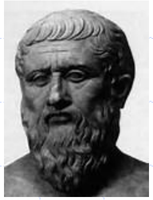
Modern Type Theories for NL Semantics

Zhaohui Luo

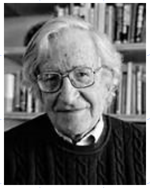
Royal Holloway, Univ. of London

Natural Language Semantics

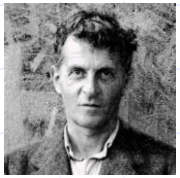
- Semantics – study of meaning (communicate = convey meaning)
- Various kinds of theories of meaning
  - Meaning is reference ("referential theory")
    - Word meanings are things (abstract/concrete) in the world.
    - c.f., Plato, …



  - 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.



Formal semantics

- Model-theoretic semantics
  - Meaning is given by denotation.
  - c.f., Tarski, …, Montague.
  - e.g., Montague grammar (MG)
    - NL → simple type theory → set theory
- Proof-theoretic semantics
  - In logics, meaning is inferential use (proof/consequence).
  - c.f., Gentzen, Prawitz, …, Martin-Löf.
  - e.g., Martin-Löf's meaning theory




Simple example for MTS and PTS

- Model-theoretic semantics
  - John is happy. ➔ happy(john)

    ➔ John is a member of the set of entities that are happy.
  - Montague's semantics is model-theoretic – it has a wide coverage (powerful).
- Proof-theoretic semantics
  - How to understand a proposition like happy(john)?
  - In logic, its meaning can be characterised by its uses – two respects:
    - How it can be arrived at (proved)?
    - How it can be used to lead to other consequences?

(*)

Montague's semantics and MTT-semantics

- Formal semantics (MG)
  - Montague Grammar Church's simple type theory (Montague, 1930–1971), dominating in linguistic semantics since 1970s
  - Other development of formal semantics in last decades (e.g., Discourse Representation Theory & Situation Semantics)
- MTT-semantics: formal semantics in modern type theories
  - Early use of dependent type theory in formal semantics (cf, Ranta 1994)
  - Recent development (since 2009) – full-scale alternative to MG
  - Advantages: both model/proof-theoretic, proof technological support, …
  - Refs at http://www.cs.rhul.ac.uk/home/zhaohui/lexsem.html, including
    - Z. Luo. Formal Semantics in MTTs with Coercive Subtyping. Ling & Phil, 35(6). 2012.
    - 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)

TTs as foundational languages for NL semantics

- What is a type theory?
  - a : A
    - a is an object of type A
    - the most basic "judgement" to make in type theory
  - The worlds of types – examples:
    - Simply typed λ-calculus (with A→B)
    - Church's simply type theory as in Montague's semantics (A→B with HOL of formulas like $P \supset Q$ and $\forall x{:}A.P$)
    - Richer types (eg, in MTTs: dependent, inductive, …; see latter)
  - Logical language (often part of type theory)
    - In Church/Montague: formulas & provability/truth
    - In modern type theories (MTTs): formulas-as-types & proofs-as-objects

      E.g., $\forall x{:}Man.\ handsome(x) \rightarrow \neg ugly(x)$ can be seen as a type (later)

- What typing is not:
  - "a : A" is not a logical formula.
    - 7 : Nat, j : Man, …
    - Different from logical formulae nat(7)/man(j), where nat/man are predicates. (Note: whether a formula is true is undecidable, while the :-judgements are.)
  - "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 (some example notions):
  - Meaningfulness (ill-typed → meaningless)
  - Semantic/category errors (eg, "A table talks." – later)
  - Type presuppositions (Asher 2011)

This course – MTTs in NL semantics

- MTTs – Modern Type Theories
  - Rich type structures
    - much richer than simple type theory in MG
  - Proof-theoretically specified by rules
    - proof-theoretic meanings (e.g., Martin-Löf's meaning theory)
  - Embedded logic
    - based on propositions-as-types principle
- Informally, MTTs, for NL semantics, offer
  - "Real-world" modelling as in model-theoretic semantics
  - Effective inference based on proof-theoretic semantics

Remark: New perspective & new possibility not available before!

An episode: 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 (on map colouring) in Coq
    - Kepler conjecture (on sphere packing) in Isabelle/HOL
  - Computer Science:
    - program verification and advanced programming
  - Computational Linguistics
    - E.g., MTT-sem based NL reasoning in Coq (Chatzikyriakidis & Luo 2014)

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

A focus of the course

- However, this course
  - is not one on MTT-semantics only;
  - is one on MTTs with examples in MTT-semantics!
- Reason for this focus:
  - Learning MTTs is laborious, even for logic-oriented semanticists
  - New logical concepts: judgement, context, inductive & dependent types, universe, subtyping, …
  - Hope: making learning MTTs (hence MTT-semantics) easier!
- Goal: learning MTTs as well as MTT-semantics

Overview of the Course

- This lecture:
  - Introduction to MTT-semantics (a first taste)
- Each lecture from L2-5 will consist of two parts:
  - Some key MTT concepts/mechanisms
  - Introduction of some MTT types with several applications in MTT-semantics.
  - Example: Lecture 2 of "Judgements and Π-polymorphism" introduces these in MTTs and then uses Π-polymorphism to model coordination, predicate-modifying adverbs (quickly) and subsective adjectives (large).
- Goal: learn MTTs with examples in MTT-semantics

- Material available on the web:
  - Lecture slides
  - Course proposal (good summary, but the organisation and descriptions of lectures are)
  - Papers/books on MTT-semantics available at

    http://www.cs.rhul.ac.uk/home/zhaohui/lexsem.html

I. Type-theoretical semantics: introduction

- Introduction to MG and MTT-semantics, starting with examples
- Two basic semantic types in MG/MTT-semantics

Category	**MG's type**	**MTT-semantic type**
S (sentence)	t	Prop
IV (verb)	$e \to t$	$A \to Prop$ (A: "meaningful domain")

Simple example

- John talks.
  - Sentences are (interpreted as) logical propositions.
  - Individuals are entities or objects in certain domains.
  - Verbs are predicates over entities or certain domains.

	Montague	**MTT-semantics**
john	e	Human
talk	$e \rightarrow t$	$Human \rightarrow Prop$
talk(john)	t	Prop

Three issues: a first taste

- Selection restriction
  - (*) The table talks.
  - Is (*) meaningful?
  - In MG, yes: (*) has a truth value
    - talk(the table) is false in the intended model.
  - In MTT-semantics, no: (*) is not meaningful
    - since "the table" : Table and it is not of type Human and, hence, talk(the table) is ill-typed as talk requires that its argument be of type Human.
    - So, in MTT-semantics, meaningfulness = well-typedness

- **Subtyping**
  - Necessary for a multi-type language such as MTTs
  - Example: What if John is a man in “John talks”?
    - john : Man
    - talk : Human→Prop
    - talk(john)? (john is not of type Human …?)
  - Problem solved if Man ≤ Human
    - A ≤ B and a : A ➔ a : B
    - Man ≤ Human and john : Man ➔ john : Human
    - Hence, talk(john) : Prop

Later (Lecture 3): “coercive subtyping”, and we use it in modelling various linguistic features such as sense selection & copredication.

- Propositions as types in MTTs
  - Formula A is provable/true if, and only if, there is a proof of A, i.e., an object p of type A (p : A).

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

- MTTs have a consistent logic based on the propositions-as-types principle.

❖ Two more basic MG/MTT-semantic types

Category	**MG's Type**	**MTT-semantic type**
S	t	Prop
IV	$e \rightarrow t$	$A \rightarrow Prop$
CN (book, man)	$e \rightarrow t$	types (Book, Σx:Man.handsome(x))
Adj (CN/CN)	$(e \rightarrow t) \rightarrow (e \rightarrow t)$ or $e \rightarrow t$	$A \rightarrow Prop$ (A: meaningful domain)

Adjective modifications of CNs

- One of the possible/classical classifications:

classification	**property**	**example**
Intersective	Adj(N) ➔ Adj & N	handsome man
Subsectional	Adj(N) ➔ N	large mouse
Privative	Adj(N) ➔ $\neg$N	fake gun
Non-committal	Adj(N) ➔ ?	alleged criminal

Intersective adjectives

- Example: handsome man

	Montague	MTT-semantics
man	man : $e \rightarrow t$	Man : Type
handsome	handsome : $e \rightarrow t$	Man$\rightarrow$Prop
handsome man	$\lambda x.$ man(x) & handsome(x)	Σ(Man,handsome)

- In general:

	Montague	MTT-semantics
CNs	predicates	types
Adjectives	predicates	predicates
CNs modified by intersective adj	Predicate by conjunction	Σ-type

- adjective : CNs → CNs
  - In MG, predicates to predicates.
  - In MTT-semantics, types to types.
- Proposals in MTT-sem (Chatzikyriakidis & Luo, FG13 & JoLLI17)

classification	**example**	**types employed**
Intersective	handsome man	Σ-types (of pairs)
Subsectional	large mouse	Π-types (polymorphism)
Privative	fake gun	disjoint union types
Non-committal	alleged criminal	belief contexts

Σ-types: a taste of dependent types

- First, we start with "product types" of pairs:
  - A x B of pairs (a,b) such that a:A and b:B
  - Rules to specify these product types:
    - Formation rule for A x B
    - Introduction rule for pairs (a,b) : A x B
    - Elimination rules for projections $\pi_1(p)$ and $\pi_2(p)$
    - Computation rule: $\pi_1(a,b)=a$ and $\pi_2(a,b)=b$.
- This generalises to Σ-types of "dependent pairs" (next page)

- “Family” of types
  - Tyoe-valued function
  - Dog(John) = {d}, Dog(Mary)={d1, d2}, …
  - Dog : Human→Type
- Σ-types of “dependent pairs”:
  - Σ(A,B) of dependent pairs (a,b) such that a:A and b:B(a), where A:Type and B : A→Type.
  - Rules for Σ-types:
    - Formation rule for Σ(A,B) for B : A→Type
    - Introduction rule for dependent pairs (a,b) : Σ(A,B)
    - Elimination rules for projections $\pi_1(p)$: A and $\pi_2(p)$: $B(\pi_1(p))$
    - Computation rule: $\pi_1(a,b)=a$ and $\pi_2(a,b)=b$.

