Type-Theoretical Lexical Semantics

Nicholas Asher and Zhaohui Luo
(Lectures 4/5 on Lexical Semantics at ESSLLI 2011)

Overview

- Formal semantics (lexical semantics, in particular)
- Typing – formal calculi with typing
  - How to capture linguistic data?
  - 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: eg,
    Nat, List(Nat), Table, Man, Man→Prop, Phy×Info, Phy•Info
  - Propositions ("propositions-as-types"): eg,
    ∀x:[man]. [handsome](x) → ¬[ugly](x)
  - Advanced types: dependent types, type universes (see later)

- What typing is not:
  - "a : A" is not a logical formula.
    - 7 : Nat
    - Different from a logical formula is_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∈S" (the latter is a logical formula in FOL).
- What typing is related to:
  - Meaningfulness (ill-typed ➔ meaningless)
  - Semantic/category errors (eg, "A table talks.")
  - Type presuppositions (Asher 2011)

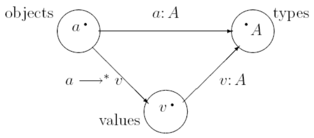
Types v.s. Sets

- 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 "$a \in S$" 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 $S \cup S'/S \cap S'$, but union or intersection types would usually make the type theory undecidable or logically inconsistent.
  - There is set inclusion $S \subseteq S'$, but subtyping $A \leq B$ 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: $e \to t$, $(e \to t) \to t$, …
  - Formulas in HOL (eg, membership of sets)
    - Eg, s : $e \to t$ is a set of entities ($a \in s$ iff s(a))
- Modern type theories (eg, Martin-Löf's type theory)
  - Many types of entities – "many-sorted"
    - Table, Man, Human, Phy, … are all types (of certain entities).
  - Different MTTs have different embedded logics
    - Martin-Löf's type theory: first-order logic (but not the standard one)
    - Impredicative UTT: higher-order logic (standard one)

MTTs (1) – canonical objects



Examples:

- A = N, a = 3+4, v = 7.
- A = N×N, a = (λx:N.⟨x,x+1⟩)(2), v = ⟨2,3⟩.

MTTs (2) – Embedded Logic

- Propositions-as-types

formula	type	example
$A \supset B$	$A \to B$	If …, then …
$\forall x{:}A.B(x)$	$\Pi x{:}A.B(x)$	Every man is handsome.

- Prop – totality of logical propositions
  - Impredicative type theories such as UTT [Luo 94] (as in *t* in Montague grammar)
  - Example – predicates to interpret adjectives/verbs:
    handsome : Man→Prop
    walk : Man→Prop

MTTs (3): dependent/inductive types

- Σ-types – an example
  - Types of dependent pairs ("inductive": the only objects are pairs)
  - <u>Intuitively</u>, for A : Type and B : A→Type,
    $\Sigma(A,B) = \{ \langle x,y \rangle \mid x : A \ \& \ y : B(x) \}$
    ("dependent": type B(x) depends on object x.)
  - Example (when B is an A-indexed proposition):
    Σ(Man, handsome)
  - Σ-types could be used (as in Martin-Löf's type theory) as existential formulas – but this is a non-standard strong quantifier.
- Other inductive types
  - finite types, nats, lists, vectors, trees, ordinals, …

Types in MTTs: summary

- Propositional types
  - $P \supset Q$, $\forall x{:}A.P(x)$, …
- Inductive types
  - Nat, AxB, List(A), …
- Dependent types
  - $\Sigma x{:}A.B(x)$ (intuitively, { (a,b) | a : A & b : B(a) })
  - $\Pi x{:}A.B(x)$ (intuitively, { f : $A \to \cup_{a \in A} B(a)$ | a : A & b : B(a) })
- Universes
  - A universe is a type of (some other) types
  - Eg, 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, t in simple TT)
    - Internal totality (a type, and can hence form types, eg Table→Prop, Man →Prop, ∀X:Prop.X,
  - F/F^{ω} (Girard), CC (Coquand & Huet)
  - ECC/UTT (Luo, implemented in Lego/Plastic)
  - pCIC (implemented in Coq/Matita)

MTTs: Technology and Applications (in CS)

- Proof technology based on type theories
  - Proof assistants – ALF/Agda, Coq, Lego, NuPRL, Plastic, …
- Applications
  - Formalisation of mathematics (eg, 4-colour Theorem in Coq)
  - Program verification (eg, 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:
  - CNs interpreted as <u>types</u> (not predicates of type e→t)
- Some work on TT semantics
  - Ranta 1994: basics of TT semantics
  - Luo 1998/2010/2011: coercive subtyping in TT semantics

