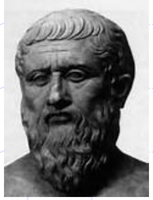
Formal Semantics in Modern Type Theories (MTT-semantics): advanced topics

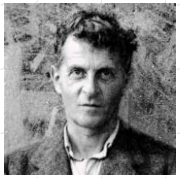
Zhaohui Luo
Royal Holloway, Univ. of London

Lecture I. Introduction

Natural Language Semantics

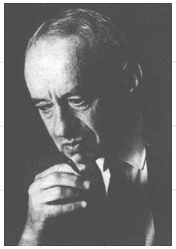
- Semantics – study of meaning
  - communicate = convey meaning
- Various kinds of theories of meaning
  - Meaning is reference ("referential theory")
    - Word meanings are things (abstract/concrete) in the world.
    - c.f., Plato, …
  - Meaning is concept ("internalist theory")
    - Word meanings 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, Church, …, Montague
  - e.g., Montague grammar (MG)
    - NL → simple type theory → set theory



- Proof-theoretic semantics
  - In logics, meaning is inferential use (two aspects: proof + consequence)
  - c.f., Gentzen , …, Prawitz
  - e.g., meaning theories (c.f., previous page)

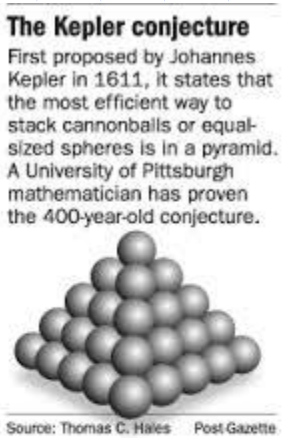
Simple v.s. Modern Type Theories

- Church's simple type theory (1940)
  - As in Montague semantics
  - Types ("single entity-sort": **e**, **t**, **e**$\rightarrow$**t**); HOL/predicates
- Modern type theories
  - Many types of entities – "many-sorted"
    - Human, Table, Σx:Man.handsome(x), Evt_T(t), Phy•Info, …
    - Dependent types, inductive types, universes, …
  - Examples of MTTs:
    - Predicative [non-standard FOL]: MLTT (Martin-Löf 1984)
    - Impredicative [HOL]: pCIC (Coq manual) and UTT (Luo 1994)

An episode: MTT-based technology and applications

- Proof technology based on type theories
  - Proof assistants
    - MTT-based: ALF/Agda, Coq, Lean, Lego, NuPRL, Plastic, …
    - HOL-based: Isabelle, 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
    - NL reasoning based on MTT-semantics

      (In Coq: Chatzikyriakidis & Luo 2014/2016/2020; Luo 2023)



Type theories as foundational languages

- Type theories as foundational languages for
  - Maths: classical/Church's STT and constructive/Martin-Löf's
  - NL semantics: Montague semantics and MTT-semantics
- A comment – what typing is not:
  - "a : A" is not a logical formula (A is not a predicate).
    - j : **e**; ugly(j) : **t**; 7 : Nat; j : Human; …
    - 7:Nat/j:Human are different from formulae nat(7)/human(j), where nat/human are predicates.
  - "a : A" is different from "a∈S" (the latter is a logical formula).
- What typing is related to (some example notions):
  - Meaningfulness (ill-typed ➔ meaningless)
  - Semantic/category errors (eg, "A table talks." – later)
  - Type presuppositions (Asher 2011)

MTT-semantics

- MTT-semantics
  - Formal semantics in modern type theories (MTTs), not simple TT
  - Has both model-/proof-theoretic characteristics. (Luo 2014)
- Development of MTT-semantics
  - Early use of dependent type theory in formal semantics:
    - Mönnich 1985, Sundholm 1986, Ranta 1994
  - Development since 2009 – full-scale alternative to Montague semantics
    - Z. Luo. Modern Type Theories: Their Development & Applications. Tsinghua Univ Press. 2023. (Monograph on MTTs with chapters on MTT-semantics, in Chinese)
    - S. Chatzikyriakidis & Z. Luo. Formal Semantics in Modern Type Theories. Wiley/ISTE. 2020. (Monograph on MTT-semantics)
    - S. Chatzikyriakidis & Z. Luo (eds.) Modern Perspectives in Type Theoretical Semantics. Springer, 2017. (Collection of papers on rich typing in NL semantics)

Course Plan

- Motivations of the course
- This lecture (Lecture I):
  - MTT-semantics: intro & case study (adjectival modification)
  - Introductory overview of the topics in Lectures II - IV
- Several traditionally "advanced" topics
  - Lect II: Events (Davidsonian ➔ dependent event types)
  - Lect III: Anaphora (Russellian/dynamic ➔ type-theoretic solution)
  - Lect IV: Copredication (dot-types ➔ formalisation in MTTs)
  - Lect V: More + analysis (e.g., dependent CGs, …; "open")

Each: history/Montague/MTT-semantics (informal & understandable)

Material and References

- Material available on the ESSLLI23 course web link:
  - Lecture slides for the first lecture (Lecture I)
  - Course proposal (good summary, but the organisation and descriptions of lectures are slightly different.)
- The slides for all 5 lectures, and the course proposal, will be available at

  https://www.cs.rhul.ac.uk/home/zhaohui/lecture-notes.html

  with references (cited in lectures) listed in the end of the slides.
- Papers/books on MTT-semantics available at

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

I.1. Introduction to MTT-semantics

Dependent types

- MTTs are dependent type theories – what's a dependent type?
- What is not a dependent type:
  - A dependent type is not a type dependent on types.
    - E.g., List(A) depends on types A and is not a dependent type.
- A dependent type is a type dependent on objects.
  - Parent(x) – it depends on objects x : Human.
  - Event ➔ Evt(h) with h:Human (events performed by h)
  - Π-types of dependent functions (see next page)
- Dependent types give, among other things:
  - Logical quantifiers (e.g., Π for ∀) in a propositions-as-types logic
  - Powerful tools in semantic construction (eg, Π-polymorphism)

Π-types: a taste of dependent types

- Πx:Human.Parent(x)
  - Type of functions, where Parent(x) is the type of x's parents.
  - f : Πx:Human.Parent(x), then
    f(h) : Parent(h), for h : Human.
- A→Prop (i.e., Πx:A.Prop)
  - Type of predicates over type A
- Π-polymorphism
  - small : ΠA:CN.(A→Prop)
  - small(Elephant) : Elephant→Prop
  - small(Mouse) : Mouse→Prop
  - small(Table) : Table→Prop

$$\frac{\Gamma \vdash A\ type \quad \Gamma,\ x{:}A \vdash B\ type}{\Gamma \vdash \Pi x{:}A.B\ type}$$

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

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

$$\frac{\Gamma,\ x{:}A \vdash b : B \quad \Gamma \vdash a : A}{\Gamma \vdash (\lambda x{:}A.b)(a) = [a/x]b : [a/x]B}$$

Montague semantics and MTT-semantics

- Two basic semantic types in MG/MTT-semantics

Category	**MG's type**	**MTT-semantic type**
S (sentence)	t	Prop
IV (verb)	$e \rightarrow t$	$A \rightarrow Prop$ (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→t**	Human→Prop
talk(john)	**t**	Prop

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 <u>"the table" : Table</u> 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

CNs as types and subtyping

- MTTs have many types (informally, collections).
  - Dependent types (Π-types, Σ-types, …)
  - Inductive types (Nat, Fin(n), …)
  - And more … (universes, logical types, …)

  Some can be used to represent CNs. (Ranta 1994, Luo 2012)
- Subtyping (necessary for multi-type languages such as MTTs)
  - Example: What if John is a man in "John talks"?
    - john : Man and talk : Human→Prop
    - talk(john)? (john is not of type Human …?)
  - Problem solved if Man ≤ Human
  - Coercive subtyping – adequate for MTTs (Luo 1997, Luo, Soloviev & Xue 2012)

Adjectival modification of CNs – case study

- A traditional classification
  - Kamp 1975, Parsons 1970, Clark 1970, Montague 1970

classification	property	example
Intersective	Adj(N) → Adj & N	handsome man
Subsectional	Adj(N) → N	large mouse
Privative	Adj(N) → ¬N	fake gun
Non-committal	Adj(N) → ?	alleged criminal

Intersective adjectives

- Example: handsome man (see next page for Σ-types)

	Montague	**MTT-semantics**
man	man : **e→t**	Man : Type
handsome	handsome : **e→t**	Man→Prop
handsome man	λx. man(x) & handsome(x)	Σ(Man,handsome)

- In general:

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

Σ-types

- An extension of the product types A x B of pairs
- Σ-types of "dependent pairs"
  - Σ(A,B) of (a,b) for a:A & b:B(a)
- Rules for Σ-types:
  - Σ(A,B) also written as Σx:A.B(x)
- Examples:
  - Σ(Human,dog)

    with dog(j)={d}, dog(m)=∅, …
  - Σ(Man,handsome)

$$(\Sigma)\quad \frac{\Gamma \vdash A\ type \quad \Gamma,\ x : A \vdash B\ type}{\Gamma \vdash \Sigma x : A.B\ type}$$

$$(pair)\quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B\ type}{\Gamma \vdash (a, b) : \Sigma x : A.B}$$

$$(\pi_1)\quad \frac{\Gamma \vdash p : \Sigma x : A.B}{\Gamma \vdash \pi_1(p) : A}$$

$$(\pi_2)\quad \frac{\Gamma \vdash p : \Sigma x : A.B}{\Gamma \vdash \pi_2(p) : [\pi_1(p)/x]B}$$

$$(proj_1)\quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B\ type}{\Gamma \vdash \pi_1(a, b) = a : A}$$

$$(proj_2)\quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B\ type}{\Gamma \vdash \pi_2(a, b) = b : [a/x]B}$$

- An adjective maps CNs to CNs:
  - In MG, predicates to predicates.
  - In MTT-semantics, types to types.
- MTT-semantics (Chatzikyriakidis & Luo 2020, Luo 2023)

classification	**example**	**types employed**
Intersective	handsome man	Σ-types with simple predicates
Subsectional	large mouse	Π-polymorphic predicates and Σ-types
Privative	fake gun	Disjoint union types with Π/Σ-types
Non-committal	alleged criminal	special predicates

I.2. Introductory Overview of “Advanced” Topics

Note:

For each topic/lecture, I shall try to cover

- History/examples in introduction
- Montague or traditional approaches
- Type-theoretical approaches

and informal/understandable.

