**Overview**

- Formal semantics (lexical semantics, in particular)
- Typing – formal calculi with typing
  - How to capture “type presuppositions” with typing?
  - What is (is not) typing/subtyping?
- Which typing formalisms?
  - FOL with types
  - HOL as in Church’s Simple Type Theory (Montague)
  - TCL (Asher 2011)
  - Other formalisms?

---

**Proposal**

- Type-theoretic semantics
  - Formal semantics in Modern Type Theories
  - In particular, lexical semantics in MTTs
  - Why are MTTs useful?

Remarks:

1. Focussing on providing formal mechanisms for formal (lexical) semantics, not on empirical issues.
2. Another way to look at this: Formal realisation or implementation of the categorical ideas in earlier lectures. (Using seen examples etc.)

---

**These lectures**

- Basics of MTTs and type-theoretical semantics
- Lexical semantics in MTTs with coercive subtyping
  - Remark: Coercive Subtyping/Coercions are different from (but related to) and developed independently upon the notion of coercion in linguistics.
- Revisiting the linguistic issues in MTTs
  - “Type presuppositions” via typing
  - Copredication
  - Coercions (in linguistics)
  - Other issues (eg, sense enumeration/selection)

---

**I. Typing and Modern Type Theories**

- The typing relation (or judgement)
  \[ a : A \]
  Usually specified by means of a proof system.
  What can be “\(A\)” in “\(a : A\)?”

- Types:
  - \(\mathbb{N}\), \(\mathbb{L}ist(\mathbb{N})\), Table, Man, Man → Prop, Phy→Info, Phy→Info
  - Propositions (“propositions-as-types”): e.g.
    \[ \forall x : [\text{man}]. [\text{handsome}](x) \rightarrow [\text{ugly}](x) \]
  - Advanced types: dependent types, type universes (see later)

- What typing is not:
  - “\(a : A\)” is not a logical formula.
  - 7 : \(\mathbb{N}\)
  - Different from a logical formula \(\text{is}_{\text{nat}}(7)\)
  - Eg: Typing judgements (in intensional TTs) is decidable while the truth/provability of a formula (in FOL or a stronger calculus) is not.
  - “\(a : A\)” is different from the set-theoretic membership relation “\(a \in S\)” (the latter is a logical formula in FOL).

- What typing is related to:
  - Meaningfulness (it typed ⇒ meaningful)
  - Semantic/category errors (eg, “A table talks.”)
  - Type presuppositions (Asher 2011)
Types v.s. Sets

- Types may be thought of as "manageable sets".
- Some typical differences:
  - Typing is decidable: "a:A" is decidable (in intensional TTs), while the set membership "a∈S" is not.
  - Type theories can have an embedded/consistent logic, by propositions-as-types principle, while set theory is different.
  - There are union/intersection sets S∪S′/S∩S′, but subtyping A⊂B is much more restricted (and we do need a powerful subtyping mechanism for lexical semantics.)

Simple v.s. Modern Type Theories

- Church's simple type theory (Montague semantics)
  - Base types ("single-sorted"): e and t
  - Composite types: e→t, (e→t)→t, ...
  - Formulas in HOL (eg, membership of sets)
  - Eq: s:a→t is a set of entries (a∈S)

- Modern type theories (eg, Martin-Löf's type theory)
  - Many types of entities – "many-sorted"
  - Formulas in HOL (eg, membership of sets)
  - Composite types: e→t, (e→t)→t, ...
  - Prop – totality of logical propositions
  - Example – predicates to interpret adjectives/verbs:
    - handsome : Man→Prop
    - walk : Man→Prop

MTTs (1) – canonical objects

- Types v.s. Sets
  - Types may be thought of as "manageable sets".
  - Some typical differences:
    - Typing is decidable: "a:A" is decidable (in intensional TTs), while the set membership "a∈S" is not.
    - Type theories can have an embedded/consistent logic, by propositions-as-types principle, while set theory is different.
    - There are union/intersection sets S∪S′/S∩S′, but subtyping A⊂B is much more restricted (and we do need a powerful subtyping mechanism for lexical semantics.)

MTTs (2) – Embedded Logic

- Propositions-as-types
  - formula type example
    - x : B  A→B  x ∈ B
    - (∀a : A)(A(a))  ∀a : A(A(a))

MTTs (3): dependent/inductive types

- Σ-types – an example
  - Types of dependent pairs ("inductive": the only objects are pairs)
  - Intuitively, for A : Type and B : A→Type, Σ(A,B) = \{ (\langle x,y \rangle | x : A & y : B(x) ) \}
  - "(dependent") type B(x) depends on object x.
  - Example (when B is an A-indexed proposition):
    - Σ(Man, handsome)

- Other inductive types
  - finite types; nats, lists, vectors, trees, ordinals, ...