- "handsome man" is interpreted as type

$$\Sigma(Man,handsome)$$

- So,
  - A handsome man is an object of the above type
  - It is a pair (m,p) such that m : Man and p : handsome(m), i.e., m is a man and p is a proof that m is handsome.

II. Judgements and Π-polymorphism

II.1. Overview of Modern Type Theories

- Difference from simple type theory
- Example MTTs
- Judgements (basic "statements" in MTTs)

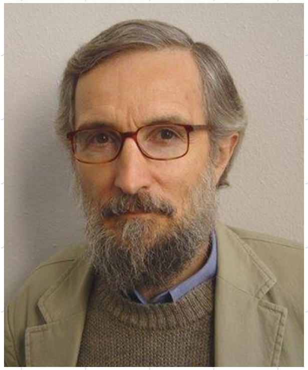
II.2. Dependent product types (Π-types)

- Basic constructions
- →-types as special cases of Π-types (examples in semantics)

II.3. Universes – Π-polymorphism and examples like

- Coordination
- Quantifiers and Adverbs (predicate modifying)
- Subsective adjectives (e.g., large)

II.1. Modern Type Theories: overview



- 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')
- Modern type theories
  - Many types of entities – "many-sorted"
    - Table, Man, Human, Σx:Man.handsome(x), Phy•Info, …
  - Dependent types: "types segmented by indexes"
    - List ➔ Vect(n) with n:Nat (lists of length n)
    - Event ➔ Evt(h) with h:Human (events performed by h)
  - Examples of MTTs:
    - Martin-Löf's TT (predicative; non-standard FOL; proof assistants Agda/NuPRL)
    - CIC_p (Coq) & UTT (Luo 1994) (impredicative; HOL; Coq/Lego/Plastic/Matita)

Predicativity/impredicativity: technical jargon

- This refers to a possibility of forming a logical proposition "circularly":
  - $\forall X{:}Prop.X : Prop$
  - Quantifying over all propositions to form a new proposition.
  - Is this OK? Martin-Löf thinks not, while Ramsey (1926) thinks yes (it is circular, but it is not vicious.)
- Allowing the above leads to impredicative type theories, which have in particular, Prop:
  - Impredicative universe of logical propositions (cf, t in MG)
  - Internal totality (a type, and can hence form types, eg $Table \rightarrow Prop$, $Man \rightarrow Prop$, $\forall X{:}Prop.X$, …)

Judgements: MTTs' statements

- A statement in an MTT is a judgement, one of whose forms (the most important form) is

  $$(*) \quad \Gamma \vdash a : A$$

  which says that "a is of type A under context Γ".
- Types represent collections (they are different from sets, although they both represent collections) or propositions.
- $\Gamma \equiv x_1 : A_1, \ldots, x_n : A_n$ is a context, which is a sequence of "membership entries" declaring that x_i is a variable of type A_i.
  - When Γ is empty, (*) is non-hypothetical; (in this case, we may just write a : A by omitting "$\Gamma \vdash$".)
  - When Γ is non-empty, (*) is hypothetical.

Examples of judgements

- John is a man.
  - ➔ john : Man, where Man is a type. (non-hypothetical)
- If John is a student, he is happy.
  - ➔ j : Student $\vdash$ p : happy(j)   (for some p) (hypothetical)
- Truth of a formula:
  - "happy(j) true"
  - The above is a shorthand for "p : happy(j) for some p"

Other forms of judgements (1)

- Γ valid
  - Γ is a valid ("legal") context
  - When is $\Gamma \equiv x_1 : A_1, \ldots, x_n : A_n$ valid? (1) x_i's are different; (2) A_i's are types in the prefix on their left.
- Question:
  - Why is this necessary?
  - In traditional logics, we do not need this – just consider a set of formulas – this would seem enough …
  - Answer: because we have dependent types – it is possible that x_i's occur freely in the A_j's after them!
  - Eg, we can have a context

$$x{:}Man, \ldots, y{:}handsome(x), \ldots$$

Situations represented as contexts: an example

- Beatles' rehearsal
  - Domain: $\Sigma_1 \equiv D : Type,$
    $$John : D,\ Paul : D,\ George : D,\ Ringo : D,\ Brian : D,\ Bob : D$$
  - Assignment: $\Sigma_2 \equiv B : D \to Prop,\ b_J : B(John),\ ...,\ b_B : \neg B(Brian),\ b'_B : \neg B(Bob),$
    $$G : D \to Prop,\ g_J : G(John),\ ...,\ g_G : \neg G(Ringo),\ ...$$
  - Context representing the situation of Beatles' rehearsal:
    $$\Sigma \equiv \Sigma_1,\ \Sigma_2,\ ...,\ \Sigma_n$$
  - We have, for example,
    $$\Sigma \vdash G(John)\ \text{true} \quad \text{and}\ \Sigma \vdash \neg B(Bob)\ \text{true}$$
    i.e., under Σ, "John played guitar" & "Bob was not a Beatle".

Other forms of judgements (2)

- $\Gamma \vdash A$ type
  - A is a type under Γ.
  - E.g. when is AxB or $\Sigma x{:}A.B$ a valid type?
- $\Gamma \vdash A = B$ and $\Gamma \vdash a{=}b : A$ (equality judgements)
  - A and B are (computationally) the same types.
  - a and b are (computationally) the same objects of type A.
  - E.g., do we have $\pi_1(a,b)=a$?

Now let's illustrate by types of pairs.

Σ-types: a taste of dependent types

- First, we start with "product types" of pairs:
  - A x B of pairs (a,b) such that a:A and b:B
  - Rules to specify these product types:
    - Formation rule for A x B
    - Introduction rule for pairs (a,b) : A x B
    - Elimination rules for projections $\pi_1(p)$ and $\pi_2(p)$
    - Computation rule: $\pi_1(a,b)=a$ and $\pi_2(a,b)=b$.
- This generalises to Σ-types of "dependent pairs" (next page)

- “Family” of types
  - B[x] type – type “indexed” by x : A
  - Dog[x] type for x : Human
  - Dog[John] = {d}, Dog[Mary] = {d_1, d_2}, …
    (Here, {…} are finite types.)
- Σ-types of “dependent pairs”:
  - Σx:A.B[x] of dependent pairs (a,b) such that a:A and b:B[a].
  - Rules for Σ-types:
    - Formation rule for Σx:A.B
    - Introduction rule for dependent pairs (a,b) : Σx:A.B[x]
    - Elimination rules for projections $\pi_1(p) : A$ and $\pi_2(p) : B[\pi_1(p)]$
    - Computation rule: $\pi_1(a,b)=a$ and $\pi_2(a,b)=b$.

- "handsome man" is interpreted as type

$$\Sigma x{:}Man.handsome(x)$$

- So,
  - A handsome man is an object of the above type.
  - It is a pair (m,p) such that m : Man and p : handsome(m), i.e., m is a man and p is a proof that m is handsome.

Judgements v.s. Formulas/Types

- First, judgements are <u>not</u> formulas/propositions.
  - Propositions correspond to types (P in p : P).
  - For example, "P is true" corresponds to "p : P for some p".
- You may think judgements as meta-level statements that cannot be used "internally".
  - For example, unlike a formula, you cannot form, for example, $\neg$J for a judgement J.
  - This is similar to subtyping judgements $A \leq B$. Such assumptions may be considered in "signatures" – see my LACL14 invited talk/paper and work in Lungu's thesis (2017).

We stop here: Further discussions are out of the scope here, but relevant papers are available, if requested.

II.2. Dependent product types (Π-types)

- Informally (borrowing set-theoretical notations, formal rules next slide),

  $\Pi x{:}A.B[x] = \{\ f \mid \text{for any } a : A,\ f(a) : B[a]\ \}$

- Examples
  - $\lambda x{:}Nat.[1,\ldots,x] : \Pi x{:}Nat.Vect(x)$
  - $\forall x{:}Student.\ work\_hard(x)$
    - This is just another notation for $\Pi x{:}Student.\ work\_hard(x)$
  - $\forall x{:}Man.\ handsome(x) \supset \neg ugly(x)$
- Notational conventions:
  - $A \rightarrow B$ stands for $\Pi x{:}A.B(x)$ when $x \notin FV(B)$.
  - $P \supset Q$ stands for $\forall x{:}A.B(x)$ when $x \notin FV(Q)$.
  - In other words, $A \rightarrow B$/$P \supset Q$ are just special cases of Π-types.