Montague Semantics: examples

- Sentences (as propositions)
  - [John walks], [A man walks] : t
- Common nouns (as functional subsets of entities)
  - man : CN
  - [man] : e → t
- Verbs (as subsets of entities)
  - walk : IV
  - [walk] : e → t
  - [John walks] = [walk](j), if j = [John] : e.
  - [A man walks] = ∃m:e. [man](m) & [walk](m)
- Adjectives (as functions from subsets to subsets)
  - handsome : CN/CN
  - [handsome] : (e→t) → (e→t)
  - [handsome man] = [handsome]([man]) : e → t

Type-theoretic semantics (1): sentences & CNs

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

Type-theoretical semantics (2): verbs

- Verbs are interpreted as predicates over "meaningful" domains
  - [shout] : [human]→Prop
  - Note: "A table shouts" is meaningless (a "category error") in the sense that ∃t:[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] = ∃m:[man]. [shout](m) : Prop
  - But these are ill-typed! ([man] is not [human])
- Subtyping
  - [man] ≤ [human], the above become well-typed.
  - Subtyping is crucial for type-theoretical semantics! (Things only work in the presence of subtyping.)

TT semantics (3): adjectives & modified CNs

- Adjectives, like verbs, are interpreted as predicates over "meaningful" domains
  - [handsome] : [man]→Prop
  - Note: "A table is handsome" is meaningless (a "category error") in the sense that ∃t:[table].[handsome](t) is ill-typed (not "false", as in Montague's semantics).
- Modified CNs
  - Σ-types for modified CNs
  - [handsome man] = Σ([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
  - Semantics to adverbs: example of using type universes
- Montague semantics:
  - [quickly] : (e→t)→(e→t)
  - [John walked quickly] = [quickly]([walk], j) : t
- How in MTT?
  - Problem: We have many types that interpret CNs (Table, Man, Animated, …), not a single e.
  - Solution:
    - Introduce universe CN of types that interpret CNs
    - [quickly] : ∏A:CN. (A→Prop)→(A→Prop)
    - [John walked quickly] = [quickly]([animated], [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
  - Eg, "Every farmer who owns a donkey beats it."
  - Montague semantics
    ∀x. farmer(x) & [∃y. donkey(y) & own(x,y)]
    ⇒ beat(x,?y)
  - Modern TTs (Π for ∀ and Σ for ∃):
    Πx:FarmerΠz:[Σy:Donkey. own(x,y)] beat(x,π_1(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 "subsumptive 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
    Phy•Info→t ≤ Phy→t
    By contravariance, we need
    Phy ≤ Phy•Info
    But, this is <u>not</u> 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":

$$\frac{a : A \qquad A \leq B}{a : B}$$

- Fundamental principle of subtyping

  If A≤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.

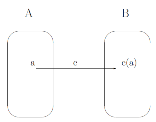
- This is a basis of TTs with canonical objects.
  - This is why the elimination rule is adequate.
  - Eg, Elimination rule for List(T):

    "For any family C, if C is inhabited for all canonical T-lists nil(T) and cons(T,a,l), then so is C for all T-lists."

- Canonicity is lost in subsumptive subtyping.
  - Eg, $\frac{A \leq B}{List(A) \leq List(B)}$
  - nil(A) : List(B), by subsumption;
  - But nil(A) ≠ any canonical B-list nil(B) or cons(B,b,l).
  - The elim rule for List(B) is inadequate: it does not cover nil(A) … …

Coercive subtyping: basic idea

- A≤B if there is a coercion c from A to B:



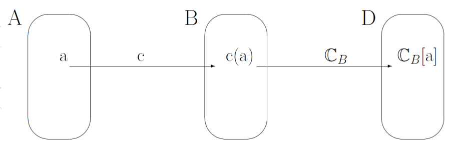
Eg. Even ≤ Nat; Man ≤ Human; Σ(Man, handsome) ≤ Man; …

- Subtyping as abbreviations:

  $a : A \leq_c B$

  ➔ "a" can be regarded as an object of type B

  ➔ $\mathbf{C}_B[a] = \mathbf{C}_B[c(a)]$, ie, "a" 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
  - [man] ≤ [human]
  - [handsome man] = Σ([man], [handsome]) $\leq_{\pi_1}$ [man]
  - [Paul walks] = [walk](p) : Prop
    because
    [walk] : [human]→Prop and
    p : [handsome man] $\leq_{\pi_1}$ [man] ≤ [human]

Coercive subtyping: adequacy etc.

- Inadequacy of subsumptive subtyping
  - Canonical objects
  - Canonicity: key for TTs with canonical objects
  - Subsumptive subtyping violates canonicity.
- Adequacy of coercive subtyping
  - Coercive subtyping preserves canonicity & other properties.
  - Conservativity (Luo & Soloviev 2002; Xue, Luo & Adams 2011)
- Historical development and applications in CS
  - Formal presentation (Luo 1997/1999)
  - Implementations in proof assistants: Coq, Lego, Plastic, Matita

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 Σ-types

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

2. Sense selection via overloading

- Sense enumeration (cf, Pustejovsky 1995 and others)
  - Homonymy
  - Automated selection
  - Existing treatments (eg, Asher et al via +-types)
- For example,
  1. John runs quickly.
  2. John runs a bank.

  with homonymous meanings
  1. $[run]_1$: Human→Prop
  2. $[run]_2$: Human→Institution→Prop

  "run" is overloaded – how to disambiguate?