Types in MTTs: summary

- Propositional types
  - F:Q, ∀a.P(x)
- Inductive types
  - Nat, A:B, list(A), ...
- Dependent types
  - ∀a.B(a) (intuitively, (∀a)(a : A → B(a))
  - ∀a:B(a) (intuitively, (∀a : A → B(a))
- Universes
  - A universe is a type of (some other) types
  - Eq: CN – a universe of the types that interpret CNs (see later for an example of using this)
- Other types: Phy, Table, ..., A+B, ...
MTTs: example TTs

- Predicative type theories
  - Martin-Löf’s type theory
  - Extensional and intensional equalities in TTs
- Impredicative type theories
  - Prop
    - Impredicative universe of logical propositions (cf. \(1\) in simple TT)
    - Internal totality: a type, and can hence form types, e.g. Table\(\rightarrow\)Prop,
      Man \(\rightarrow\)Prop, \(\forall\)Prop, \(\forall\)Prop:X,
- F/P’ (Girard), CC (Coquand & Huet)
- ECC/VT (Luo, implemented in Lego/Plastic)
- pCIC (implemented in Coq/Plataan)

MTTs: Technology and Applications (in CS)

- Proof technology based on type theories
  - Proof assistants – ALF/Agda, Coq, Lego, NuPRL, Plastic, ...
- Applications
  - Formalisation of mathematics (e.g., 4-colour Theorem in Coq)
  - Program verification (e.g., security protocols)
  - Dependently-typed programming (Cayenne, DML, Epigram)

Here: type-theoretical lexical semantics

II. Type-Theoretical Semantics

- Type-theoretical semantics
  - Formal semantics in the Montagovian style
  - But, in modern type theories (not in simple TT)
    - Remark: proof-theoretic v.s. model-theoretic semantics of logical systems
- A key difference from the Montague semantics:
  - NPs interpreted as types (not predicates of type \(e\rightarrow t\))
- Some work on TT semantics
  - Ranta 1994: basics of TT semantics

Type-theoretical semantics (1): sentences & CNs

- Sentences (as propositions)
  - [John walks] : Prop
  - [A man walks] : Prop
- Common nouns are interpreted as types
  - [man], [book], [table] : Type (fine-grained)
- Remark: not as sets of type \(e\rightarrow t\) as in Montague semantics
- Other semantics types
  - E.g., Phy/Info – the type of physical/informational entities

Type-theoretical semantics (2): verbs

- Verbs are interpreted as predicates over “meaningful” domains
  - [shout] : [human] \(\rightarrow\)Prop
  - Note: “A table shouts” is meaningless (a “category error”) in the sense that
    - [table]: (shout)(t) is ill-typed (not “false”, as in Montague’s semantics).
  - We need:
    - [John shouts] = [shout](j) : Prop, for j : [man]
    - [A man shouts] = \(\exists\)m:[man]. [shout](m) : Prop
    - But these are ill-typed: [man] is not [human]!
- Subtyping:
  - [man] \(\subseteq\) [human], the above become well-typed.
  - Subtyping is crucial for type-theoretical semantics! (Things only work in the presence of subtyping.)

Montague Semantics: examples

- Sentences (as propositions)
  - [John walks] : [A man walks] : t
- Common nouns (as functional subsets of entities)
  - man : CN
  - [man] : e \(\rightarrow\) t
- Verbs (as subsets of entities)
  - walk : \(\exists\)t : e. [walk](t)
  - [A man walks] = \(\exists\)m:[man]. [walk](m)
- Adjectives (as functions from subsets to subsets)
  - handsome : [CN/CN] \(\rightarrow\) [CN]
  - [handsome man] : e \(\rightarrow\) t
  - [human], the above become well-typed.

Note: “A table shouts” is meaningless (a “category error”) in the sense that
- [[man], [book], [table]] : Type (fine-grained)
- [[[man], [book], [table]]] : Type (fine-grained)
- [[[man], [book], [table]]] : Type (fine-grained)
- [[[man], [book], [table]]] : Type (fine-grained)
TT semantics (3): adjectives & modified CNs

- Adjectives, like verbs, are interpreted as predicates over "meaningful" domains.
- [handsome]: [man] → Prop
- "A table is handsome" is meaningless (a "category error") in the sense that ∃ t: [table]. [handsome](t) is ill-typed

Modified CNs

- ∑-types for modified CNs
  - [handsome man] = ∑ (man, [handsome])
- Subtyping is needed as well (A handsome man is a man ...)
- More on subtyping later

Predicate-modifying adverbs: an advanced example

- Advanced features in MTTs are useful
- Montague semantics:
  - [quickly]: (e → t) → (e → t)
  - [John walked quickly] = [quickly]([walk], j) : Prop
- Remark: the above type of [quickly] is both polymorphic and dependent.