Π-types/∀-propositions

$$(\Pi_T) \qquad \frac{\Gamma \vdash A\ type \qquad \Gamma,\ x : A \vdash B[x]\ type}{\Gamma \vdash \Pi x : A.B[x]\ type}$$

$$(\Pi_P) \qquad \frac{\Gamma \vdash A\ type \qquad \Gamma,\ x : A \vdash P[x]\ prop}{\Gamma \vdash \Pi x : A.P[x]\ prop}$$

$$(\lambda) \qquad \frac{\Gamma,\ x : A \vdash b : B}{\Gamma \vdash \lambda x : A.b : \Pi x : A.B[x]}$$

$$(app) \qquad \frac{\Gamma \vdash f : \Pi x : A.B[x] \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B[a]}$$

Π_T for Π-types and Π_P for universal quantification

Π-polymorphism – a first informal look

- Use of Π-types for polymorphism – an example:
  - How to model predicate-modifying adverbs (eg, quickly)?
  - Informally, it can take a verb and return a verb.
  - Montague:

    $$\text{quickly} : (e \to t) \to (e \to t)$$

    $$\text{quickly(run)} : e \to t$$

  - MTT-semantics, where A_q is the domain/type for quickly:

    $$\text{quickly} : (A_q \to Prop) \to (A_q \to Prop)$$

    What about other verbs? A_{talk}=Human, … Can we do it generically with one type of all adverbs?
  - Π-types for polymorphism come for a rescue:

    $$\text{quickly} : \Pi A{:}CN.\ (A \to Prop) \to (A \to Prop)$$

  - Question: What is CN?

    Answer: CN is a universe of types – next slide.

II.3. Universes and Π-polymorphism

- Universe of types
  - Martin-Löf introduced the notion of universe (1973, 1984)
  - A universe is a type of types (Note: the collection Type of all types is not a type itself – logical paradox if one allowed Π-quantification over Type.)
- Examples
  - Math: needing to define type-valued functions
    - f(n) = N x … x N (n times)
  - MTT-semantics: for example,
    - CN is the universe of types that are (interpretations of) CNs. We have: Human : CN, Book : CN, Σ(Man,handsome) : CN, …
    - We can then have: quickly : ΠA:CN. (A→Prop)→(A→Prop)
    - Note: one cannot have ΠA:Type..., since Type is not a type.

Modelling subsective adjectives

- Nature of such adjectives
  - Their meanings are dependent on the nouns they modify.
  - Eg, "a large mouse" is not a large animal
- This leads to our following proposal:
  - large : $\Pi A{:}CN.\ (A \rightarrow Prop)$
    - CN – type universe of all (interpretations of) CNs
    - Π is the type of dependent functions
      - large(Mouse) : Mouse → Prop
      - [large mouse] = $\Sigma x{:}Mouse.\ large(Mouse)(x)$
  - skilful : $\Pi A{:}CN_H.\ (A \rightarrow Prop)$
    - CN_H – sub-universe of CN of subtypes of Human
      - skilful(Doctor) : Doctor → Prop
      - Skilful doctor = $\Sigma x{:}Doctor.\ skilful(Doctor)(x)$
    - Excludes expressions like "skilful car".

Another example – type of quantifiers

- Generalised quantifiers
  - Examples: some, three, a/an, all, …
  - In sentences like: "Some students work hard."
- With Π-polymorphism, the type of binary quantifiers is:

  $\Pi A{:}CN.\ (A \to Prop) \to Prop$

  For Q of the above type

  $N : CN,\ \ V : N \to Prop\ \ \Rightarrow\ \ Q(N,V) : Prop$

  E.g., Student : CN, work_hard : Human→Prop

  ➔ Some(Student,work_hard) : Prop

Note: the above only works because Student ≤ Human – subtyping, a topic to be studied in the next lecture.

Modelling NL coordination

- Examples of conjoinable types
  - John walks and Mary talks. (sentences)
  - John walks and talks. (verbs)
  - A friend and colleague came. (CNs)
  - Every student and every professor came. (quantified NPs)
  - Some but not all students got an A. (quantifiers)
  - John and Mary went to Italy. (proper names)
  - I watered the plant in my bedroom but it still died slowly and agonizingly. (adverbs)
  - … …
- Question: can we consider coordination <u>generically</u>?

- Consider a universe LType
  - LType – the universe of “linguistic types”, with formal rules in the next slide.
- Example types in Ltype:
  - Type CN of common nouns
  - Type of predicate-modifying adverbs:
    $\Pi A{:}CN.\ (A \rightarrow Prop) \rightarrow (A \rightarrow Prop)$
  - Type of quantifiers:
    $\Pi A{:}CN.\ (A \rightarrow Prop) \rightarrow Prop$
  - …

$$\frac{}{PType : Type} \qquad \frac{}{Prop : PType} \qquad \frac{A : LType \quad P(x) : PType\ [x{:}A]}{\Pi x{:}A.P(x) : PType}$$

$$\frac{}{LType : Type} \qquad \frac{}{\mathrm{CN} : LType} \qquad \frac{A : \mathrm{CN}}{A : LType} \qquad \frac{A : PType}{A : LType}$$

Fig. 1. Some (not all) introduction rules for $LType$.

- Then, coordination can be considered generically:
  - Every (binary) coordinator is of the following type:

    $$\Pi A : LType.\ A \to A \to A$$

  - For example,

    $$\text{and} : \Pi A : LType.\ A \to A \to A$$

- We can then type the coordination examples we have considered.
- Remark: of course, there are further considerations such as collective readings verses distributive readings – beyond our discussions here.

Plan of Lecture III

- Brief recap of Π-types and polymorphism
  - Illustrate the use of Π and universes by GQs/coordination
- Subtyping in MTTs and applications
  - Subsumptive v.s. coercive subtyping
  - Uses of coercive subtyping in
    - Sense selection
    - Copredication
    - … …
  - Adequacy of coercive subtyping for MTTs

Let’s start with two slides seen yesterday.

II.2. Dependent product types (Π-types)

- <u>Informally</u> (borrowing set-theoretical notations, formal rules next slide),

  $\Pi x{:}A.B[x] = \{ f \mid \text{for any } a : A,\ f(a) : B[a] \}$
- Examples
  - $\lambda x{:}Nat.[1,\ldots,x] : \Pi x{:}Nat.Vect(x)$
  - $\forall x{:}Student.\ work\_hard(x)$
    - This is just another notation for $\Pi x{:}Student.\ work\_hard(x)$
  - $\forall x{:}Man.\ handsome(x) \supset \neg ugly(x)$
- Notational conventions:
  - $A \rightarrow B$ stands for $\Pi x{:}A.B(x)$ when $x \notin FV(B)$.
  - $P \supset Q$ stands for $\forall x{:}A.B(x)$ when $x \notin FV(Q)$.
  - In other words, $A \rightarrow B$/$P \supset Q$ are just special cases of Π-types.

II.3. Universes and Π-polymorphism

- Universe of types
  - Martin-Löf introduced the notion of universe (1973, 1984)
  - A universe is a type of types (Note: the collection Type of all types is not a type itself – logical paradox if one allowed Π-quantification over Type.)
- Examples
  - Math: needing to define type-valued functions
    - f(n) = N x … x N (n times)
  - MTT-semantics: for example,
    - CN is the universe of types that are (interpretations of) CNs. We have: Human : CN, Book : CN, Σ(Man,handsome) : CN, …
    - We can then have: quickly : $\Pi A{:}CN.\ (A \rightarrow Prop) \rightarrow (A \rightarrow Prop)$
    - Note: one cannot have ΠA:Type..., since Type is not a type.

Another example – type of quantifiers

- Generalised quantifiers
  - Examples: some, three, a/an, all, …
  - In sentences like: "Some students work hard."
- With Π-polymorphism, the type of binary quantifiers is:

$$\Pi A{:}CN.\ (A\rightarrow Prop)\rightarrow Prop$$

  - For Q of the above type

    N : CN, V : N→Prop

    ➔ Q(N,V) : Prop
  - E.g., for Some of the above type

    Student : CN, work_hard : Human→Prop

    ➔ Some(Student,work_hard) : Prop

Note: This only works because Student ≤ Human – subtyping, a topic to be studied later.

Modelling NL coordination

- Examples of conjoinable types
  - John walks and Mary talks. (sentences)
  - John walks and talks. (verbs)
  - A friend and colleague came. (CNs)
  - Every student and every professor came. (quantified NPs)
  - Some but not all students got an A. (quantifiers)
  - John and Mary went to Italy. (proper names)
  - I watered the plant in my bedroom but it still died slowly and agonizingly. (adverbs)
  - … …
- Question: can we consider coordination <u>generically</u>?

- Consider a universe LType
  - LType – the universe of "linguistic types", with formal rules in the next slide.
- Example types in LType:
  - Prop of logical propositions (sentence coordination)
  - Type of predicates (verb coordination)
  - CN of common nouns (CN coordination)
  - Type of predicate-modifying adverbs:
    $\Pi A{:}CN.\ (A \to Prop) \to (A \to Prop)$ (adverb coordination)
  - Type of quantifiers:
    $\Pi A{:}CN.\ (A \to Prop) \to Prop$ (quantifier coordination)
  - …

$$\frac{}{PType : Type} \qquad \frac{}{Prop : PType} \qquad \frac{A : LType \quad P(x) : PType\ [x{:}A]}{\Pi x{:}A.P(x) : PType}$$

$$\frac{}{LType : Type} \qquad \frac{}{\mathrm{CN} : LType} \qquad \frac{A : \mathrm{CN}}{A : LType} \qquad \frac{A : PType}{A : LType}$$

Fig. 1. Some (not all) introduction rules for *LType*.