Lecture II: Events (overview)

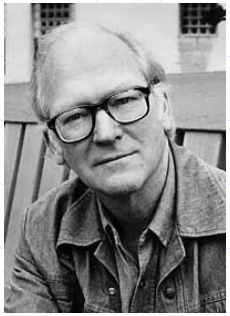
- Davidsonian event semantics
  - Original motivation: adverbial modifications

    (1) John buttered the toast.

    (2) John buttered the toast with the knife in the kitchen.

    Do we have (2) $\Rightarrow$ (1)?

    - Cumbersome in MG with meaning postulates
    - Davidson (1967): verbs tacitly introduce existentially quantified events.



- Neo-Davidsonian notation with thematic roles (eg, Parsons 1990):

  (1′) ∃v:Event. butter(v) & agent(v)=john & patient(v)=toast

  (2′) ∃v:Event. butter(v) & agent(v)=john & patient(v)=toast
  & with(v,knife) & at(v,kitchen)

  Obviously, (2′) $\Rightarrow$ (1′)

- Event semantics is an extremely popular topic
  - Casati and Varzi. Events: An Annotated Bibliography. 1997. [25 years ago, already 235 pages!]
  - Some researchers even take it for granted …
- But:
  - What is an event?
  - Are events atomic? Structured? If the latter, how?
  - Is the introduction of events completely harmless?
  - … …

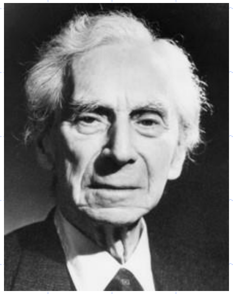
still unsettled/debated/…

Dependent event types

- Dependent event types (DETs)
  - Events are classified into "event types", dependent on (or classified according to) thematic roles.
- Focus: Montague (simple type theory) + DETs
  - Rather than MTT+DETs (stepping-stone for easier understanding)
- Several application examples
  - Solve problems such as "Event Quantification Problem"
  - Facilitate semantic constructions of tensed sentences
  - Restore "selection restriction" in MTT-event semantics

See (Luo & Soloviev 2017, Chatzikyriakidis & Luo 2020, Luo 2023).

Lecture III. Indefinites & Anaphora (overview)



- We'll discuss indefinites like "a man". Are they
  - Quantifier phrases (as Russell suggests)?
  - Referring expressions?
- Russell (1919): the ∃-view
  - A man came in. ➔ ∃x:**e**. man(x)∧come_in(x)
  - A lot of arguments/examples for the ∃-view.
- But what about, for example,
  - <u>A man</u> came in. <u>He</u> lit a cigarette. [compositional semantics?]
  - Every farmer who owns <u>a donkey</u> beats <u>it</u>.

$$(\#)\ \forall x.\ [farmer(x)\ \&\ \exists y.(donkey(y)\ \&\ own(x,y))] \Rightarrow beat(x,y)$$

Both referring to a variable outside its scope (e.g. the last "y").

Dynamic semantics

- Dynamic approaches (widely accepted for anaphora treatment)
  - Discourse Representation Theory (Kamp 1981, Heim 1982)
  - Dynamic Predicate Logic (Groenendijk and Stokhof 1991)

$$\boxed{(\exists x \varphi); \psi} \Leftrightarrow ([x]; \varphi); \psi \Leftrightarrow [x]; (\varphi; \psi) \Leftrightarrow \boxed{\exists x (\varphi; \psi)}$$

  where ";" is the dynamic conjunction (so, previous "y" would be OK …)
- However, logics in dynamic semantics are non-standard.
  - For example, DPL (G&S91) is rather non-standard:
    - non-monotonic, irreflexive/intransitive entailment, …
  - Substantial changes required for underlying logic(s) in semantics
- Two "extremes"? Anything in the middle?

  Russell (∃) |------------?------------| Dynamic

Type-theoretical approaches

- Using dependent types (Mönnich 1985, Sundholm 1986)
  - (Donkey) Every farmer who owns a donkey beats it.
  - $\forall z : F_\Sigma.\ beat(\pi_1(z), \pi_1(\pi_2(z)))$ where $F_\Sigma = \Sigma x{:}F\ \Sigma y{:}D.\ own(x, y)$
  - Σ is the "strong sum" with two projections π_1 and π_2 .
- This gives a compromise:
  - Σ is "strong" so that witnesses can be referred to outside its scope.
  - The change for underlying logic is much less substantial.
- However, a problem – Σ plays a double role:
  - Subset constructor (1^{st} Σ) and existential quantifier (2^{nd} Σ).
  - But this is problematic ➔ counting problem.
  - A satisfactory solution with both strong/weak sums (Luo 2021)

Lecture IV. Copredication (overview)

- Copredication is a special case of logical polysemy.
  - See (Pustejovsky 1995, Asher 2011), among others.
- Examples
  - (*) The lunch was delicious but took a long time.
    - delicious : Food→**t**; take_long_time : Process→**t**
    - Their domains Food/Process do not share any common objects, but they can both apply to the same noun (lunch) …
  - (**) All three books are heavy and boring.
    - heavy : Phy→**t**; boring : Info→**t**
    - Phy/Info (similar to the above) and heavy/boring can both apply to a book.

How to analyse it formally?

- Very interesting issue
  - Easy to understand, but intriguing (nice research topic)
  - Numerous papers in the literature
- Many approaches, including (just to name a few):
  - Dot-types and related approaches
    - E.g., Pustejovsky 95, Asher 2011, Luo 2010, …
  - Mereological approaches
    - E.g., Gotham 2014, 2017
  - Others
    - E.g., Retoré 2013, Liebesman & Magidor 2023, …

Dot-types in MTTs

- Dot-types – idea by Pustejovsky (1995)
  - Objects of type A•B have two aspects: being both A and B.
  - Informally, the above sentences (*)/(**) can now be interpreted.
- How to
  - formalise dot-types?
  - formalise dot-types in MTTs?
  - We'll try to explain them informally – see (Luo 2010, 2023)
- What happens when copredication interacts with …?
  - Interacting with quantification ➔ identity criteria of CNs (Luo 2012)
  - See (Chatzikyriakidis and Luo 2018, Luo 2023)

Lecture V. Reasoning, CGs, and Beyond (overview)

- More MTT-related topic(s) may be briefly introduced; examples include:
  - Natural language reasoning based on MTT-semantics in "proof assistants" – computer-assisted reasoning systems; c.f. (Chatzikyriakidis and Luo 2020, Chap 6; Luo 2023, Chap 5)
  - Dependent types in categorial grammars – substructural dependent type theory; c.f. (Luo 2023, Sect 4.5)
- This lecture is intentionally left as "open" at the moment; besides the above, it may also include some "tidying up" of previous lectures.

NL Reasoning in Proof Assistants

- Interactive theorem proving based on MTTs
- An ITP system consists of three parts for:
  (1) contextual defns (2) proof development (3) proof checking

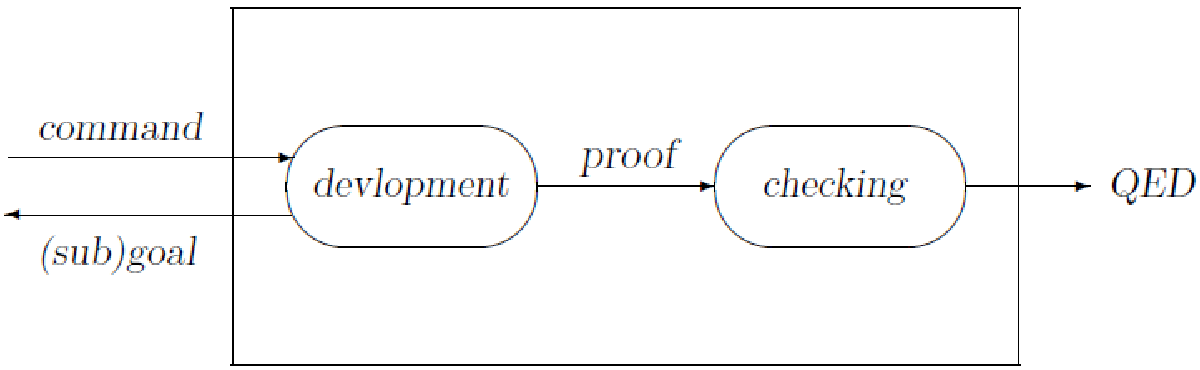


Figure 1: Interactive proof development and proof checking

Dependent Categorial Grammar

- Categorial Grammars (or typelogical grammars)
  - An approach to syntactic analysis
  - CGs are based on substructural logics
    - Moortgat: 'Typelogical grammars are substructural logics, designed for reasoning about the composition of form and meaning in natural language.' (Stanford Encyclopedia of Philosophy, 2010)
- What is a substructural logic?
  - In a proof system, there are usually three kinds of "structural" rules: weakening, contraction (strengthening), exchange
  - Weakening: adding more assumptions is OK.
  - Contraction: removing a repeated assumption is OK.
  - Exchange: swapping the order of two assumptions is OK.

  In substructural (resource-sensitive) logics, the above may not be OK.

Lambek calculus and beyond

- Categorial grammar and historical developments:
  - Ajdukiewicz, Bar, Hillel, …
- Lambek calculus (1958)
  - Ordered formulae B/A and A\B
    - John runs – "run applies to a np on the left".
      John : NP and run : NP\S
  - Resource sensitive – substructural (eg, no exchange – word order)
- Linear/hybrid CGs (Oehrle 1994, Kubota & Levine's HTLG, …)
- Substructural type theory (Luo 2023)

	Π-type	Non-dependent type	Abstraction	Application
Linear	$\overline{\Pi} x{:}A.B$	$A \multimap B$	$\overline{\lambda} x{:}A.b$	$\overline{app}(f,a)$
Ordered (right)	$\Pi^r x{:}A.B$	B/A	$\lambda^r x{:}A.b$	$app^r(f,a)$
Ordered (left)	$\Pi^l x{:}A.B$	$A \setminus B$	$\lambda^l x{:}A.b$	$app^l(a,f)$

Table 1 Three substructural function types in $\bar{\lambda}_\Pi$: summary of notations.