Remark on anaphora analysis

- Various treatments of "dynamics"
  - DRTs, dynamic logic, ...
  - MTTs provide a suitable (alternative) mechanism.
- Donkey sentences
  - Eq. "Every farmer who owns a donkey beats it..."
  - Montague semantics:
    - ∀x. farmer(x) & (∃y. donkey(y) & own(x,y)) ⇒ beat(x, y)
    - Modern TTs: (I for ∨ and ∑ for ∃):
      - 1x:Farmer(I[z:y:Donkey, own(x,y)]) beat(x,y, z)
  - But, this is only an interesting point ...

Type-theoretical lexical semantics: why/how?

- MTTs provide a promising formalism for
  - Formal semantics (basics as above)
  - Lexical semantics, in particular (next)
- Many promising mechanisms in MTTs to represent
  - Sense enumeration/selection model
  - Dot-types and copredication
  - Type presuppositions
  - Coercions (in linguistics)
  - and ... (other difficult cases)
- How?
  - Coercive subtyping etc

III. Coercive Subtyping

- Need for subtyping
  - Some subtypes of entities: Phy/Info ≤ e
  - More crucially needed for TT semantics
  - Many-sorted (CNs & modified CNs are interpreted as types)
  - Representation of relationships between these types is needed in TT semantics
- Coercive subtyping
  - Adequate (and powerful) framework for MTTs
  - Traditional "subsumption subtyping" is inadequate for MTTs
  - Coercive subtyping are very useful in lexical semantics.

Subtyping problem in the Montagovian setting

- Problematic example (in Montague semantics)
  - [heavy]: (Phy → t) → (Phy → t)
  - [book] : Phy → Info → t
  - [heavy book] = [heavy]([book])
  - In order for the above to be well-typed, we need PhyInfo → t ≤ Phy → t
    - By contravariance, we need Phy ≤ PhyInfo
      - But, this is not the case (the opposite is)!
    - In TT sem, because CNs are interpreted as types, things work as intended (see later).
Subsumptive subtyping: traditional notion

"Subsumptive subtyping":
\[ a : A \leq B \]

 Fundamental principle of subtyping
If \( A \leq B \) and, wherever a term of type \( B \) is required, we can use a term of type \( A \) instead.
For example, the subsumption rule realises this.

Question:
Is subsumptive subtyping adequate for type theories with canonical objects?
Answer:
No (canonicity fails) and then what?

Canonicity

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

This is a basis of TTs with canonical objects.
+ This is why the elimination rule is adequate.
+ Eg, Elimination rule for List(T):
  "For any family \( C \), if \( C \) is inhabited for all canonical T-lists \( \text{nil}(T) \) and \( \text{cons}(T,a,l) \), then so is \( C \) for all T-lists."

Coercive subtyping: basic idea

\( A \leq B \) if there is a coercion \( c \) from \( A \) to \( B \):

Eg. \( \text{Even} \leq \text{Nat}; \text{Man} \leq \text{Human}; \sum(\text{Man}, \text{handsome}) \leq \text{Man} \) ...

Subtyping as abbreviations:
\[ a : A \leq B \]
\[ C[a] = C(c(a)), \text{ ie, } "a" \text{ stands for } "c(a)" \]
Subtyping: basic need in TT semantics

- What about, eg, “A man is a human.”
- “A handsome man is a man”?
- “Paul walks”, with p : [Paul] : [handsome man]?

Solution: coercive subtyping

- \([\text{man}] \leq [\text{human}]\)
- \([\text{handsome man}] = \Sigma\{[\text{man}], [\text{handsome}]\} \leq [\text{man}]\)
- \([\text{Paul walks}] = ([\text{walk}])(p) : \text{Prop}\)

because

\[ [\text{walk}] : [\text{human}] \to \text{Prop} \]

\[ p : [\text{handsome man}] \leq [\text{man}] \leq [\text{human}] \]

IV. Coercive subtyping in TT semantics

1. Need for subtyping (earlier slides)
2. Sense enumeration/selection via. overloading
3. Coercion contexts and local coercions
4. Dot-types and copredication
5. Structured lexical entries as \(\Sigma\)-types

Notes:

- Focus on representation mechanisms, rather than NL semantic treatments.
- However, linguistic examples, rather than formal details.