`

- Then, coordination can be considered generically:
  - Every (binary) coordinator is of the following type:

    $\Pi A : LType.\ A \rightarrow A \rightarrow A$

  - For example,

    and : $\Pi A : LType.\ A \rightarrow A \rightarrow A$

- With this typing for coordinators like and, we can then type the coordination examples we have considered.
- Remark: Further considerations such as collective verses distributive readings can be dealt with similarly – beyond our discussions here.

III. Subtyping

- Basics on subtyping
  - Subsumptive v.s. coercive subtyping
  - Adequacy for MTTs
- Importance and applications of subtyping in NL sem.
  - Crucial for MTT-semantics
  - Several uses, including
    - Sense selection via overloading
    - Dot-types for copredication

(Here, we shall illustrate applications first and, if time allows, adequacy issue afterwards.)

Subsumptive subtyping: traditional notion

- Subsumptive subtyping:

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

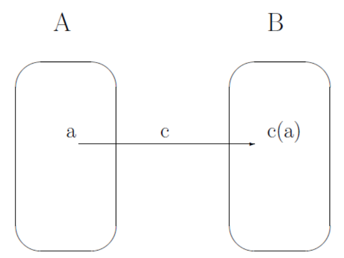
  This is called the "subsumption rule".
- Fundamental principle of subtyping

  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.

Coercive subtyping: basic idea

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



  Eg. Even ≤ Nat; Man ≤ Human; ∑(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)"

- This is more general than subsumptive subtyping and adequate for MTTs as well.

Coercive subtyping: summary

- Inadequacy of subsumptive subtyping
  - Canonical objects
  - Canonicity: key for MTTs (TTs with canonical objects)
  - Subsumptive subtyping violates canonicity.
- Adequacy of coercive subtyping for MTTs
  - Coercive subtyping preserves canonicity & other properties.
  - Conservativity (Soloviev & Luo 2002, Luo, Soloviev & Xue 2012)
- Historical development and applications in CS
  - Formal presentation (Luo 1996/1999, Luo, Soloviev & Xue 2012)
  - Implementations in proof assistants: Coq, Lego, Plastic, Matita

III.1. Modelling Advanced Linguistic Features

- MTTs
  - Very useful in modelling various linguistic features
- Why? Partly because of
  - Rich/powerful typing mechanisms
  - Subtying
  - … …

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

    $\forall x.\ farmer(x)\ \&\ [\exists y.\ donkey(y)\ \&\ own(x,y)]$
    $\Rightarrow beat(x,?y)$
  - Modern TTs (Π for $\forall$ and $\sum$ for $\exists$; Sundholme):

    $\Pi x{:}Farmer\Pi z{:}[\sum y{:}Donkey.\ own(x,y)]\ beat(x,\pi_1(z))$
- But, this is only an interesting point … We shall focus on several other things.

Uses of coercive subtyping in MTT-semantics

1. Needs for subtyping in MTT-semantics
2. Sense enumeration/selection via. overloading
3. Linguistic coercions
4. Dot-types and copredication

1. Subtyping: basic need in MTT-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 $\leq$ Human
  - [handsome man] = Σx:Man.handsome(x) $\leq_{\pi_1}$ Man
  - [Paul walks] = walk(p) : Prop

    because

    walk : Human$\rightarrow$Prop and

    p : [handsome man] $\leq_{\pi_1}$ Man $\leq$ Human

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. $[\text{run}]_1$: Human→Prop
2. $[\text{run}]_2$: Human→Institution→Prop

"run" is <u>overloaded</u> – how to disambiguate?

Overloading via coercive subtyping

- Overloading can be represented by coercions
  Eg.

$$\begin{tikzcd} & {[run]_1 : Human \to Prop} \\ run : 1_{run} \arrow[ur, "c_1"] \arrow[dr, "c_2"'] & \\ & {[run]_2 : Human \to Institution \to Prop} \end{tikzcd}$$

- Now, "John runs quickly" = "John $[\text{run}]_1$ quickly".
  "John runs a bank" = "John $[\text{run}]_2$ a bank".
- Homonymous meanings can be represented so that automated selection can be done according to typings.

3. Linguistic Coercions

- Basic linguistic coercions can be represented by means of coercions in coercive subtyping:
  - (*) Julie enjoyed a book.
  - (**) $\exists x$: Book. enjoy(j, x)
  - enjoy : Human → Event → Prop
  - Book $\leq_{reading}$ Event
  - (*) Julie enjoyed reading a book.
- Local coercions to disambiguate multiple coercions:
  - **coercion** Book $\leq_{reading}$ Event **in** (**)
  - **coercion** Book $\leq_{writing}$ Event **in** (**)

Dependent typing

- What about (example by Asher in [Asher & Luo]):

  (#) Jill just started War and Peace, which Tolstoy finished after many years of hard work. But that won't last because she never gets through long novels.

  - Overlapping scopes of "reading" and "writing".
- A solution with dependent typing
  - Evt : Human → Type
    - Evt(h) is the type of events conducted by h : Human.
  - start, finish, last : ∏h: Human. (Evt(h)→Prop)
  - read, write : ∏h: Human. Book→Evt(h)
  - Book $\leq_{c(h)}$ Evt(h), where c(h,b)=writing if "h wrote b" & c(h,b)=reading if otherwise (<u>parameterised</u> coercion over h)

- Then, (#) is formalised as
  - start(j,wp)
    & finish(t,wp)
    & ¬last(j,wp)
    & ∀lb : LBook. finish(j, π_1(lb))

  which is (equal to)

    start(j,reading(j,wp))
    & finish(t,writing(t,wp))
    & ¬last(j,reading(j,wp))
    & ∀lb : LBook. finish(j, c(j,π_1(lb)))

  as intended.

Plan of Lecture IV

- Logic in an MTT
  - Propositions-as-types, consistency, and HOL in UTT
- Brief recap of coercive subtyping
  - Explain the inadequacy of subsumptive subtyping for MTTs
- Two applications of coercive subtyping
  - Copredication via dot-types
    - Dot-types in MTTs for copredication
  - Disjoint union types (A+B)
    - Modelling privative adjective modifications (eg, fake gun)

IV.1. Logics in MTTs – propositions as types

- Curry-Howard correspondence (1958,1969):
  - Formulae as types
  - Proofs as objects

formula	type	example
$P \supset Q$	$P \rightarrow Q$	If … then …
$\forall x{:}A.P(x)$	$\prod x{:}A.P(x)$	Every man is handsome.

Eg: $\lambda x{:}P.x : P \rightarrow P$

Curry-Howard correspondence: basic example

- Theorem.
  $\vdash^{L}$ for the implicational intuitionistic logic and
  $\vdash$ for the simply typed λ-calculus.
  Then,
  - if $\Gamma \vdash M : A$, then $e(\Gamma) \vdash^{L} A$, where $e(\Gamma)$ maps $x{:}A$ to A;
  - if $\Delta \vdash^{L} A$, then $\Gamma \vdash M : A$ for some Γ & M such that $e(\Gamma) \equiv \Delta$.

Implicational propositional logic

$$(Ax) \qquad \frac{}{\Gamma,\ A \vdash A}$$

$$(\to I) \qquad \frac{\Gamma,\ A \vdash B}{\Gamma \vdash A \to B}$$

$$(\to E) \qquad \frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

where Γ is a set of formulas A.

Simply-typed λ-calculus (rules as before)

(Var) $$\frac{}{\Gamma,\ x : A \vdash x : A}$$

(Abs) $$\frac{\Gamma,\ x : A \vdash b : B}{\Gamma \vdash \lambda x : A.b : A \to B}$$

(App) $$\frac{\Gamma \vdash f : A \to B \quad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B}$$

where Γ is a set of assumptions of the form $x : A$.

Logic in impredicative type theories

- Prop – universe of logical propositions

$$\frac{\Gamma\ valid}{\Gamma \vdash Prop : Type} \qquad \frac{\Gamma \vdash A : Prop}{\Gamma \vdash A : Type}$$

Notational notes:

In these three slides, "A : Type" stands for "A type".

Π-types/universal quantification with Prop

$$(\Pi_T) \qquad \frac{\Gamma \vdash A : Type \quad \Gamma,\ x : A \vdash B : Type}{\Gamma \vdash \Pi x : A.B : Type}$$

$$(\Pi_P) \qquad \frac{\Gamma \vdash A : Type \quad \Gamma,\ x : A \vdash P : Prop}{\Gamma \vdash \Pi x : A.P : Prop}$$

$$(\lambda) \qquad \frac{\Gamma,\ x : A \vdash b : B}{\Gamma \vdash \lambda x : A.b : \Pi x : A.B}$$

$$(app) \qquad \frac{\Gamma \vdash f : \Pi x : A.B \quad \Gamma \vdash a : A}{\Gamma \vdash f(a) : [a/x]B}$$

Π_T for Π-types and Π_P for universal quantification

Logical operators in, eg, UTT

$$
\begin{array}{rcl}
\forall x{:}A.P[x] & =_{df} & \Pi x{:}A.P[x] \\
P_1 \supset P_2 & =_{df} & \forall x{:}P_1.P_2 \\
\mathbf{true} & =_{df} & \forall X{:}Prop.\ X \supset X \\
\mathbf{false} & =_{df} & \forall X{:}Prop.X \\
P_1 \ \&\ P_2 & =_{df} & \forall X{:}Prop.\ (P_1 \supset P_2 \supset X) \supset X \\
P_1 \vee P_2 & =_{df} & \forall X{:}Prop.\ (P_1 \supset X) \supset (P_2 \supset X) \supset X \\
\neg P_1 & =_{df} & P_1 \supset \mathbf{false} \\
\exists x{:}A.P[x] & =_{df} & \forall X{:}Prop.\ (\forall x{:}A.(P[x] \supset X)) \supset X.
\end{array}
$$

- Why are these definitions reasonable?
  - Usual introduction/elimination rules are all derivable.
- Examples
  - Conjunction
    - If P and Q are provable, so is P & Q.
    - If P & Q is provable, so are P and Q.
  - Falsity
    - false has no proof in the empty context (logical consistency).
    - false implies any proposition.