Overloading via coercive subtyping

- Overloading can be represented by coercions

  Eg

  $run : 1_{run} \xrightarrow{c_1} [run]_1 : Human \rightarrow Prop$

  $run : 1_{run} \xrightarrow{c_2} [run]_2 : Human \rightarrow Institution \rightarrow Prop$
- Homonymous meanings can be represented.
- Automated selection according to typings

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

3. Coercion contexts and local coercions

- Coercion contexts

  x:C, ..., A $\leq_c$ B, ... ⊢ ... ...
- Useful in representing context-sensitivity
  - Eg, reference transfer

    The ham sandwich shouts.

    This can be interpreted in a context that contains, eg,

    [sandwich] < [human]

    that coerces sandwich into the person who has ordered a sandwich.

Remark: "coherent contexts" needed, not just valid contexts.
(Formal details omitted.)

- Local coercions (in terms/judgements)

  coercion A $\leq_c$ B in t
- Useful in disambiguation
  - Eg, "bank" has different meanings in

    (1) the bank of the river

    (2) the richest bank in the city
  - We might consider two coercions:

    $c_1 : 1_{bank} \to Type \quad c_1(bank) = [bank]_1$

    $c_2 : 1_{bank} \to Type \quad c_2(bank) = [bank]_2$

    But this is incoherent!

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

    coercion $1_{bank} \leq_{c1}$ Type in [(1)]

    coercion $1_{bank} \leq_{c2}$ Type in [(2)]

4. Dot-types and copredication

- Dot-types in Pustejovsky's GL theory
  - Example: PHY•INFO
  - PHY•INFO ≤ PHY and PHY•INFO ≤ INFO
- Copredication

  "John picked up and mastered the book."

  [pick up] : Human → PHY → Prop

  ≤ Human → PHY•INFO → Prop

  ≤ Human → [book] → Prop

  [master] : Human → INFO → Prop

  ≤ Human → PHY•INFO → Prop

  ≤ Human → [book] → Prop

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

Modelling dot-types in type theory

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

$$\frac{A : Type \quad B : Type \quad \mathscr{C}(A) \cap \mathscr{C}(B) = \emptyset}{A \bullet B : Type}$$

$$\frac{a : A \quad b : B}{\langle a, b\rangle : A \bullet B} \qquad \frac{c : A \bullet B}{p_1(c) : A} \qquad \frac{c : A \bullet B}{p_2(c) : B} \qquad \frac{a : A \quad b : B}{p_1(\langle a, b\rangle) = a : A} \qquad \frac{a : A \quad b : B}{p_2(\langle a, b\rangle) = b : B}$$

$$\frac{A \bullet B : Type}{A \bullet B <_{p_1} A : Type} \qquad \frac{A \bullet B : Type}{A \bullet B <_{p_2} B : Type}$$

Example

- "heavy book"
  - [heavy] : Phy→Prop

    ≤ Phy•Info→Prop

    ≤ [book]→Prop
  - So,

    [heavy book] = Σ([book], [heavy])

    is well-formed!

Another example

- Privative adjectives (cf, material modifiers)
  - Eg, "fake" in "fake gun"
  - A tentative proposal: use disjoint union types
    - Eg, [gun]* = [real gun] + [fake gun]
    - The injection operators inl/inr as coercions:
      inl/inr : [real gun]/[fake gun] → [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

- Proposal (1998, 2011):
  Basic CNs represented by Σ-types, eg,

$$[\![book]\!] = \Sigma \left\{ \begin{array}{lll} Arg & : & \text{PHY} \bullet \text{INFO} \\ Qualia & : & \Sigma \left\{ \begin{array}{lll} Formal & : & Hold(p_1(Arg), p_2(Arg)) \\ Telic & : & R(Arg) \\ Agent & : & \exists h{:}Human.W(h, Arg) \end{array} \right\} \end{array} \right\}$$

- Remarks
  - Should lexicon be complex/structured/generative?
  - Non-CN lexical entries: a general structure (A, φ)?
  - 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)

- N. Asher. *Lexical Meaning in Context: A Web of Words.* Cambridge University Press. 2011.
- Z. Luo. Coercive subtyping. J. of Logic and Computation, 9(1). 1999.
- Z. Luo. Type-theoretical semantics with coercive subtyping. SALT20. 2010.
- Z. Luo. Contextual analysis of word meanings in type-theoretical semantics. LACL 2011, LNAI 6736. 2011.
- A. Ranta. *Type-Theoretical Grammar.* Oxford University Press. 1994.

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