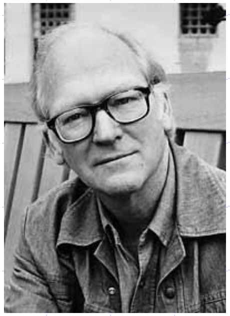
Lecture II. Event Semantics

This lecture

1. Davidsonian event semantics
2. Dependent event types
   - DETs in simple type theory (Montague's setting)
     - Focus: stepping stone for easier understanding
     - Adequacy: conservativity over Church's simple type theory
   - DETs in modern type theories (MTT-event semantics)
3. Three applications of DETs
   - Event quantification problem and its DET solution
   - Temporal semantic constructions (*)
   - Selection restriction in MTT-event semantics (*)

See (Luo & Soloviev 2017, Chatzikyriakidis & Luo 2020, Luo 2023), where those marked with (*) are new.

II.1. Davidsonian event semantics



- Original motivation: adverbial modifications
  (1) John buttered the toast.
  (2) John buttered the toast with the knife in the kitchen.
- Do we have (2) $\Rightarrow$ (1)? How?
  - Cumbersome in MG with meaning postulates
  - Davidson (1967): verbs tacitly introduce existentially quantified events, doing away with meaning postulates.

Two MG approaches without events

- (1) John buttered the toast.
  - (1″) butter(j,t)
  - Here, butter : $\mathbf{e}^2 \rightarrow \mathbf{t}$ and j, t : $\mathbf{e}$
- (2) John buttered the toast with the knife in the kitchen.
  - A1: change type of butter to butter* : $\mathbf{e}^4 \rightarrow \mathbf{t}$, with k_1, k_2 : $\mathbf{e}$
    (2″) butter*(j,t,k_1,k_2)
  - A2: keep butter : $\mathbf{e}^2 \rightarrow \mathbf{t}$, with knife/kitchen : $(\mathbf{e} \rightarrow \mathbf{t}) \rightarrow (\mathbf{e} \rightarrow \mathbf{t})$
    (2‴) kitchen(knife(butter(j)))(t)
- Both need *ad hoc* <u>meaning postulates</u> to get (2″)/(2‴) $\Rightarrow$ (1″).
  - E.g., we may assume $\forall x{:}\mathbf{e}$.knife(p,x)/kitchen(p,x)$\Rightarrow$p(x), then (2‴) $\Rightarrow$ (1″).

Neo-Davidsonian event semantics

- Neo-Davidsonian (Parsons 1990) with thematic roles (next slide)

(1) John buttered the toast.

(1′) $\exists$v:Event. butter(v) & agent(v)=john & patient(v)=toast

(2) John buttered the toast with the knife in the kitchen.

(2′) $\exists$v:Event. butter(v) & agent(v)=john & patient(v)=toast & with(v,knife) & at(v,kitchen)

Obviously, (2′) $\Rightarrow$ (1′)

Thematic roles like agent/patient/time

Major thematic relations [edit]

Here is a list of the major thematic relations.[3]

- **Agent**: deliberately performs the action (e.g., **Bill** ate his soup quietly.).
- **Experiencer**: the entity that receives sensory or emotional input (e.g. **Susan** heard the song. **I** cried.).
- **Stimulus**: Entity that prompts sensory or emotional feeling - not deliberately (e.g. David Peterson detests **onions!**).
- **Theme**: undergoes the action but does not change its state (e.g., We believe in one **God**. I have **two children**. I put **the book** on the table. He gave **the gun** to the police officer.) (Sometimes used interchangeably with patient.)
- **Patient**: undergoes the action and changes its state (e.g., The falling rocks crushed **the car**.). (Sometimes used interchangeably with theme.)
- **Instrument**: used to carry out the action (e.g., Jamie cut the ribbon **with a pair of scissors**.).
- **Force** or **Natural Cause**: mindlessly performs the action (e.g., **An avalanche** destroyed the ancient temple.).
- **Location**: where the action occurs (e.g., Johnny and Linda played carelessly **in the park**. I'll be **at Julie's house** studying for my test.).
- **Direction** or **Goal**: where the action is directed towards (e.g., The caravan continued on **toward the distant oasis**. He walked **to school**.).
- **Recipient**: a special kind of goal associated with verbs expressing a change in ownership, possession. (E.g., I sent **John** the letter. He gave the book **to her**.)
- **Source** or **Origin**: where the action originated (e.g., The rocket was launched **from Central Command**. She walked **away from him**.).
- **Time**: the time at which the action occurs (e.g., The pitcher struck out nine batters **today**)
- **Beneficiary**: the entity for whose benefit the action occurs (e.g.. I baked **Reggie** a cake. He built a car **for me**. I fight **for the king**.).
- **Manner**: the way in which an action is carried out (e.g., **With great urgency**, Tabitha phoned 911.).
- **Purpose**: the reason for which an action is performed (e.g., Tabitha phoned 911 right away **in order to get some help**.).
- **Cause**: what caused the action to occur in the first place; not *for what*, rather *because of what* (e.g., **Because Clyde was hungry**, he ate the cake.).

Events? Event structure?

- What is an event?
  - Mysterious concept … Philosophically argued for (and against …)
  - Are they individuals/entities? Event < **e**? Formally, either is possible – we leave it open.
  - Do events have structures/properties/classifications?
- We propose to introduce
  - Dependent event types (DETs), dependent on thematic roles
- This
  - Solves the problems such as "EQP" (see later)
  - Facilitates semantic constructions of tensed sentences
  - Solves selection restriction problem in MTT-event semantics

but doesn't attempt to answer the above questions.

II.2. Dependent event types

- Dependent event types (Luo & Soloviev 2017)
  - Refining event structure by (dependent) typing
- How:

Refining event structure:

Event ➔ Evt(a)/Evt(a,p)/Evt(a,p,t)

which are event types dependent on thematic roles

a/p/t (agents/patients/times),

respectively.

DETs and their subtyping relationships

- For a:Agent and p:Patient, consider DETs

  $$\text{Event},\ \text{Evt}_A(a),\ \text{Evt}_P(p),\ \text{Evt}_{AP}(a,p)$$

- Subtyping ($A \leq B$ means that any a of type A is also of type B)

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

- Subtyping between DETs

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

  - Any event with agent a and patient p is an event with agent a.
  - Any event with agent a is an event.

Two systems with DETs

- Extension of Montague's simple TT with DETs
  - C_e extends Church's simple type theory (1940) with DETs
  - Montague's system is familiar for many – hopefully better understanding of DETs.

We shall focus on this – stepping stone for easier understanding.

- Extension of modern type theories with DETs
  - T[E] extends MTT T with DETs; e.g., T = UTT (Luo 1994).
  - This shows how DETs work with MTTs – "MTT-event sem."

Only informally/briefly in dealing with selection restriction in MTT-event semantics

DETs 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 $Evt_A(j) \leq Event$.

C_e: Church's simple TT with DETs (Luo 2023)

- First, Church's simple type theory C (1940)
  - Employed in Montague's semantics (c.f., Gallin 1975)
  - Its rules are presented in the Natural Deduction style as follows.
- Rules for sorts/judgements and λ-calculus

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

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

Note: the side condition in the λ-rule is there only for DETs.

- Rules for truth of logical formulas

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

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

- Rule for “conversion” of logical formulas (λ-conversion omitted)

$$\frac{P\ true \quad Q : \mathbf{t}}{Q\ true} \quad (P \simeq Q)$$

Dependent event types in C_e

$$\frac{\Gamma\ valid}{\Gamma \vdash Event\ type}$$

$$\frac{\Gamma \vdash a : \mathbf{e}}{\Gamma \vdash Evt_A(a)\ type} \qquad \frac{\Gamma \vdash p : \mathbf{e}}{\Gamma \vdash Evt_P(p)\ type} \qquad \frac{\Gamma \vdash a : \mathbf{e} \quad \Gamma \vdash p : \mathbf{e}}{\Gamma \vdash Evt_{AP}(a,p)\ type}$$

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

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

$$\begin{tikzcd}
 & Evt_A(a) \arrow[dr, no head, "\leq"] & \\
Evt_{AP}(a,p) \arrow[ur, no head, "\leq"] \arrow[dr, no head, "\leq"'] & & Event \\
 & Evt_P(p) \arrow[ur, no head, "\leq"'] &
\end{tikzcd}$$

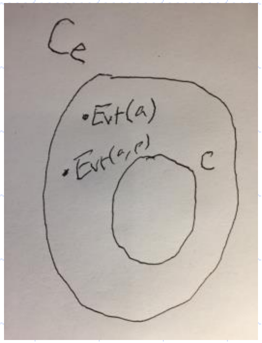
Conservativity (Luo & Soloviev 2020, Luo 2023)

Background notes

(1) Conservative extension: "J in C and |- J in C_e, then |- J in C."

(2) Logical consistency is preserved by conservative extensions.

Theorem. C_e is a conservative extension over Church's simple type theory.



Proof. Define R : $C_e \rightarrow$ C that preserves derivations.

- R maps event types (DETs) Event/Evt(…) to **e**.
- R(t)=t for t$\in$C.

For any C_e-derivation D, R(D) is a C-derivation. Therefore, any derivable C-judgement in C_e can also be derived in C.

Corollary. C_e is logically consistent.

II.3. Applications of DETs

- In this course, three applications of DETs:
  - DET solution to event quantification problem (EQP)
  - Temporal semantic constructions with DETs
  - Selection restriction in MTT-event semantics

II.3.1. Incompatibility problems in event sem.

- Introducing an extra/artificial existential event quantifier "$\exists v$" may lead to interference with other quantifiers.
  - E.g., "event quantification problem" (EQP, Winter & Zwarts 2011)
  - Incompatibility between event semantics and MG (Champollion 2015)

(1) Nobody talked.

Intended neo-Davidsonian event semantics is (2):

(2) $\neg\exists x{:}\mathbf{e}.\ [\mathrm{human}(x)\ \&\ \exists v{:}\mathrm{Event}.\ \mathrm{talk}(v)\ \&\ \mathrm{agent}(v){=}x]$

But the incorrect semantics (#) is also possible (well-typed!)

(#) $\exists v{:}\mathrm{Event}.\ \neg\exists x{:}\mathbf{e}.\ \mathrm{human}(x)\ \&\ \mathrm{talk}(v)\ \&\ \mathrm{agent}(v){=}x$

It moves the event quantifier "$\exists v$:Event" in (2) to the beginning.