An episode: logic-enriched type theories

- Curry-Howard naturally leads to *intuitionistic* logics.
  - What about, say, *classical* logics?
- But:
  - Type-checking and logical inference are orthogonal.
  - They can be independent with each other.
  - In particular, the embedded logic of a type theory is not necessarily intuitionistic.
  - Type theories are not just for constructive mathematics.
- A possible answer to the above question:
  - Logic-enriched type theories (LTTs)
  - Some work: Gambino & Aczel 2006, Luo 2006, Adams & Luo 2010.

IV.2. Subtyping: recap and the adequacy issue

Let's start with three slides seen yesterday – the basic concepts in subsumptive subtyping and coercive subtyping.

Subsumptive subtyping: traditional notion

- Subsumptive subtyping:

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

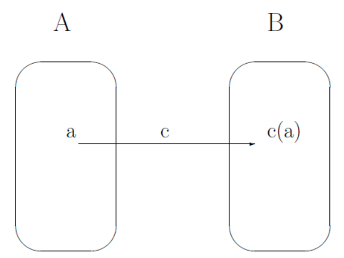
  This is called the "subsumption rule".
- Fundamental principle of subtyping

  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.

Coercive subtyping: basic idea

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



  Eg. Even ≤ Nat; Man ≤ Human; ∑(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)"

- This is more general than subsumptive subtyping and adequate for MTTs as well.

Adequacy of subtyping

Question:

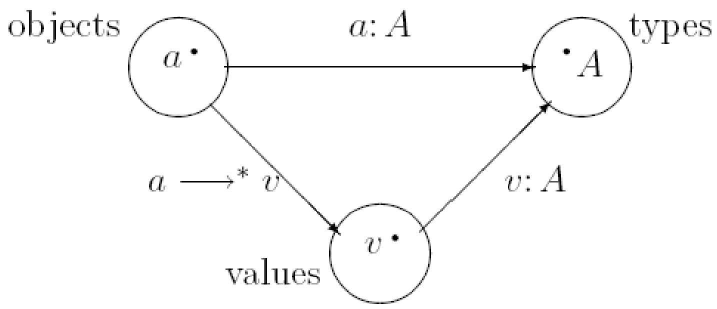
Is subsumptive subtyping adequate for MTTs (or type theories with canonical objects)?

Answer:

No (canonicity fails)!

(Hence coercive subtyping.)

Canonicity



Example:

- A = Nat, a = 3+4, v = 7.

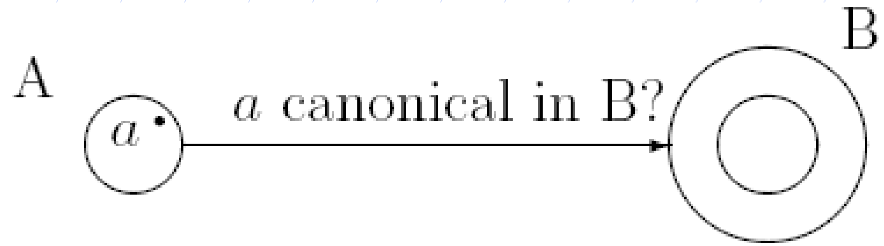
- Definition

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

- This is a basis of MTTs – type theories with canonical objects.
  - This is why the elimination rule is adequate.
  - For Σ-types, for example, its elimination rules say that any closed object in a Σ-type is a pair.

Canonicity for subsumptive subtyping?

Q: If A≤B and a:A is canonical in A, is it canonical in B?



- 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) $\neq$ 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: summary

- Inadequacy of subsumptive subtyping
  - Canonical objects
  - Canonicity: key for MTTs (TTs with canonical objects)
  - Subsumptive subtyping violates canonicity.
- Adequacy of coercive subtyping for MTTs
  - Coercive subtyping preserves canonicity & other properties.
  - Conservativity (Soloviev & Luo 2002, Luo, Soloviev & Xue 2012)
- Historical development and applications in CS
  - Formal presentation (Luo 1996/1999, Luo, Soloviev & Xue 2012)
  - Implementations in proof assistants: Coq, Lego, Plastic, Matita

IV.3. Dot-types and copredication

- Copredication (Asher, Pustejovsky, …)
  - John picked up and mastered the book.
  - The lunch was delicious but took forever.
  - The newspaper you are reading is being sued by Mia.
  - … …
- How to deal with this in formal semantics
  - Dot-objects (eg, Asher 2011, in the Montagovian setting)
  - It has a problem: subtyping and CNs-as-predicates strategy do not fit with reach other …

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

    $$\mathrm{Phy}\bullet\mathrm{Info}\to t \leq \mathrm{Phy}\to t$$

    By contravariance, we need

    $$\mathrm{Phy} \leq \mathrm{Phy}\bullet\mathrm{Info}$$

    But, this is <u>not</u> the case (the opposite is)!
- In MTT-semantics, because CNs are interpreted as types, things work as intended (see next slide).

- In MTT-semantics, CNs are types – we have:

  "John picked up and mastered the book."

  $[\![\text{pick up}]\!]: \ \text{Human} \rightarrow \text{PHY} \rightarrow \text{Prop}$

  $\quad \leq \text{Human} \rightarrow \text{PHY}\bullet\text{INFO} \rightarrow \text{Prop}$

  $\quad \leq \text{Human} \rightarrow [\![\text{book}]\!] \rightarrow \text{Prop}$

  $[\![\text{master}]\!]: \ \text{Human} \rightarrow \text{INFO} \rightarrow \text{Prop}$

  $\quad \leq \text{Human} \rightarrow \text{PHY}\bullet\text{INFO} \rightarrow \text{Prop}$

  $\quad \leq \text{Human} \rightarrow [\![\text{book}]\!] \rightarrow \text{Prop}$

  Hence, both have the same type (in LType) and therefore can be coordinated by "and" to form "picked up and mastered" in the above sentence.

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

Question: How to introduce dot-types like PHY•INFO in an MTT?

Dot-types in MTTs

- What is A•B?
  - Inadequate accounts (as summarised in (Asher 08)):
    - Intersection type
    - Product type
- Proposal (SALT20, 2010)
  - A•B as type of pairs that do not share components
  - Both projections as coercions
- Implementations
  - Coq implementations (Luo/LACL11,
  - Implemented in proof assistant Plastic by Xue (2012).

Key points of a dot-type

- A dot-type is not an ordinary type (eg, not an inductive type).
- To form A•B, A and B cannot share components:
  - E.g., "Phy•Phy" and "(Phy•Info)•Phy" are not dot-types.
  - This is in line with Pustejovsky's view that dot-objects "*appear in selectional contexts that are contradictory in type specification*." (2005)
- A•B is like AxB but both projections are coercions:
  - $A \bullet B \leq_{\pi_1} A$ and $A \bullet B \leq_{\pi_2} B$
  - This is OK because of the non-sharing requirement. (Note: to have both projections as coercions would not be OK for product types AxB since coherence would fail.)

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

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

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

Another example

- "heavy book"
  - [heavy] : Phy $\rightarrow$ Prop

    $\leq$ Phy•Info $\rightarrow$ Prop

    $\leq$ Book $\rightarrow$ Prop
  - So, the following is well-formed:

    [heavy book] = Σ(Book, [heavy])

IV.4. Disjoint union types

- Disjoint union types
  - A+B with two injections inl : A→A+B and inr : B→A+B
  - Rules for A+B – formation/introduction/elimination/computation rule(s)

Recall the following slide on adjectives:

- adjective : CNs → CNs
  - In MG, predicates to predicates.
  - In MTT-semantics, types to types.
- Proposals in MTT-sem (Chatzikyriakidis & Luo, FG13 & JoLLI17)

classification	**example**	**types employed**
Intersective	handsome man	Σ-types (of pairs)
Subsective	large mouse	Π-types (polymorphism)
Privative	fake gun	disjoint union types
Non-committal	alleged criminal	belief contexts

Privative adjectives

- "fake gun"
  - G_R – type of real guns
  - G_F – type of fake guns
  - $G = G_R + G_F$ – type of all guns
  - Declare inl and inr both as coercions: $G_R \leq_{inl} G$ and $G_F \leq_{inr} G$
- Now, eg,
  - Can define "real gun" or "fake gun" inductively as predicates of type $G \rightarrow Prop$ so that $\neg$[real gun](g) iff [fake gun](g).
  - We can interpret, for $f : G_F$, "f is not a real gun" as $\neg$[real gun](f), which is logically equivalent to [fake gun](f), which is True.
  - Note that, in the above, [real gun](f) and [fake gun](f) are only well-typed because $G_R \leq_{inr} G$ and $G_F \leq_{inr} G$.

