thus, we formulate NLIs as theorems
    - ⋆ The antecedent and consequent must be of type *Prop* in order to be used in proof mode
    - ⋆ Thus, the first can be formulated as a theorem, but not the second:

```
Theorem EX:(walk) John-> some Man (walk).
Theorem WA:walk -> drive.
```

# Reasoning with NL

- The same tactics that can be used in proving mathematical theorems are used for NL reasoning
  - The aim is to predict correct NLIs while avoiding unwanted inferences
    - ★ For example, given the semantics for quantifier *some*, one can formulate the following theorem and further try to prove it

```
Theorem EX:  (walk) John-> some Man (walk).
```

# An NLI example

- Basically, from a sentence like *John walks*, we should infer that *a man walks*
- We formulate the theorem

```
Theorem EX:  (walk) John-> some Man (walk).
```

- We unfold the definition for *some* and use *intro*

```
EX < intro.
1 subgoal
H : walk John
============================
exists x : Man, walk x
```

- We use the exists tactic to substitute $x$ for *John*. Using *assumption* the theorem is proven

# An NLI example

- To the contrary, we should not be able to prove the opposite

  ```
  Theorem EX: some Man (walk) -> (walk) John.
  ```

- Indeed, no proof can be found in this case.
  - We unfold *some* and use *intro*

    ```
    EX < intro.
    1 subgoal
    H : exists x : Man, walk x
    ============================
    walk John
    ```

  - From this point on, we can use any of the *elim, induction, case* tactics but at the end we reach a dead end

    ```
    EX < intro.
    1 subgoal
    H : exists x : Man, walk x
    x : Man
    H0 : walk x
    ============================
    walk John
    ```

# The FraCas test suite

- As already said, the examples involve a number of premises, followed by a question ($h$).
  - We reformulate the examples as involving declarative forms in Coq (this is a usual approach, at least with deep approaches)
    - In cases of *yes* in the FraCas test suite, we formulate a declarative hypothesis as following from the premise
    - In cases of *no*, we formulate the negation of a declarative hypothesis as following from the premise
    - In cases of *UNK*, for both the positive and the negated $h$, no proof should be found. If it is, we overgenerate inferences we do no want

# The FraCas test suite

- Quantifier monotonicity

  (3) Some Irish delegates finished the survey on time
  Did any delegates finish the survey on time? [YES]

  - Standard semantics for indefinites *some* and *any* (no presuppositions encoded)

```
Definition some:= fun A:CN, fun P:A->Prop=> exists x:A,
 P(x).
```

- $\Sigma$ types as dependent record types

```
Record Irishdelegate:CN:=mkIrishdelegate{c:> Man;_:
Irish c}.
```

- These assumptions suffice, the subtyping relation via $\pi_1$ does the trick here

```
Theorem IRISH: (some Irishdelegate(On_time(finish(the
survey)))->(some Delegate)(On_time (finish(the survey))).
```

# Quantifier monotonicity

- Monotonicity on the second argument

  (4) Some delegates finished the survey on time
  Did some delegates finish the survey? [UNK, FraCas 3.71]

- We define the auxiliary object and then *on_time*

```
Parameter ADV: forall (A : CN) (v : A -> Prop),sigT
(fun p : A -> Prop => forall x : A, p x -> v x).
Definition on_time:= fun A:CN, fun v:A->Prop=> projT1
(ADV(v)).
```

- These assumptions suffice for these cases

```
IRISH2 < Theorem IRISH2: (some delegate)(on_time
(finish(the survey)))->(some delegate)((finish
(the survey))).
```

# Adjectives

- Affirmative adjectives (these are intersective adjectives)

  (5) John has a genuine diamond
  Does John have a diamond? [Yes, FraCas 3.197]

  - The $\Sigma$ type approach will work here

- Opposites
  - Here we need to get:

  (6) $\text{Small(N)} \Rightarrow \neg\ \text{Large(N)}.$
  $\text{Large(N)} \Rightarrow \neg\ \text{Small(N)}$
  $\neg\ \text{Small(N)} \not\Rightarrow \text{Large(N)}.$
  $\neg\ \text{Large(N)} \not\Rightarrow \text{Small(N)}$

- The problem is that there are other sizes than a binary opposition *small-large*, e.g. normalsized items
- Use this intuition:

```
Definition small:= fun A:CN, fun a:A=> not (large (a)
/\ not (normalsized (a)).
```

# Adjectives

- Some examples that are correctly predicted with the definition given

(7) Mickey is a small animal
Is Mickey a large animal? [No, FraCas 3.204)

(8) Fido is not a large animal
Is Fido a small animal? [UNK, FraCas 3.207)

(9) All mice are small animals
Mickey is a large mouse
Is Mickey a large animal? [No, FraCas 3.210)

# Overview

- Evaluation against 30% of the suite
  - Extremely precise (more than 90% in all categories)
    - Full automation via user-defined tactics has been possible
  - Issue of recall: We have not yet have an automatic translation from the GF parser to the syntax of Coq (the translation was done manually)
  - In order to have a reliable measure of recall this needs to be done

# Ongoing and future work

- Define a translation from English to Coq sugared syntax (define a Coq concrete syntax in GF)
  - The syntax we need is a kind of quasi NL
    - For example we need (some man) walks for *some man walks* and (some man) (fast walks) for *some man walks fast*
    - The complexity will come in when Coq unfolds the definitions
- Evaluate against the whole suite or at least a very large frament
  - MacCartney has attempted the most until now: approx. 50% of the suite

# Ongoing and future work

- Semi automatic construction of TT semantics: Use of Lexical Network information to extract base types, predicates (Chatzikyriakidis et al. 2015)
  - WordNet for lexical semantics information for base types (subtyping, synonyms etc.)
  - Extract this information into Coq code is not difficult
  - More elaborate lexical networks can be used
    - We have used a GWAP constructed lexical network, jeuxdemots in Chatzikyriakidis et al. (2015)
- Entailment approximation?
- Can proof-automation be maintained when going for wider coverage?