Some proposed solutions to EQP

- Many different proposals (only mentioning two below)
  - Purpose: to force scope of event quantifier to be narrower.
- Champollion's quantificational event semantics (2015)
  - Trick: taking a <u>set</u> E of events as argument, but **talk**(e) …
    - talk : (Event→**t**)→**t** with talk(E) = ∃e:Event. e∈E & **talk**(e)
  - Debatable: intuitive meanings, compositionality & complexity
- Winter-Zwarts (2011) & de Groote (2014)
  - Use Abstract Categorial Grammar (ACG, de Groote 2001)
    - ACG structure prevents incorrect interpretation.
  - Seemingly coincidental (and what if one does not use ACG?)
- Our proposal: dependent event types (solution to EQP & …)

DET-solution to EQP

(1) Nobody talked.

Neo-Davidsonian semantics (repeated):

(2) $\neg\exists x{:}\mathbf{e}.\ \text{human}(x)\ \&\ \exists v{:}\text{Event}.\ \text{talk}(v)\ \&\ \text{agent}(v,x)$

(3) $\exists v{:}\text{Event}.\ \neg\exists x{:}\mathbf{e}.\ \text{human}(x)\ \&\ \text{talk}(v)\ \&\ \text{agent}(v,x)$

where (2) is intended, while (3) is <u>incorrect</u>, but <u>well-typed</u>.

Dependent event types in Montague's setting:

(4) $\neg\exists x{:}\mathbf{e}.\ \text{human}(x)\ \&\ \exists v{:}\text{Evt}_A(x).\ \text{talk}(v)$

(#) $\exists v{:}\text{Evt}_A(x).\ \neg\exists x{:}\mathbf{e}.\ \text{human}(x)\ \&\ \text{talk}(v)$

where (#) is <u>ill-typed</u> since the first "x" is outside scope of "$\exists x{:}\mathbf{e}$".

II.3.2. Tense and time-indexed DETs

- Event typed dependent on times, for example:
  - $Evt_{AT}(a,t)$: type of events whose agents are a and which occur at time t.
  - $Evt_{AT^2}(a,t_1,t_2)$: type of events whose agents are a and which occur during interval (t_1,t_2).
- A simple model of time
  - Time (a type)
  - < : Time → Time → **t**
  - Corresponding relation ≤ is a total order.

$$\frac{a:\mathbf{e} \quad t:Time}{Evt_{AT}(a,t) \ \ type} \qquad \frac{a:\mathbf{e} \quad t_1:Time \quad t_2:Time}{Evt_{AT^2}(a,t_1,t_2) \ \ type}$$

- Intervals as predicates: $t \in (t_1,t_2)$ means $t_1 < t < t_2$.
  - Similarly for the other intervals $[t_1,t_2]$, $(t_1,t_2]$ and $[t_1,t_2)$.

DET-semantics of tensed sentences

- Let's assume
  - now : Time (standing for the speech time)
  - ref : Time (standing for the reference time)

Example	Event semantics with DETs
John is talking	$\exists e{:}Evt_{AT}(j, now).\ talk(e)$
John talked	$\exists t{:}Time.\ t < now \wedge \exists e{:}Evt_{AT}(j, t).\ talk(e)$
John will talk	$\exists t{:}Time.\ now < t \wedge \exists e{:}Evt_{AT}(j, t).\ talk(e)$
John had talked	$\exists t{:}Time.\ t < ref < now \wedge \exists e{:}Evt_{AT}(j, t).\ talk(e)$
John will have talked	$\exists t{:}Time.\ now < t < ref \wedge \exists e{:}Evt_{AT}(j, t).\ talk(e)$

Table 1: Simple examples in event semantics with DETs

Remarks

- Temporal logic?
  - Numerous work based on traditional logics such as propositional logic or FOL (Prior (1967), van Benthem 1991, …)
    - A workshop at this ESSLLI (focusing on non-linguistic issues)
  - Unclear how to study modal/temporal logics for MTTs (on-going, mainly model-theoretically; unclear at all proof-theoretically)
- How to relate events with time/tense?
  - Event ➔ time (in set theory; Kamp 1979)
    - Question: how can one benefit from such connections?
  - In DETs, we only assume that events are dependent on their occurrence times, but that's all.
    - Is this appropriate? Otherwise, what …?

II.3.3. MTT-event sem. and selection restriction

- Events can similarly be introduced into MTT-semantics.
  - Original motivations (eg, better adverbial modification) still applies.
  - It also leads to problems such as EQP.
  - DETs can be introduced in MTT-semantics, solving EQP etc.

  Exactly similar as in the Montagovian setting – omitted here.
- MTT-event semantics: a brief description
  - Let T be any modern type theory such as UTT (Luo 1994) and E the basic coercions characterizing DET-subtyping.
  - Then, $T_e[E]$ extends T with DET-subtyping (next page; Luo 2023).

$T_e[E]$ (presentation in LF, here only for completeness)

- Constant types/families:
  - $Agent,\ Patient : Type.$
  - $Event : Type,$
    $Evt_A : (Agent)Type,$
    $Evt_P : (Patient)Type,$ 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 $c_3[a] \circ c_1[a,p] = c_4[p] \circ c_2[a,p].$

- $T_e[E]$ has nice properties such as normalisation and consistency if T does (Luo, Soloviev & Xue 2012, Luo 2023).

Selection restriction in MTT-event semantics

- (#) Tables talk.
  - Montague: $\forall x{:}\mathbf{e}.talk(x)$ – well-typed but false (talk : $\mathbf{e}\rightarrow\mathbf{t}$)
  - MTT-sem: $\forall x{:}Table.talk(x)$ – ill-typed (talk : $Human\rightarrow Prop$)
- What happens when we have events? (talk : $Event\rightarrow\mathbf{t}/Prop$)
  - Montague: $\forall x{:}\mathbf{e}\ \exists v{:}Event.\ talk(v)\ \&\ agent(v)=x$ (well-typed)
  - MTT-sem: $\forall x{:}Table\ \exists v{:}Evt_A(x).\ talk(v)$

    where we have $Table \leq Agent$. (Also well-typed!)

So? How to recover?

  - There are several approaches (Luo 2018).
  - We'll introduce "DETs with domains", the most flexible one.

DETs with domains

- Refined DETs with "domains" (Consider subtypes of Agent, wlg.)
  - Let $D \leq_k$ Agent.
    - $Evt_A[D] : D \rightarrow Type$
    - $Evt_A[D](d) = Evt_A(k(d))$

$$\frac{\langle\rangle \vdash D \leq_\kappa Agent \quad \Gamma \vdash d : D}{\Gamma \vdash Evt_A[D](d) = Evt_A(\kappa(d))}$$

  - Note: this is only a definitional extension.
- Examples
  - Men talk.
    - $\forall x{:}Man\ \exists v{:}Evt_A[Human](x).\ talk(v)$ (OK because Man≤Human)
  - Tables talk.
    - (#) $\forall x{:}Table\ \exists v{:}Evt_A[Human](x).\ talk(v)$ (ill-typed - x is not a human.)
  - John picked up and mastered the book.
    - $\exists v{:}Evt_{AP}[Human, P\bullet I](j,b).\ pick\text{-}up(v)\ \&\ master(v)$, where $b : Book \leq P\bullet I$

Related (and some future) work on DETs

- Original idea
  - Came from my treatment of an example in (Asher & Luo 2012)
    - Evt(h) to represent collection of events conducted by h : Human.
  - Further prompted by de Groote's talk at LENLS14 (on EQP etc.)
- Other applications of DETs
  - For example, problem with negation in event semantics
- DETs dependent on other parameters
  - Dependency on other kinds of parameters than thematic roles?

Lecture III. Indefinites and Anaphora

- (Recap) MTT-semantics for adjectival modification
  - Left from Lecture I
  - Σ-types for the following Lecture III

Adjectival modification of CNs – case study

- A traditional classification
  - Kamp 1975, Parsons 1970, Clark 1970, Montague 1970

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

Intersective adjectives

- Example: handsome man (see next page for Σ-types)

	Montague	**MTT-semantics**
man	man : **e→t**	Man : Type
handsome	handsome : **e→t**	Man→Prop
handsome man	λx. man(x) & handsome(x)	Σ(Man,handsome)

- In general:

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

Σ-types

- An extension of the product types A x B of pairs
- Σ-types of "dependent pairs"
  - Σ(A,B) of (a,b) for a:A & b:B(a)
- Rules for Σ-types:
  - Σ(A,B) also written as Σx:A.B(x)
- Examples:
  - Σ(Human,dog)

    with dog(j)={d}, dog(m)=∅, ...
  - Σ(Man,handsome)

$$(\Sigma)\quad \frac{\Gamma \vdash A\ type \quad \Gamma,\ x : A \vdash B\ type}{\Gamma \vdash \Sigma x : A.B\ type}$$

$$(pair)\quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B\ type}{\Gamma \vdash (a,b) : \Sigma x : A.B}$$

$$(\pi_1)\quad \frac{\Gamma \vdash p : \Sigma x : A.B}{\Gamma \vdash \pi_1(p) : A}$$

$$(\pi_2)\quad \frac{\Gamma \vdash p : \Sigma x : A.B}{\Gamma \vdash \pi_2(p) : [\pi_1(p)/x]B}$$

$$(proj_1)\quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B\ type}{\Gamma \vdash \pi_1(a,b) = a : A}$$

$$(proj_2)\quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B\ type}{\Gamma \vdash \pi_2(a,b) = b : [a/x]B}$$

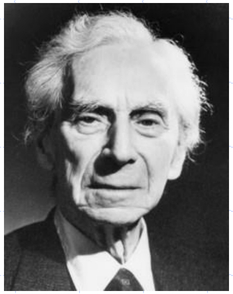
- An adjective maps CNs to CNs:
  - In MG, predicates to predicates.
  - In MTT-semantics, types to types.
- MTT-semantics (Chatzikyriakidis & Luo 2020, Luo 2023)

classification	**example**	**types employed**
Intersective	handsome man	Σ-types with simple predicates
Subsectional	large mouse	Π-polymorphic predicates and Σ-types
Privative	fake gun	Disjoint union types with Π/Σ-types
Non-committal	alleged criminal	special predicates

This lecture

1. Indefinites and the Russellian ∃-view
2. Dynamic semantics
3. Type-theoretical approach
4. Problem with the type-theoretic approach and solution with both strong/weak sums (possibly in Lecture IV)

III.1. Indefinites and Russellian ∃-view



- We'll discuss indefinites like "a man". Are they
  - Quantifier phrases (as Russell suggests)?
  - Referring expressions?
- Russell (1919): the ∃-view
  - A man came in. ➔ $\exists x{:}\mathbf{e}.\ \text{man}(x) \wedge \text{come\_in}(x)$
- Arguments/examples in favour of the ∃-view
  - John saw <u>a dog</u> and Mary saw <u>a dog</u>, too.