V. Advanced Topics

- Advanced topics in MTT-semantics
  - Dependent types in event semantics
  - MTT-semantics is both model-theoretic & proof-theoretic
  - Dependent Categorial Grammars
    - Syntactic analysis corresponding to MTT-semantics
    - Two papers: Lambek dependent types (Luo 2015) and Linear dependent types (Luo and Zhang 2016)
  - … …

We shall consider the first two in this lecture.

(BTW, references for all lectures are available – see the last several slides of this lecture.)

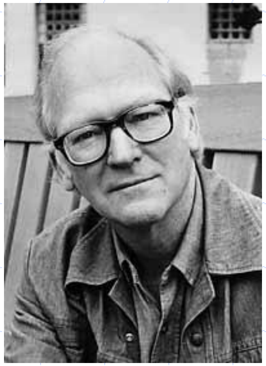
V.1. Dependent Event Types

- This part is based on the slides for my last week's presentation of the following paper:
  - Z. Luo and S. Soloviev. Dependent Event Types. London, WoLLIC 2017.

I. Dependent event types
  - C_e: DETs in simple type theory (Montague's setting)
  - UTT[E]: DETs in modern type theories (MTT-semantics)
  - Adequacy of C_e: embedding into UTT[E]
  - Comparison of traditional event semantics, C_e and UTT[E]

II. Event quantification problem: an example
  - EQP in traditional event sem. and solutions in C_e and UTT[E]

Davidson's event semantics



- Consider:
  - (*) John buttered the toast.
    [(*)] = butter(j,t), where butter : $\mathbf{e}^2 \to \mathbf{t}$.
  - (**) John buttered the toast with the knife at midnight.
    (?) [(**)] = butter(j,t,k,m), where butter : $\mathbf{e}^4 \to \mathbf{t}$
    (?) [(**)] = m(k(butter(j)))(t), where butter : $\mathbf{e} \to \mathbf{e} \to \mathbf{t}$, m/k : $(\mathbf{e} \to \mathbf{t}) \to (\mathbf{e} \to \mathbf{t})$
- Davidson's original motivation (1967): better treatment of adverbial modifications – e.g., butter : $\mathbf{e}^2 \to \mathrm{Event} \to \mathbf{t}$, and
  - [(*)] = $\exists$e:Event. butter(j,t,e)
  - [(**)] = $\exists$e:Event. butter(j,t,e) & with(e,k) & at(e,m)
  - Note: [(**)]$\supset$[(*)], among many other desirable inferences.
    (No need for meaning postulates, needed in both (?)-approaches.)
- Neo-Davidson semantics (1980s): eg, butter : $\mathrm{Event} \to \mathbf{t}$ and
  - [(*)] = $\exists$e:Event. butter(e) & agent(e)=j & patient(e)=t.

I. Dependent event types

- Refined types of events: Event ➔ Evt(…)
- Event types dependent on agents/patients
  - For a:Agent and p:Patient, consider dependent event types

    Event, Evt_A(a), Evt_P(p), Evt_{AP}(a,p)
  - Note: the subscripts A, P and AP are just symbols.
- Subtyping (a:A and A≤B ➔ a:B) between DETs:

$$
\begin{tikzcd}
 & Evt_A(a) \arrow[dr, dash, "\leq"] & \\
Evt_{AP}(a,p) \arrow[ur, dash, "\leq"] \arrow[dr, dash, "\leq"'] & & Event \\
 & Evt_P(p) \arrow[ur, dash, "\leq"'] &
\end{tikzcd}
$$

Dependent event types in Montagovian setting

- Eg. John talked loudly.
  - talk, loud : Event→**t**
  - agent : Event→**e**→**t**
- (neo-)Davidsonian event semantics

$$\exists e : Event.\ talk(e)\ \&\ loud(e)\ \&\ agent(e, j)$$

- Dependent event types in Montagovian setting:

$$\exists e : Evt_A(j).\ talk(e)\ \&\ loud(e)$$

which is well-typed because $Evt_A(j) \le Event$.

C_e: Underlying formal system

- C_e extends Church's simple type theory (1940) (as used by Montague in MG), by dependent event types
- Church's STT

$$\frac{}{\mathbf{e}\ \mathit{type}} \qquad \frac{}{\mathbf{t}\ \mathit{type}} \qquad \frac{}{x : A\ [x : A]} \qquad \frac{}{P\ \mathit{true}\ [P\ \mathit{true}]}$$

$$\frac{A\ \mathit{type} \quad B\ \mathit{type}}{A \to B\ \mathit{type}} \qquad \frac{b : B\ [x : A] \quad x \notin FV(B)}{\lambda x{:}A.b : A \to B} \qquad \frac{f : A \to B \quad a : A}{f(a) : B}$$

$$\frac{P : \mathbf{t} \quad Q : \mathbf{t}}{P \supset Q : \mathbf{t}} \qquad \frac{Q\ \mathit{true}\ [P\ \mathit{true}]}{P \supset Q\ \mathit{true}} \qquad \frac{P \supset Q\ \mathit{true} \quad P\ \mathit{true}}{Q\ \mathit{true}}$$

$$\frac{A\ \mathit{type} \quad P : \mathbf{t}\ [x : A]}{\forall(A, x.P) : \mathbf{t}} \qquad \frac{P\ \mathit{true}\ [x : A]}{\forall(A, x.P)\ \mathit{true}} \qquad \frac{\forall(A, x.P[x])\ \mathit{true} \quad a : A}{P[a]\ \mathit{true}}$$

Dependent event types in C_e

$$\frac{}{Agent\ type} \qquad \frac{}{Patient\ type}$$

$$\frac{}{Event\ type} \qquad \frac{a : Agent}{Evt_A(a)\ type} \qquad \frac{p : Patient}{Evt_P(p)\ type} \qquad \frac{a : Agent \quad p : Patient}{Evt_{AP}(a,p)\ type}$$

$$\frac{a : Agent \quad p : Patient}{Evt_{AP}(a,p) \leq Evt_A(a)} \qquad \frac{a : Agent \quad p : Patient}{Evt_{AP}(a,p) \leq Evt_P(p)} \qquad \frac{a : Agent}{Evt_A(a) \leq Event} \qquad \frac{p : Patient}{Evt_P(p) \leq Event}$$

$$\frac{A\ type}{A \leq A} \qquad \frac{A \leq B \quad B \leq C}{A \leq C} \qquad \frac{A' \leq A \quad B \leq B'}{A \to B \leq A' \to B'}$$

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

$$\begin{tikzcd}
 & Evt_A(a) \arrow[dr, dash, "\leq"] & \\
Evt_{AP}(a,p) \arrow[ur, dash, "\leq"] \arrow[dr, dash, "\leq"'] & & Event \\
 & Evt_P(p) \arrow[ur, dash, "\leq"'] &
\end{tikzcd}$$

UTT[E]: Dependent event types in MTT-sem

- UTT[E]: UTT with coercions in E
  - UTT: a modern type theory (Luo 1994)
  - E characterising subtyping for DETs
- Dependent event types in MTT-semantics

John talked loudly.

$talk : \Pi h : Human.\ Evt_A(h) \to Prop.$

$loud : Event \to Prop.$

$[\![\text{John talked loudly}]\!] = \exists e : Evt_A(j).\ talk(j, e)\ \&\ loud(e).$

UTT[E]: formal presentation in LF

- Constant types/families:
  - $Entity : Type$
  - $Agent,\ Patient : Type.$
  - $Event : Type,$

    $Evt_A : (Agent)Type,$

    $Evt_P : (Patient)Type,$ and

    $Evt_{AP} : (Agent)(Patient)Type.$
- Coercive subtyping in E for DETs:

$$Evt_{AP}(a,p) \leq_{c_1[a,p]} Evt_A(a), \quad Evt_{AP}(a,p) \leq_{c_2[a,p]} Evt_P(p),$$
$$Evt_A(a) \leq_{c_3[a]} Event, \quad Evt_P(p) \leq_{c_4[p]} Event,$$

  where $c_3[a] \circ c_1[a,p] = c_4[p] \circ c_2[a,p].$
- UTT[E] has nice properties such as normalisation and consistency (Luo, Soloviev & Xue 2012).

Faithful embedding of C_e into UTT[E]

- Definition (embedding of C_e into UTT[E])
  - $[x] = x$; $[e] = Entity$; $[t] = Prop$
  - $[A \rightarrow B] = [A] \rightarrow [B]$;

    $[\lambda x{:}A.b] = \lambda([A],T,[x{:}[A]].[b])$, if $[b] : T$;

    $[f(a)] = app(S,T,[f],[a])$, if $[f] : S \rightarrow T$ and $[a] : S_0 \leq S$.
  - $[P \supset Q] = [P] \supset [Q]$; $[\forall(A,x.P)] = \forall([A], [x{:}[A]].[P])$
- Theorem (embedding is "faithful")
  - $\Gamma \vdash A$ type $\Rightarrow [\Gamma] \vdash [A] : Type$.
  - $\Gamma \vdash a : A \Rightarrow [\Gamma] \vdash [a] : A_0$ for some A_0 s.t. $[\Gamma] \vdash A_0 \leq_d [A]$ for some d.
  - $\Gamma \vdash P$ true $\Rightarrow [\Gamma] \vdash p : [P]$, for some p.
  - $\Gamma \vdash A \leq B \Rightarrow [\Gamma] \vdash [A] \leq_c [B] : Type$, for some unique c.
- Corollary: C_e inherits nice properties from UTT[E] including, e.g., normalisation and logical consistency.