Overloading via coercive subtyping

- Overloading can be represented by coercions

\[ \text{Eg} \quad \gamma_1 : [\text{run}] : \text{Human} \to \text{Prop} \]

\[ \text{run} : \text{I} \to \gamma_1 \]

- Homonymous meanings can be represented
- Automated selection according to typings

Question: What if typings cannot disambiguate (eg, bank)?
A solution: Local coercions

Coercive subtyping: adequacy etc.

- Inadequacy of subsumptive subtyping
  - Canonical objects
  - Conservativity: key for TTs with canonical objects
  - Subsumptive subtyping violates conservativity
- Adequacy of coercive subtyping
  - Coercive subtyping preserves conservativity & other properties.
- Historical development and applications in CS
  - Formal presentation (Luo 1997/1999)
  - Implementations in proof assistants: Coq, Lego, Plastic, Meta
Local coercions (in terms/judgements)

- Coercion $A \leq_c B$ in $t$
- Useful in disambiguation
  - E.g., "bank" has different meanings in
    1. the bank of the river
    2. the richest bank in the city
  - We might consider two coercions:
    
    $c_1 : \text{bank} \to \text{Type}$
    $c_1(\text{bank}) = \{\text{bank}\}_1$
    
    $c_2 : \text{bank} \to \text{Type}$
    $c_2(\text{bank}) = \{\text{bank}\}_2$
    
    But this is incoherent!

Solution: local coercions

Rather than two coercions for "bank" in the same context, (which is incoherent), we can use

$\text{coercion } 1_{\text{bank}} \leq_c \text{Type in (1)}$
$\text{coercion } 1_{\text{bank}} \leq_c \text{Type in (2)}$

4. Dot-types and copredication

- Dot-types in Pustejovsky’s GL theory
  - Example: $\text{PHY}\bullet\text{INFO}$
  - $\text{PHY}\bullet\text{INFO} \leq \text{PHY}$ and $\text{PHY}\bullet\text{INFO} \leq \text{INFO}$
- Copredication
  - "John picked up and mastered the book."
    - $\text{pick up} : \text{Human} \to \text{PHY} \to \text{Prop}$
    - $\text{master} : \text{Human} \to \text{INFO} \to \text{Prop}$

Remark: CNs as types in type-theoretical semantics – so things work.

Modelling dot-types in type theory

- What is $A\bullet B$?
  - Inadequate accounts (cf, (Asher 08)):
    - Intersection type
    - Product type
  - Proposal (SALT20, 2010):
    - $A\bullet B$ as type of pairs that do not share components
    - Both projections as coercions
  - Implementation
    - Being implemented in proof assistant Plastic by Xue.

Example

"heavy book"

- $[\text{heavy}] : \text{PHY} \to \text{Prop}$
  - $\text{PHY} \leq \text{PHY}\bullet\text{INFO}$
  - $\leq [\text{book}] \to \text{Prop}$
- So,
  - $[\text{heavy book}] = \Sigma([\text{book}], [\text{heavy}])$
  - is well-formed!
Another example

- Privative adjectives (cf. material modifiers)
  - Eg., “fake” in “fake gun”
  - A tentative proposal: use disjoint union types
    - Eg., \([\text{gun}]^* = [\text{real gun}] + [\text{fake gun}]\)
    - The injection operators inl/inr as coercions:
      \[\text{inl} : [\text{real gun}] \rightarrow [\text{gun}]^*\]
      \[\text{inr} : [\text{fake gun}] \rightarrow [\text{gun}]^*\]
  - “A fake gun is not a gun.”
  - In most of the cases, we do not want this!
    - Local coercions! (In the situations we do, use + and the associated coercions.)

5. Structured lexical entries

  Basic CNs represented by \(\Sigma\)-types, eg,
  \[
  \{\text{Agent} : \exists h : \text{human.}\ W(h, \text{Arg})\}
  \]

- Remarks
  - Should lexicon be complex/structured/generative?
  - Non-CN lexical entries: a general structure \((A, \phi)\)?
  - Cf. Cooper’s work on record types (2005, 2007)

Future work

- Some interesting topics
  - How well MTTs capture, eg, type presuppositions?
  - How far may a type-theoretical semantics go?
  - Proof-theoretic semantics for linguistic interpretations?

- Implementation for linguistic inference
  - Extending mathematical vernacular
  - Exploiting the existing TT-based proof technology

References (for Lectures 4/5)


(For further references, see the references of the associated notes.)