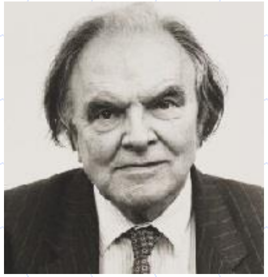
    [Could be different dogs. Russell's ∃-view predicts it.]

    [Different "a dog" could refer to different things. c.f., He likes him.]
  - It is not the case that <u>a man</u> came in.
    Every child owns <u>a dog</u>.

    [Not a particular man/different dogs. Russell's ∃-view predicts it.]

- But what about, for example,
  - (1) A man came in. He lit a cigarette.

    (#) $[\exists x{:}\mathbf{e}.\mathrm{man}(x) \wedge \mathrm{come\_in}(x)]$ $[\exists y{:}\mathbf{e}.\mathrm{cigarette}(y) \wedge \mathrm{light}(x_?, y)]$

    Geach's proposed solution (1962): put the latter into the scope of $\exists x$. But, this is non-compositional …
  - (2) Every farmer who owns a donkey beats it.

    (#) $\forall x{:}\mathbf{e}.\ \mathrm{farmer}(x) \wedge \exists y{:}\mathbf{e}.(\mathrm{donkey}(y) \wedge \mathrm{own}(x,y) \rightarrow \mathrm{beat}(x,y)$
  - (3) Every person who buys a TV and has a credit card uses it to pay for it.
- In the above sentences, "it" seems to refer to something
  - Variable? E.g., $x_?$ for (1) and the last "y" for (2)
  - But they are outside their scopes!

III.2. Dynamic semantics

- Dynamic approaches (widely accepted for anaphora treatment)
  - Discourse Representation Theory (Kamp 1981, Heim 1982)
  - Dynamic Predicate Logic (Groenendijk and Stokhof 1991)

$$\boxed{(\exists x\varphi);\psi} \Leftrightarrow ([x];\varphi);\psi \Leftrightarrow [x];(\varphi;\psi) \Leftrightarrow \boxed{\exists x(\varphi;\psi)}$$

$$\boxed{(\exists x\varphi) \rightarrow \psi} \Leftrightarrow ([x];\varphi) \rightarrow \psi \Leftrightarrow [x] \rightarrow (\varphi \rightarrow \psi) \Leftrightarrow \boxed{\forall x(\varphi \rightarrow \psi)}$$

where ";" is the dynamic conjunction and ψ may have free x !

  - So, if we replace $\wedge$ by ; then $x_?$ and "y" in previous interpretations would be OK (because of the above equivalences)!

    ∀x:**e**. [farmer(x) ; ∃y:**e**.(donkey(y) ; own(x,y)] → beat(x,y)

    ⇔ ∀x:**e**∀y:**e**. [farmer(x) ; donkey(y) ; own(x,y)] → beat(x,y)

    This equivalence is true because of the above 2^{nd} equivalence.

- However, logics in dynamic semantics are rather non-standard.
  - Dynamic Predicate Logic (DPL, Groenendijk and Stokhof 1991) is non-monotonic, has irreflexive/intransitive entailment, …
  - Substantial changes required for underlying logic(s) in semantics
- Two "extremes"? Anything "in the middle"?

  Russell (∃) |------------?------------| Dynamic

- Σ-types in MTTs may provide such a "middle" solution!

III.3. Type-theoretical approach

- Using dependent types (Mönnich 1985, Sundholm 1986)
- Every farmer who owns a donkey beats it.
  - (#) $\forall x{:}\mathbf{e}.\ \text{farmer}(x) \wedge \exists y{:}\mathbf{e}.(\text{donkey}(y) \wedge \text{own}(x,y) \rightarrow \text{beat}(x,y)$
- In type theory, we could give semantics as follows:
  - $\forall z : [\Sigma x{:}\text{Farmer}\ \Sigma y{:}\text{Donkey}.\ \text{Own}(x,y)].\ \text{Beat}(\pi_1(z), \pi_1(\pi_2(z)))$
  - Σ is the "strong sum" with two projections π_1 and π_2 .
  - Therefore, "it" refers to "a donkey" – by means of π_i, as $\pi_1(\pi_2(z))$
- This gives a compromise – something "in the middle" – see below.

Σ-types (recap)

- An extension of the product types A x B of pairs
- Σ-types of "dependent pairs"
  - Σ(A,B) of (a,b) for a:A & b:B(a)
- Rules for Σ-types:
  - Σ(A,B) also written as Σx:A.B(x)
- Examples:
  - Σ(Human,dog)

    with dog(j)={d}, dog(m)=∅, …
  - Σ(Man,handsome)

$$(\Sigma)\quad \frac{\Gamma \vdash A\ type \quad \Gamma,\ x : A \vdash B\ type}{\Gamma \vdash \Sigma x : A.B\ type}$$

$$(pair)\quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B\ type}{\Gamma \vdash (a,b) : \Sigma x : A.B}$$

$$(\pi_1)\quad \frac{\Gamma \vdash p : \Sigma x : A.B}{\Gamma \vdash \pi_1(p) : A}$$

$$(\pi_2)\quad \frac{\Gamma \vdash p : \Sigma x : A.B}{\Gamma \vdash \pi_2(p) : [\pi_1(p)/x]B}$$

$$(proj_1)\quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B\ type}{\Gamma \vdash \pi_1(a,b) = a : A}$$

$$(proj_2)\quad \frac{\Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B \quad \Gamma, x : A \vdash B\ type}{\Gamma \vdash \pi_2(a,b) = b : [a/x]B}$$

So, in more details:

- Σ is a quantifier – $\Sigma x{:}A.P(x)$
  - Quantifying over x in the scope P(x).
  - $\Sigma x{:}Man.handsome(x)$
  - $\Sigma y{:}Donkey.own(j,y)$
- Σ is like the existential quantifier $\exists$
  - $\exists y{:}Donkey.own(j,y)$

  but different: it has the first projection π_1:

$$(a,b) : \Sigma x{:}A.P(x) \rightarrow \pi_1(a,b) = a$$

- This first projection does not exist for $\exists$. That's why Σ is also called the "strong sum", while $\exists$ the "weak sum".

- Two "extremes"? Anything "in the middle"?

  Russell (∃) |------------?------------| Dynamic

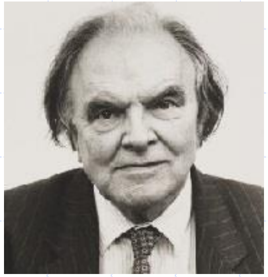
- Σ-types in MTTs may provide such a "middle" solution!
  - Σ is "strong" so that witnesses can be referred to outside its scope (by means of π_1 and π_2).
  - The change for the underlying logic is much less substantial in the sense that we just use Σ instead of ∃.
- However, still a (minor?) problem – see below.

A problem

- Σ has played two related but different roles.
  - "Subset":
    - Σx:Farmer. P(x) for "the farmers such that P holds"
  - Existential:
    - Σx:Farmer Σy:Donkey.own(x,y) for "the farmers who own a donkey"
- This is problematic ➔ counting problem.
  - Satisfactory solution with both strong/weak sums (Luo 2021)
  - We'll use donkey anaphora as a case study.

- (III.4 is moved to Lecture IV)

III.4. Donkey anaphora: problem and solution (Luo 2021)



- Examples (Geach 1962, …)

  (1) Every farmer who owns a donkey beats it.

  (2) Every person who buys a TV and
  has a credit card uses it to pay for it.

- Strong/weak readings (Chierchia 1990):
  - Strong reading of (1):

    Every farmer who owns a donkey beats every donkey s/he owns.
  - Weak reading of (1):

    Every farmer who owns a donkey beats some donkeys s/he owns.

Original prob & use of dep types (recap)

- Every farmer who owns a donkey beats it.
- In traditional logics:
  - $(\#)\ \forall x.\ [farmer(x)\ \&\ \exists y.(donkey(y)\ \&\ own(x,y))] \Rightarrow beat(x,y)$
    where $\exists$ is a "weak sum" and the last y is outside its scope.
- Using dependent types (Mönnich 85, Sundholm 86)
  - $\forall z : F_\Sigma.\ beat(\pi_1(z), \pi_1(\pi_2(z)))$ with $F_\Sigma = \Sigma x{:}F\ \Sigma y{:}D.\ own(x,y)$
    where Σ is the "strong sum" with two projections π_1 and π_2.
  - Note: the interpretation only conforms to the strong reading.
- Σ plays a double role:
  - subset constructor (1st Σ) and existential quantifier (2nd Σ).
  - But this is problematic ➔ counting problem.

Problem of counting (Sundholm 1989, Tanaka 2015)

- Cardinality of finite types (c.f., Luo 2021)
  - $|A| = n$ if $A \cong Fin(n)$ (Fin(n) has exactly n objects – see next page)
- Consider the donkey sentence with "most":
  - Most farmers who own a donkey beat it.
  - $Most_S\ z : F_\Sigma .\ beat(\pi_1(z), \pi_1(\pi_2(z)))$ with $F_\Sigma = \Sigma x{:}F\ \Sigma y{:}D.\ own(x,y)$
- But, this is inadequate – failing to "count" correctly:
  - $|F_\Sigma|$ = the number of (x,y,p) $\neq$ #(donkey-owning farmers)
    - E.g., 10 farmers:
    - 1 owns 20 donkeys and beats all of them, and
    - the other 9 own 1 donkey each and do not beat them.
    - The above sentence with "most" could be true – incorrect semantics.
  - C.f., the "proportion problem" in using DRT to do this.