Comparison (John talked loudly)

- (neo-)Davidsonian event semantics
  - talk, loud : Event→**t** and agent : Event→**e**→**t**.

    $$\exists e : Event.\ talk(e)\ \&\ loud(e)\ \&\ agent(e, j)$$

- Dependent event types in Montagovian setting:
  - talk, loud : Event→**t** and agent : Event→**e**→**t**.

    $$\exists e : Evt_A(j).\ talk(e)\ \&\ loud(e)$$

    which is well-typed because $Evt_A(j) \leq Event$.

- Dependent event types in MTT-semantics:

  $$talk : \Pi h : Human.\ Evt_A(h) \rightarrow Prop.$$

  $$loud : Event \rightarrow Prop.$$

  $$[\![\text{John talked loudly}]\!] = \exists e : Evt_A(j).\ talk(j, e)\ \&\ loud(e).$$

  Note: talk's type requires that e have a dependent event type.

II. Event quantification problem

- A form of incompatibility between event semantics and MG (Champollion, Winter-Zwarts, de Groote-Winter).
- No man talked.

(neo-)Davidson (even the incorrect (#) is legal)

$$(1)\quad \neg\exists x : \mathbf{e}.\ \ man(x)\ \&\ \exists e{:}Event.\ talk(e)\ \&\ agent(e,x)$$

$$(2)\quad (\#)\ \exists e : Event.\ \neg\exists x : \mathbf{e}.\ man(x)\ \&\ talk(e)\ \&\ agent(e,x)$$

DETs in Montague (the incorrect (*) is illegal)

$$(3)\quad \neg\exists x : \mathbf{e}.\ \ man(x)\ \&\ \exists e : Evt_A(x).\ talk(e)$$

$$(4)\quad (*)\ \exists e : Evt_A(x).\ \neg\exists x : \mathbf{e}.\ man(x)\ \&\ talk(e)$$

But, we still have a problem, albeit a small one …

- What if one changes $Evt_A(x)$ into Event?
- That still would not prevent the following incorrect semantics:

$$(\#)\ \exists e : Event.\ \neg\exists x : \mathbf{e}.\ man(x)\ \&\ talk(e)$$

- MTT-semantics helps:

DETs in MTT-sem

$$(5)\quad \neg\exists x : Man\ \exists e : Evt_A(x).\ talk(x,e)$$

$$(6)\quad (^*)\ \exists e : Evt_A(x).\ \neg\exists x : Man.\ talk(x,e)$$

- Note: talk's type "dictates" the use of $Evt_A(x)$: talk(x,e) would not be well-typed if e : Event <u>only</u> (and not of type $Evt_A(x)$). So, something like (#) would not be available.

Future work related to DETs: questions

- Why thematic roles as indexes of DEPs?
  - Conceptual precedency/dependency of existence?
    - $Evt_A(a)$ for a:Agent
    - "a exists" in order for an event in $Evt_A(a)$ to exist …
- Several questions on DETs
  - Dependency on other kinds of parameters than thematic roles? (eg, Evt(h) where h:Human in (Asher & Luo 12))
  - Potential applications of DETs (not just event quantification problem.)
  - Other forms of dependent event types

V.2. MTT-sem is both model-/proof-theoretic

- The above claim was first made in the following talk/paper:
  - Z. Luo. Formal Semantics in Modern Type Theories: Is It Model-theoretic, Proof-theoretic, or Both? Invited talk at LACL 2014.
- Since then, further discussions and developments have been made, although the basic theme and arguments have remained the same.

Let's start by revisiting two slides in Lecture 1.

Formal semantics

- Model-theoretic semantics
  - Meaning is given by denotation.
  - c.f., Tarski, …, Montague.
  - e.g., Montague grammar (MG)
    - NL → simple type theory → set theory
- Proof-theoretic semantics
  - In logics, meaning is inferential use (proof/consequence).
  - c.f., Gentzen, Prawitz, …, Martin-Löf.
  - e.g., Martin-Löf's meaning theory




Simple example for MTS and PTS

- Model-theoretic semantics
  - John is happy. ➔ happy(john)

    ➔ John is a member of the set of entities that are happy.
  - Montague's semantics is model-theoretic – it has a wide coverage (powerful).
- Proof-theoretic semantics
  - How to understand a proposition like happy(john)?
  - In logic, its meaning can be characterised by its uses – two respects:
    - How it can be arrived at (proved)?
    - How it can be used to lead to other consequences?

(*)

- Example argument for <u>traditional</u> set-theoretic sem.
  - Or, an argument against non-set-theoretic semantics
  - "Meanings are out in the world"
    - Portner's 2005 book on "What is Meaning" – typical view
    - Assumption that set theory represents (or even <u>is</u>) the world
- Comments:
  - This is illusion! Set theory is just a theory in FOL, not "the world".
  - A good/reasonable formal system can be as good as set theory. (For example, if set theory is good enough, then so is an MTT.)

- Claim:

  Formal semantics in Modern Type Theories

  is both model-theoretic and proof-theoretic.

  - NL → MTT (representational, model-theoretic)
    - MTT as meaning-carrying language with its types representing collections (or "sets") and signatures 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 (c.f., Martin-Löf's meaning theory)

- Traditional model-theoretic semantics:

  Logics/NL → Set-theoretic representations
- Traditional proof-theoretic semantics of logics:

  Logics → Inferences
- Formal semantics in Modern Type Theories:

  NL → MTT-representations → Inferences

Remark: This was not possible without a language like MTTs; in other words, MTTs offer a new possibility for NL semantics!

Justifications of the claim

- Model-theoretic characteristics of MTT-semantics
  - Signatures – context-like but more powerful mechanism to represent situations ("incomplete worlds")
- Proof-theoretic characteristics of MTT-semantics
  - Meaning theory of MTTs – inferential role semantics of MTT-judgements

Remark: The proof-theoretic characteristics is easier to justify; what about the model-theoretic ones? A focus of some recent work such as those on signatures.

Model-theoretic characteristics of MTT-sem

- In MTT-semantics, MTT is a representational language.
  - Types represent collections (c.f., sets in set theory) – see earlier slides on using rich types in MTTs to give semantics.
  - Signatures represent situations (or incomplete possible worlds).

Signatures

- Types and signatures/contexts are embodied in judgements:

  $$\Gamma \vdash_{\Sigma} a : A$$

  where A is a type, Γ is a context and Σ is a signature.
- New: Signatures, similar to contexts, are finite sequences of entries, but
  - their entries are introducing constants (not variables; i.e., cannot be abstracted – c.f, Edinburgh LF (Harper, Honsell & Plotkin 1993)), and
  - besides membership entries, allows more advanced ones such as manifest entries and subtyping entries (see later).

Situations represented as signatures

- Beatles' rehearsal: simple example
  - Domain: $\Sigma_1 \equiv D : Type,$
    $John : D,\ Paul : D,\ George : D,\ Ringo : D,\ Brian : D,\ Bob : D$
  - Assignment: $\Sigma_2 \equiv B : D \to Prop,\ b_J : B(John),\ ...,\ b_B : \neg B(Brian),\ b'_B : \neg B(Bob),$
    $G : D \to Prop,\ g_J : G(John),\ ...,\ g_G : \neg G(Ringo),\ ...$
  - Signature representing the situation of Beatles' rehearsal:
    $\Sigma \equiv \Sigma_1,\ \Sigma_2,\ ...,\ \Sigma_n$
  - We have, for example,
    $\Gamma \vdash_\Sigma G(John)\ \ true\ \ \ and\ \ \Gamma \vdash_\Sigma \neg B(Bob)\ \ true.$

    "John played guitar" and "Bob was not a Beatle".

Remark: the same as a slide in Lecture 2, except that we now use signatures, rather than contexts.

- This shows that, by means of membership entries, we already can do things we would usually do in models (in set theory):
  - Declaring types (say, D is a type, representing a collection)
  - Declaring objects of a type (say John : D)
  - Remark: In a many-sorted FOL, one may declare a FOL-language with sorts and constants, not different sorts/constants in the same language.
- However, we need to further increase the representational power – manifest fields and subtyping assumptions in signatures.

Manifest entries

- More sophisticated situations
  - E.g., infinite domains
- In signatures, we can have a <u>manifest entry</u>:

$$x \sim a : A$$

where $a : A$.

- Informally, it assumes x that behaves the same as a.

Manifest entries: formal treatment

- Manifest entries are just abbreviations of special membership entries:
  - x ~ a : A abbreviates x : $1_A(a)$ where $1_A(a)$ is the unit type with only object $*_A(a)$.
  - with the following coercion:

$$\frac{\Gamma \vdash_\Sigma A : Type \quad \Gamma \vdash_\Sigma a : A}{\Gamma \vdash_\Sigma \mathbf{1}_A(a) \leq_{\xi_{A,a}} A : Type}$$

  where $\xi_{A,a}(z) = a$ for every z : $1_A(a)$.
- So, in any hole that requires an object of type A, we can use x which, under the above coercion, will be coerced into a, as intended.