D *Most* in UTT

Let A be a finite type with $|A| = n_A$, $P : A \to Prop$ a predicate over A, and $Fin(n)$ the types with n objects defined in Appendix B. Then, in UTT, the logical proposition $Most\ x{:}A.P(x)$ of type $Prop$ is defined as follows, where $inj(f)$ is a proposition expressing that f is an injective function:

$$\begin{aligned} Most\ x{:}A.P(x) = \exists k : N.\ (k \geq \lfloor n_A/2 \rfloor + 1) \\ \wedge \exists f{:}Fin(k) \to A.\ inj(f) \wedge \forall x{:}Fin(k).P(f(x)) \end{aligned}$$

The type $Fin(n)$, indexed by $n : N$ with N being the type of natural numbers, consists of exactly n objects and can be specified by means of the following introduction rules (we omit their elimination and computation rules):

$$\frac{n : N}{zero(n) : Fin(n+1)}$$

$$\frac{n : N \quad i : Fin(n)}{succ(n,i) : Fin(n+1)}$$

Why and …?

- “Double role” by Σ in F_Σ=Σx:FarmerΣy:Donkey.own(x,y)
  - First Σ: representing the collection of farmers such that …
  - Second Σ: representing the existential quantifier (!)
- But, unlike traditional ∃, Σ is strong:
  - |Σx:A.B(x)| is the number of pairs (a,b), not just the number of a’s such that B(a) is true. So, the 2nd Σ is problematic.
- Can we somehow replace the 2nd Σ by ∃?
  - Yes, although not directly (c.f., the original scope problem), by considering different readings of donkey sentences <u>AND IF</u> we have both Σ and ∃ in the type theory.
  - Note: ∃ in Montague’s simple TT and Σ in Martin-Löf’s TT, but <u>not both</u> in either of them.

UTT (Luo 1994): a type theory with both Σ/∃

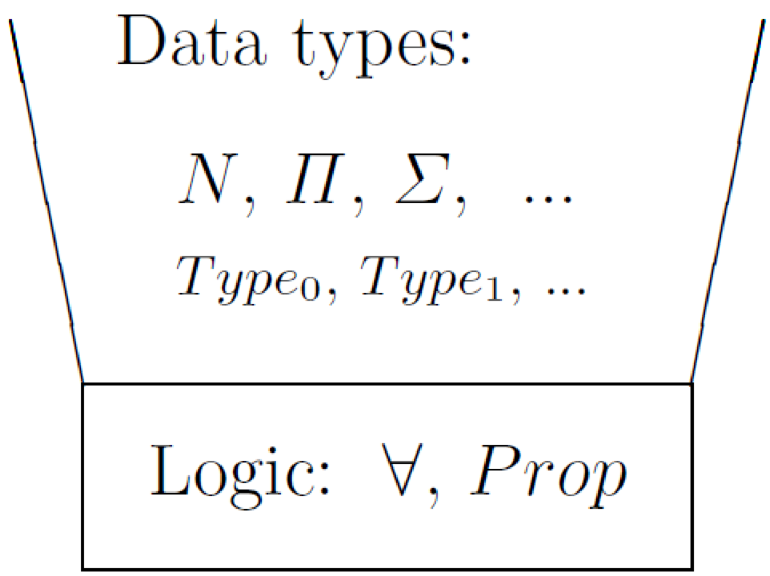


Fig. 1. The type structure in UTT.

Logic in UTT and proof irrelevance (brief)

- Formulas/propositions: ∀x:A.P, ∃x:A.P, P⇒Q, …
  - For example: $\exists x : A.P(x) = \forall X : Prop.\ (\forall x : A.(P(x) \Rightarrow X)) \Rightarrow X$
- Proof irrelevance:
  - Every two proofs of the same proposition are the same.
  - In UTT, this can be enforced by the following rule:

$$\frac{P : Prop \quad p : P \quad q : P}{p = q : P}$$

  - Note: This wouldn't be possible for Martin-Löf's type theory.
- As a consequence, we have, for example:
  - $|P| \leq 1$, if P : Prop (e.g., $|\exists x{:}A.R| \leq 1$)
  - $|\Sigma x{:}A.Q(x)| \leq |A|$, if A is a finite type and $Q : A \rightarrow Prop$

Donkey sentences in UTT

- Most farmers who own a donkey beat it.
  - (strong) Most farmers who own a donkey beat every donkey they own.
  - (weak) Most farmers who own a donkey beat some donkeys they own.
- "Most" in UTT (formal details next page)
  - Definition similar to (Sundholm 1989), but with $\exists$ as existential quantifier, instead of Σ.
- Semantics in UTT

$$F_\exists = \Sigma x{:}F.\ \exists y{:}D.own(x,y)$$

  - Strong interpretation:

$$Most\ \ z : F_\exists.\ \forall y' : \Sigma y{:}D.own(\pi_1(z),y).\ beat(\pi_1(z),\pi_1(y'))$$

  - Weak interpretation

$$Most\ \ z : F_\exists.\ \exists y' : \Sigma y{:}D.own(\pi_1(z),y).\ beat(\pi_1(z),\pi_1(y'))$$

D *Most* in UTT

Let A be a finite type with $|A| = n_A$, $P : A \to Prop$ a predicate over A, and $Fin(n)$ the types with n objects defined in Appendix B. Then, in UTT, the logical proposition $Most\ x{:}A.P(x)$ of type $Prop$ is defined as follows, where $inj(f)$ is a proposition expressing that f is an injective function:

$$\begin{aligned} Most\ x{:}A.P(x) = \exists k : N.\ & (k \geq \lfloor n_A/2 \rfloor + 1) \\ & \wedge \exists f{:}Fin(k) \to A.\ inj(f) \wedge \forall x{:}Fin(k).P(f(x)) \end{aligned}$$

The type $Fin(n)$, indexed by $n : N$ with N being the type of natural numbers, consists of exactly n objects and can be specified by means of the following introduction rules (we omit their elimination and computation rules):

$$\frac{n : N}{zero(n) : Fin(n+1)}$$

$$\frac{n : N \quad i : Fin(n)}{succ(n,i) : Fin(n+1)}$$

Another example

- Every person who buys a TV and has a credit card uses it to pay for it.
  - where "a TV" obtains a strong $\forall$-reading and "a credit card" a weak $\exists$-reading.

$$
\begin{aligned}
&\forall z : \Sigma x{:}Person.\ \exists y_1{:}TV.\ buy(x, y_1) \wedge \exists y_2{:}Card.\ own(x, y_2) \\
&\forall y : \Sigma y_1{:}TV.\ buy(\pi_1(z), y_1) \\
&\exists y' : \Sigma y_2{:}Card.\ own(\pi_1(z), y_2). \\
&\quad pay(\pi_1(z), \pi_1(y), \pi_1(y'))
\end{aligned}
$$

- Note: It would be impossible to do this in MLTT.

E-type Anaphora (Evans 77, …) (*)

- Evans' example:
  - Few congressmen admire Kennedy, and <u>they</u> are very junior.
  - Few congressmen admire Kennedy, and <u>the congressmen who do admire Kennedy</u> are very junior.
- Note: "they" cannot be "bound" by "Few congressmen" for, otherwise, the meaning is different.
  - It would mean: Few congressmen are such that they admire Kennedy and are very junior (at the same time).
- Interpretations in type theory:

$$Few\ x{:}C.admire(x,K) \wedge \forall z{:}[\Sigma x{:}C.admire(x,K)].junior(\pi_1(z))$$

  - Link of Σ with descriptions (Martin-Löf, Carlström, Mineshima)

Combining strong and weak sums (*)

- How to add Σ to an impredicative type theory with ∃-propositions?

$$\Sigma \xrightarrow{\ ?\ } \boxed{\exists}$$

- Three possibilities:
  - UTT (seen before): Σ-types + ∃-propositions
  - "Large" Σ-propositions
    ➔ logical inconsistency

$$\frac{A\ type \quad P : A \to Prop}{\Sigma x{:}A.P(x) : Prop}$$

  - "Small" Σ-propositions
    ➔ weak ∃ becoming strong

$$\frac{A : Prop \quad P : A \to Prop}{\Sigma x{:}A.P(x) : Prop}$$

Conclusion: Only the UTT's approach is OK.

(*)

- How to add ∃ to a predicative type theory with Σ-types?

$$\exists \xrightarrow{?} \boxed{\Sigma}$$

- Not clear how to do this (but see next page for $\mathrm{MLTT_h}$)
  - One might define ∃ by Π in predicative universes U_i:

$$\exists_i x{:}A.B(x) = \Pi X{:}U_i.\ (\Pi x{:}A.(B(x) \to X)) \to X$$

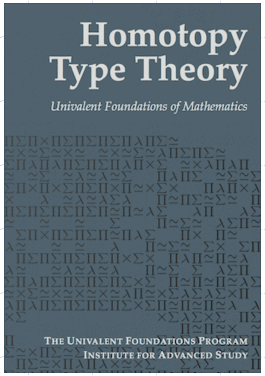
  - But, thus defined, $\exists_i$ is the same as the strong sum Σ!

    We can define $p : \exists_i x{:}A.B(x) \to \Sigma x{:}A.B(x)$ such that $\exists_i$-projections exist:

$$p_1 = \pi_1 \circ p \quad \text{and} \quad p_2 = \pi_2 \circ p$$

  (Bad side effect!)

MLTT_h: Extension of MLTT with H-logic (*)



- H-logic (in Homotopy Type Theory; HoTT book 2013)
  - A proposition is a type with at most one object.
  - Logical operators (examples):
    - $P \supset Q = P \to Q$ and $\forall x{:}A.P = \prod x{:}A.P$
    - $P \vee Q = |P+Q|$ and $\exists x{:}A.P = |\Sigma x{:}A.P|$

    where $|A|$ is <u>propositional truncation</u>, a proper extension.
- MLTT_h = MLTT + h-logic (subsystem of HoTT) [Luo 2019]
  - Proof irrelevance is "built-in" in h-logic (by definition).
  - $\exists$ defined by truncating Σ is a weak sum and can be used to give adequate semantics of donkey sentences as proposed.
  - Note: MLTT_h is a <u>proper extension</u> of MLTT.

Concluding remarks (*)

- Summary
  - Donkey sentences – old topic, but still intriguing.
  - Type theories – with "standard" logics embedded.
  - We have studied this completely proof-theoretically.
- Dynamics in semantics
  - "Dynamic type theory"? (Not a way forward, even IF possible)
    - Cf, Dynamic Predicate Logic that extends FOL.
    - But, DPL is non-standard (eg, non-monotonic …) and proof-theoretically difficult [Veltman 2000] (probably problematic).
  - Formal semantics based on such is too big a price to pay.

Lecture IV. Copredication

This lecture

1. Copredication and dot-types: informal ideas
2. Subtyping (necessary for dot-types approach)
3. Formalisation of dot-types in MTTs
4. Copredication in more sophisticated situations (if time permits)

IV.1. Copredication – examples

- Copredication is a special case of logical polysemy.
  - See (Pustejovsky 1995, Asher 2011), among others.
- Examples
  - John picked up and mastered the book.
  - (*) The lunch was delicious but took forever.
  - The newspaper you are reading is being sued by Mary.
- Consider (*):
  - delicious : Food→**t**; take_forever : Process→**t**
  - Their domains Food/Process ≤ **e** do not share any common objects, but they can both apply to the same noun (lunch) …

How to analyse it formally?

- Very interesting issue
  - Easy to understand, but intriguing (nice research topic)
  - Numerous papers in the literature
- Many approaches, including (just to name a few):
  - Dot-types and related approaches
    - E.g., Pustejovsky 95, Asher 2011, Luo 2010, …
  - Mereological approaches
    - E.g., Gotham 2014, 2017
  - Others
    - E.g., Retoré 2013, Liebesman & Magidor 2023, …

Dot-types

- Dot-types – idea by Pustejovsky (1995)
  - Objects of type A•B have two aspects: being both A and B.
  - Informally, sentences with copredication can now be interpreted.
- How to formalise? – subtyping crucial
  - Formalise dot-types in Montagovian setting?
    - Introducing subsumptive subtyping – similar to Montague+DETs – Lecture II.2.
  - Formalise dot-types in MTTs?
    - Using coercive subtyping – Luo 2010 (SALT20 paper)
- Examples – subtyping is crucial for the correct analysis. We'll try to explain this informally, by examples.

Example in the Montagovian setting

[heavy] : Phy→t

[book] : Phy•Info→t

[heavy book] : Phy•Info→t

[heavy book](x) = [heavy](x) & [book](x)

For this to be well-typed, we need

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

How to formally define A•B?

[No such defn in literature for Montague, but its subtyping aspect is similar to Montague+DETs in Lecture II.2 (omitted here)]

An example in MTT-semantics

"John picked up and mastered the book.

[book] $\leq$ PHY•INFO [Characterising book's copredication]

PHY•INFO $\leq$ PHY and PHY•INFO $\leq$ INFO [by defn of dot-types]

[pick up] : Human $\to$ PHY $\to$ Prop
$\leq$ Human $\to$ PHY•INFO $\to$ Prop
$\leq$ Human $\to$ [book] $\to$ Prop

[master] : Human $\to$ INFO $\to$ Prop
$\leq$ Human $\to$ PHY•INFO $\to$ Prop
$\leq$ Human $\to$ [book] $\to$ Prop

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

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

Dot-types in MTTs

- What is A•B?
  - Inadequate accounts, as summarised by Asher (2008):
    - Intersection type
    - Product type
- Proposal (Luo, 2010)
  - A•B as type of pairs that do not share components
  - Both projections as coercions
- Implementations
  - Coq implementations (Luo 2011, LACL11)
  - Implemented in proof assistant Plastic by Xue (2012, 2013)

Key points of a dot-type

- A dot-type is not an ordinary type
  - E.g., It is not an inductive type in MTTs.
- 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*."
- 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”

In MTT-semantics:

- [heavy] : Phy $\rightarrow$ Prop

  $\leq$ Phy•Info $\rightarrow$ Prop

  $\leq$ Book $\rightarrow$ Prop

- So, the following is well-formed:

  [heavy book] = Σ(Book, [heavy])

- One may compare this with earlier example for “heavy book” in the Montagovian setting.

Copredication in more complicated contexts

- What happens when copredication interacts with …?
  - Interacting with quantification → identity criteria of CNs (Luo 2012)
  - See (Chatzikyriakidis and Luo 2018, Luo 2023)
  - (Left open for now …)

Lecture V. Reasoning, CGs, and Beyond

This lecture

1. NL reasoning in proof assistants
2. Dependent Categorial Grammar
   2.1. Introduction to CGs
   2.2. Substructural type theory: introduction (application to syntactical analysis)

V.1. NL Reasoning in Proof Assistants

- Interactive theorem proving based on MTTs
  - Automatic TP v.s. interactive TP
- An ITP system consists of three parts for:
  (1) contextual defns (2) proof development (3) proof checking

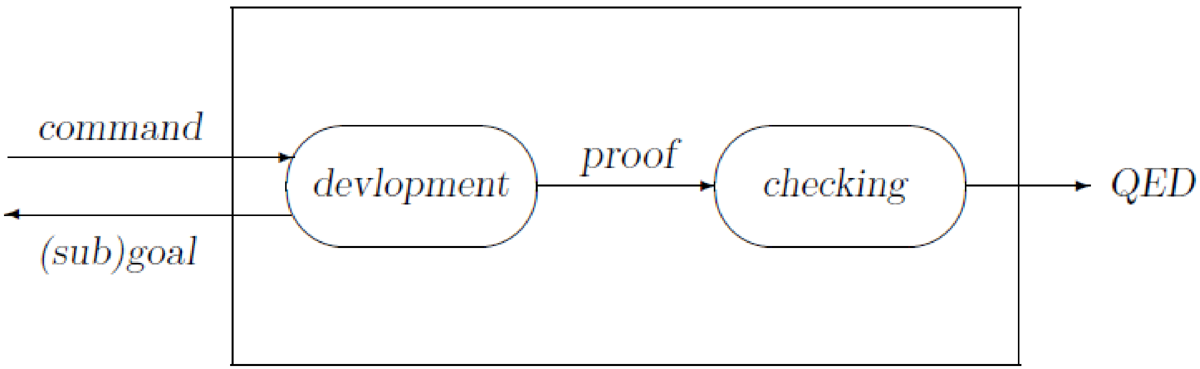


Figure 1: Interactive proof development and proof checking

Simple example (a theorem about primes)

```
(* context: properties about primes *)
Definition div (x y : nat) : Prop := exists z : nat, y = x*z.
Definition prime (n : nat) : Prop := n >= 2 /\ (forall x:nat, (div x n) -> x=1 \/ x=n).

(* Theorem: there are infinitely many primes. *)
Theorem inf_many_primes : not (exists n:nat, forall x:nat, prime x -> x < n).
```

One can then use commands to interact with the system to solve goals by generating “subgoals” and, finally (if successful), to use Qed to finish it.

(Details omitted)

Proof development process

- Enter: `Theorem tautology : forall (A : Prop), A->A.`

```
1 subgoal
______________________________________(1/1)
forall A : Prop, A -> A
```

- Enter command "Intros" (system uses the intro rule backwards, twice):

```
1 subgoal
A : Prop
H : A
______________________________________(1/1)
A
```

- Enter command "Assumption":

```
No more subgoals.
```

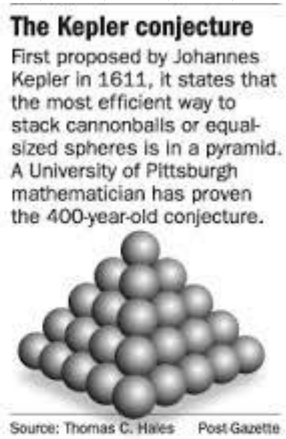
- Enter command "Qed":

```
tautology is defined
```

MTT-based technology and applications (recap)

- Proof technology based on type theories
  - Proof assistants
    - MTT-based: ALF/Agda, Coq, Lean, Lego, NuPRL, Plastic, …
    - HOL-based: Isabelle, 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
    - NL reasoning based on MTT-semantics

      (In Coq: Chatzikyriakidis & Luo 2014/2016/2020; Luo 2023)



NL Reasoning in Coq

- Proof assistant Coq (INRIA, France (Coq 2004))
- Some basic data in MTT-semantics in Coq

```
(*  CNs as types *)
Definition CN := Set.
Parameters Animal Cat Elephant Human Obj: CN.
Parameters John Julie : Human.

(* coercive subtyping relations *)
Axiom ca : Cat -> Animal.          Coercion ca : Cat >-> Animal.
Axiom ea : Elephant -> Animal.     Coercion ea : Elephant >-> Animal.
Axiom ao : Animal -> Obj.          Coercion ao : Animal >-> Obj.
```

Adjectival modification (intersective: black)

```
(* intersective adjective (black) *)
Parameter black : Obj -> Prop.

(* In Coq, "Record" types are Sigma-types *)
Record BCat := mkBC
           { cat :> Cat;
             pBlack : black(cat)
           }.

(* Any black cat is black. *)
Theorem bcat_is_black : forall bc : BCat, black(bc).
intros. apply bc.
Qed.      (* After Qed, bcat_is_black becomes the name of the proof. *)
```

- Further information, including other simple formalisations mentioned in the lectures
  - Adjective modifications (subsective, privative, …)
  - Donkey anaphora (and Most)
  - Dependant event types (e.g., EQP, selection restriction, ...)

can be found in (Luo 2023, Chap 5, esp. Sect 5.3)

V.2. Dependent Categorial Grammar

- Categorial Grammars (or type-logical grammars)
  - An approach to syntactic analysis
  - CGs are based on substructural logics
    - Moortgat: 'Typelogical grammars are substructural logics, designed for reasoning about the composition of form and meaning in natural language.' (Stanford Encyclopedia of Philosophy, 2010)
- What is a substructural logic?
  - In a proof system, there are three kinds of "structural" rules:

    (1) Weakening: adding more assumptions

    (2) Contraction (strengthening): removing repeated/unused assumptions

    (3) Exchange: swapping the order of two assumptions

  In <u>sub</u>structural (resource-sensitive) logics, the above may not be OK.

  In Lambek/CGs, none of them is OK.

Lambek calculus and beyond

- Historical developments:
  - Ajdukiewicz, Bar-Hillel, …
- Lambek calculus (1958)
  - Ordered formulae B/A and A\B
    - John runs – "run applies to a np on the left".
      John : NP and run : NP\S
  - Resource sensitive
    - A context Γ, standing for a sequence of words, represents a sentence if Γ |- S .
    - Words in a sentence cannot be arbitrarily added/removed/swapped
      ➔ context restrictions
      ➔ substructural logics

An example

(*) John runs quickly.