Manifest entries: examples

$$\Sigma_1 \equiv D : Type,$$
$$John : D,\ Paul : D,\ George : D,\ Ringo : D,\ Brian : D,\ Bob : D$$
$$\Sigma_2 \equiv B : D \to Prop,\ b_J : B(John),\ ...,\ b_B : \neg B(Brian),\ b'_B : \neg B(Bob),$$
$$G : D \to Prop,\ g_J : G(John),\ ...,\ g_G : \neg G(Ringo),\ ...$$

→→→

$$D \sim a_D : Type,\ B \sim a_B : D \to Prop,\ G \sim a_G : D \to Prop,$$

where

$$a_D = \{John,\ Paul,\ George,\ Ringo,\ Brian,\ Bob\}$$
$$a_B : D \to Prop, \text{ the predicate `was a Beatle'},$$
$$a_G : D \to Prop, \text{ the predicate `played guitar'},$$

with a_D being a finite type and a_B and a_G inductively defined. (Note: Formally, "Type" should be a type universe.)

- Infinity:
  - Infinite domain D represented by infinite type Inf

    $$D \sim Inf : Type$$

  - Infinite predicate with domain D:

    $$f \sim \text{f-defn} : D \rightarrow Prop$$

    with f-defn being inductively defined.
- "Animals in a snake exhibition":

  $$Animal_1 \sim Snake : CN$$

Subtyping entries in signatures

- Subtyping entries in a signature:

  $$c : A \leq B$$

  This is to declare $A \leq_c B$, where c is a functional operation from A to B.
- Eg, we may have

  $$D \sim \{\ \text{John}, \ldots\ \} : \text{Type},\ c : D \leq \text{Human}$$
- Note that, formally, for signatures,
  - we only need "coercion contexts" but do not need "local coercions" [Luo 2009, Luo & Part 2013];
  - this is meta-theoretically simpler (Lungu 2017)

Concluding Remarks

- Using contexts to represent situations: historical notes
  - Ranta 1994 (even earlier?)
  - Further references [Bodini 2000, Cooper 2009, Dapoigny/Barlatier 2010]
- We introduce "signatures" with new forms of entries: manifest/subtyping entries
  - Manifest/subtyping entries in signatures are <u>simpler</u> than manifest fields (Luo 2009) and local coercions (Luo & Part 2013).
- Preserving TT's meta-theoretic properties is important (eg, consistency of the embedded logic).
- Summary
  - NL ➔ MTT (model-theoretic)
  - MTT ➔ meaning theory (proof-theoretic)

References (1)

- N. Asher. A type driven theory of predication with complex types. Fundamenta Informaticae 84(2). 2008.
- N. Asher. *Lexical Meaning in Context: A Web of Words.* Cambridge University Press. 2011.
- N. Asher and Z. Luo. Formalisation of coercions in lexical semantics. Sinn und Bedeutung 17, Paris. 2012.
- J. Belo. Dependently Sorted Logic. LNCS 4941.
- P. Bodini. Formalizing Contexts in Intuitionistic Type Theory. Fundamenta Informaticae 4(2).
- Cartmell. Generalised algebraic theories and contextual categories, Ph.D. thesis, Oxford. 1978.
- S. Chatzikyriakidis. Adverbs in a Modern Type Theory. LACL 2014, LNCS 8535. 2014.
- S. Chatzikyriakidis and Z. Luo. Adjectives in a Modern Type-Theoretical Setting. The 18th Conf. on Formal Grammar, Dusseldorf. LNCS 8036. 2013.
- S. Chatzikyriakidis and Z. Luo. An Account of Natural Language Coordination in Type Theory with Coercive Subtyping. Constraint Solving and Language Processing 2012, LNCS 8114. 2013.

References (2)

- S. Chatzikyriakidis and Z. Luo. Natural Language Reasoning Using Proof-assistant Technology: Rich Typing and Beyond. EACL Workshop on Type Theory and Natural Language Semantics (TTNLS), Goteborg, 2014.
- S. Chatzikyriakidis and Z. Luo. Natural Language Inference in Coq. Journal of Logic, Language and Information, 23(4). 2014.
- S. Chatzikyriakidis and Z. Luo. Using Signatures in Type Theory to Represent Situations. T. Murata, K. Mineshima and D. Bekki (eds). New Frontiers in Artificial Intelligence - JSAI-isAI 2014 Workshops in Japan (LENLS, JURISIN and GABA), Revised Selected Papers. LNCS 9067, 2015.
- S. Chatzikyriakidis and Z. Luo. Individuation Criteria, Dot-types and Copredication: A View from Modern Type Theories. Proc of the 14th Inter. Conf. on Mathematics of Language, Chicago, 2015.
- S. Chatzikyriakidis and Z. Luo. Proof Assistants for Natural Language Semantics. Logical Aspects of Computational Linguistics 2016 (LACL 2016), Nancy. 2016.
- S. Chatzikyriakidis and Z. Luo (eds.). Modern Perspectives in Type Theoretical Semantics. Studies in Linguistics and Philosophy, Springer. 2017.

References (3)

- S. Chatzikyriakidis and Z. Luo. On the Interpretation of Common Nouns: Types v.s. Predicates. In S. Chatzikyriakidis and Z. Luo (eds.), Modern Perspectives in Type Theoretical Semantics. Springer. 2017.
- S. Chatzikyriakidis and Z. Luo. Adjectival and Adverbial Modification: The View from Modern Type Theories. Journal of Logic, Language and Information 26(1), 2017.
- S. Chatzikyriakidis and Z. Luo. Formal Semantics in Modern Type Theories. ISTE/Wiley Science Publishing Ltd. (to appear)
- A. Church. A formulation of the simple theory of types. J. Symbolic Logic, 5(1). 1940.
- H. Curry and R. Feys. Combinatory Logic, Vol 1. North Holland, 1958.
- Dapoigny and Barlatier. Modelling Contexts with Dependent Types. Fundamenta Informaticae 104. 2010.
- M. Dummett. The Logical Basis of Metaphysics Harvard University Press, 1991.
- M. Dummett. The Seas of Language. OUP, 1993.
- P. Elbourne. Meaning: A Slim Guide to Sremantics. OUP. 2011.
- W. Howard. The formulae-as-types notion of construction. In To HB Curry: Essays on Combinatory Logic (1980). 1969.

References (4)

- R. Kahle and P. Schroeder-Heister (eds.). Proof-Theoretic Semantics. Synthese, 2005.
- G. Lungu. Subtyping in Signatures. PhD thesis, Royal Holloway, Univ. of London. 2017. (forthcoming)
- G. Lungu and Z. Luo. Monotonicity Reasoning in Formal Semantics Based on Modern Type Theories. LACL 2014, LNCS 8535. 2014.
- Z. Luo. Coercive subtyping in type theory. CSL'96, LNCS 1258. 1996.
- Z. Luo. Coercive subtyping. J. of Logic and Computation, 9(1). 1999.
- Z. Luo. *Computation and Reasoning: A Type Theory for Computer Science.* OUP, 1994.
- Z. Luo. Type-theoretical semantics with coercive subtyping. SALT20. 2010.
- Z. Luo. Contextual analysis of word meanings in type-theoretical semantics. Logical Aspects of Computational Linguistics (LACL'2011). LNAI 6736, 2011.
- Z. Luo. Common nouns as types. LACL'12, LNCS 7351. 2012.
- Z. Luo. Formal Semantics in Modern Type Theories with Coercive Subtyping. Linguistics and Philosophy, 35(6). 2012.

References (5)

- Z. Luo. Formal Semantics in Modern Type Theories: Is It Model-theoretic, Proof-theoretic, or Both? Invited talk at Logical Aspects of Computational Linguistics 2014 (LACL 2014), Toulouse. LNCS 8535. 2014.
- Z. Luo. A Lambek Calculus with Dependent Types. TYPES 2015. Tallinn, May 2015.
- Z. Luo and P. Callaghan. Coercive subtyping and lexical semantics (extended abstract). Logical Aspects of Computational Linguistics (LACL'98). 1998.
- Z. Luo and S. Soloviev. Dependent event types. Proc of the 24th Workshop on Logic, Language, Information and Computation (WoLLIC'17), LNCS 10388. London, 2017.
- Z. Luo, S. Soloviev and T. Xue. Coercive subtyping: theory and implementation. Information and Computation 223. 2012.
- Z. Luo and Y. Zhang. A Linear Dependent Type Theory. TYPES 2016. Novi Sad, May 2016.
- P. Martin-Löf. On the Meanings of the Logical Constants and the Justifications of the Logical Laws. Nordic Journal of Philosophical Logic, 1(1). 1996.
- P. Martin-Löf. Intuitionistic Type Theory. 1984.

References (6)

- R. Montague. Formal philosophy. Yale Univ Press, 1974. (Collection edited by R. Thomason)
- B. Partee. Compositionality and coercion in semantics: the semantics of adjective meaning. In *Cognitive Foundations of Interpretation*, Netherlands Academy of Arts and Sciences. 2007.
- P. Portner. What is Meaning? Blackwell. 2005
- P. Pontner and B. Partee (eds). Formal Semantics: The Essential Readings. Blackwell. 2002.
- J. Pustejovsky. The Generative Lexicon. MIT. 1995.
- C. Retoré et al. Towards a Type-Theoretical Account of Lexical Semantics. JoLLI 19(2). 2010.
- T. Streicher. Investigations into Intensional Type Theory. Habilitation Thesis, 1993.
- Sundholm. Constructive Generalized Quantifiers. Synthese 79(1). 1989.
- J. Pustejovsky. *The Generative Lexicon*. MIT. 1995.
- A. Ranta. *Type-Theoretical Grammar*. Oxford University Press. 1994.
- T. Xue and Z. Luo. Dot-types and their implementation. LACL'12, LNCS 7351. 2012.