We have, corresponding to (*):

NP, NP\S, (NP\S)\(NP\S) |- S

As the following derivation shows:

	category (type)
John	NP
runs	NP \ S
quickly	(NP \ S) \ (NP \ S)

$$\dfrac{np \qquad \dfrac{(np \setminus S) \quad (np \setminus S) \setminus (np \setminus S)}{(np \setminus S)} \setminus_e}{S} \setminus_e$$

- 1958 → … → 1980s … (CGs further developed)
  - Key: nice account of syntax/semantics interface – close correspondence between CGs and Montague semantics:

    [S] = **t** [NP] = **e** [CN] = **e**→**t** [A\B] = [B/A] = A→B
- Further (more recent) developments includes
  - Linear CGs (Girard's linear logic; 1987)
    - (Oehrle 1994) to initiate, among many others
  - Hybrid CGs (combining ordered/linear types)
    - For example: Kubota & Levine's HTLG (a recent book in 2020), among others

Substructural type theory $\bar{\lambda}_{\Pi}$

- Linear types/terms:

	Π-type	Non-dependent type	Abstraction	Application
Linear	$\overline{\Pi} x{:}A.B$	$A \multimap B$	$\bar{\lambda} x{:}A.b$	$\overline{app}(f, a)$
Ordered (right)	$\Pi^r x{:}A.B$	B/A	$\lambda^r x{:}A.b$	$app^r(f, a)$
Ordered (left)	$\Pi^l x{:}A.B$	$A \setminus B$	$\lambda^l x{:}A.b$	$app^l(a, f)$

Table 1 Three substructural function types in $\bar{\lambda}_{\Pi}$: summary of notations.

- Terms, rather than contexts, represent NL phrases.

Work based on (Luo 2015, Luo & Zhang 2016; see Luo 2023)

Rules for the system without dependent types

Variables

$$\frac{}{x{:}A \vdash x : A}$$

Ordered function types

$$\frac{\Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \lambda^r x{:}A.b : B/A} \qquad \frac{\Gamma \vdash f : B/A \quad \Delta \vdash a : A \quad FV(\Gamma) \cap FV(\Delta) = \emptyset}{\Gamma, \Delta \vdash app^r(f, a) : B}$$

$$\frac{\Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \lambda^l x{:}A.b : A \setminus B} \qquad \frac{\Gamma \vdash f : A \setminus B \quad \Delta \vdash a : A \quad FV(\Gamma) \cap FV(\Delta) = \emptyset}{\Delta, \Gamma \vdash app^l(a, f) : B}$$

Linear function types

$$\frac{\Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \overline{\lambda} x{:}A.b : A \multimap B} \qquad \frac{\Gamma \vdash f : A \multimap B \quad \Delta \vdash a : A \quad FV(\Gamma) \cap FV(\Delta) = \emptyset}{\Gamma, \Delta \vdash \overline{app}(f, a) : B}$$

The lexicon rule

$$\frac{(c, A) \in \textsc{Lex}}{\langle\rangle \vdash c : A}$$

- Notes: if there is no dependent type, types can be defined first/independently:

Definition 1 (types in $\bar{\lambda}_{\rightarrow}$) *Types in $\bar{\lambda}_{\rightarrow}$ are inductively defined as follows:*

1. *The basic categories (such as* S, NP *and* CN*) are types.*
2. *If A and B are types, so are $A \multimap B$, B/A and $A \backslash B$.* $\square$

An example without dep types (c.f., earlier example)

Table 2 Lexicon for (8).

	category (type)
John	NP
runs	NP \ S
quickly	(NP \ S) \ (NP \ S)

(8) John runs quickly.

The lexicon for (8) is given in Table 2. It is straightforward to have:

$$app^l(John, app^l(runs, quickly)) : \mathrm{S}$$

When applying function ϕ to the above term, we have:

$$\phi(app^l(John, app^l(runs, quickly))) = John \circ runs \circ quickly$$

A "counter-example"

(14) (#) a very book

Table 4 Lexicon for (14).

	category (type)
a	NP/CN
very	(CN/CN)/(CN/CN)
book	CN

- Example from (Moot & Retore 2012)
- In Lambek, we'd need a side condition for (/-intro) – context's non-emptiness.
- Otherwise, (14) would be a legitimate phrase:

$$a : \mathrm{NP}/\mathrm{CN},\ \mathit{very} : (\mathrm{CN}/\mathrm{CN})/(\mathrm{CN}/\mathrm{CN}),\ \mathit{book} : \mathrm{CN}\ \vdash\ \mathrm{NP}$$

- In our setting, we have

$$\mathit{app}^r(a, \mathit{app}^r(\mathit{app}^r(\mathit{very}, \lambda^r x{:}\mathrm{CN}.x), \mathit{book}))\ :\ \mathrm{NP}$$

but this term does not represent a legitimate phrase (the λ-term blocks it!)

Rules for substructural Π-types

Formation rules for substructural Π-types

$$\frac{\Gamma,\ x{:}A \vdash B\ type}{\Gamma \vdash \Pi^r x{:}A.B\ type} \qquad \frac{\Gamma,\ x{:}A \vdash B\ type}{\Gamma \vdash \Pi^l x{:}A.B\ type} \qquad \frac{\Gamma,\ x{:}A \vdash B\ type}{\Gamma \vdash \overline{\Pi} x{:}A.B\ type}$$

Ordered Π-types ((app^r) and (app^l) have the side condition $FV(\Gamma) \cap FV(\Delta) = \emptyset$.)

$$(\lambda^r)\quad \frac{\Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \lambda^r x{:}A.b : \Pi^r x{:}A.B} \qquad (app^r)\quad \frac{\Gamma \vdash f : \Pi^r x{:}A.B \quad \Delta \vdash a : A}{\Gamma, \Delta \vdash app^r(f, a) : [a/x]B}$$

$$(\lambda^l)\quad \frac{\Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \lambda^l x{:}A.b : \Pi^l x{:}A.B} \qquad (app^l)\quad \frac{\Gamma \vdash f : \Pi^l x{:}A.B \quad \Delta \vdash a : A}{\Delta, \Gamma \vdash app^l(a, f) : [a/x]B}$$

Linear Π-types (($\overline{app}$) has the side condition $FV(\Gamma) \cap FV(\Delta) = \emptyset$.)

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

An example with dependent types

(23) Most students study hard.

Table 5 Lexicon for (23)

	category (type)
most	$\Pi^r X{:}\mathrm{CN}.\ \mathrm{S}/(X \setminus \mathrm{S})$
students	CN
study	NP \ S
hard	(NP \ S) \ (NP \ S)

- In our system, we have

$$app^r(app^r(most, students), app^l(study, hard)) : \mathrm{S}$$

$$\phi(app^r(app^r(most, students), app^l(study, hard)))$$
$$= \mathrm{most} \circ students \circ study \circ hard$$

So, (23) is a legitimate sentence.

Linearity

Variables

$$(Var) \qquad \frac{\Gamma, x{:}A \ valid}{\Gamma, x{:}A \vdash x{:}A} \quad (\forall y \in FV(\Gamma).\ x \sim_{\Gamma, x:A} y)$$

where, in the side condition of (Var), for any $\Delta = x_1{:}A_1, ..., x_n{:}A_n$, the dependency relation $\sim_\Delta$ is defined as: (1) if $y \in FV(A_i)$, then $x_i \sim_\Delta y$; (2) if $x \sim_\Delta y$ and $y \sim_\Delta z$, then $x \sim_\Delta z$.

- Theorem (linearity)

(Weak linearity in $\bar{\lambda}_\Pi$) In $\bar{\lambda}_\Pi$, every contextual variable occurs free essentially for exactly once in a well-typed term. In symbols, if $\Gamma \vdash a : A$ with $\Gamma = x_1{:}A_1, ..., x_n{:}A_n$, then each x_i occurs free essentially in a for exactly once (i.e., $x_i \in E_\Gamma(a)$ for exactly once $(i = 1, ..., n)$).

References (cited in the lecture slides)

- 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. Carlström. Interpreting descriptions in intensional type theory. Journal of Symbolic Logic 70(2). 2005.
- R. Casati and A. Varzi. Events: An Annotated Bibliography. 1997. [235 pages]
- L. Champollion. The interaction of compositional semantics and event semantics. Linguistics and Philosophy, 38(1):31–66, 2015.
- S. Chatzikyriakidis and Z. Luo. Natural Language Inference in Coq. Journal of Logic, Language and Information, 23(4). 2014.
- 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.
- S. Chatzikyriakidis and Z. Luo. Identity criteria of common nouns and dot-types for copredication. Oslo Studies in Language, 10(2). 2018.

References (cited in the lecture slides)

- S. Chatzikyriakidis and Z. Luo. *Formal Semantics in Modern Type Theories*. Wiley/ISTE. 2020.
- G. Chierchia. Anaphora and dynamic logic. ITLI Publication Series for Logic, Semantics and Philosophy of Language, LP-1990-07. 1990.
- A. Church. A formulation of the simple theory of types. J. Symbolic Logic, 5(1). 1940.
- R. Clark. Concerning the logic of predicate modifiers. Noûs, 4(4), 1970.
- Coq Development Team. The Coq Proof Assistant Reference Manual (Version 8.0). INRIA, 2004.
- D. Davidson. The logical form of action sentences. In: S. Rothstein (ed.). The Logic of Decision and Action. University of Pittsburgh Press. 1967.
- P. de Groote. Towards abstract categorial grammars [C]. Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, pages 252–259, 2001.
- P. de Groote and Y. Winter. A type-logical account of quantification in event semantics. JSAI International Symposium on Artificial Intelligence, pages 53–65. 2014.
- G. Evans. Pronouns, quantifiers and relative clauses. Canadian Journal of Philosophy 7(3). 1977.
- G. Evans. Pronouns. Linguistic Inquiry, 11(2). 1980.

References (cited in the lecture slides)

- P. Geach. Reference and Generality: An Examination of Some Medieval and Modern Theories. Cornell University Press, 1962.
- J.-Y. Girard. Linear logic. Theoretical computer science, 50(1). 1987.
- M. Gotham. Copredication, quantification and individuation. PhD thesis, University College London. 2014.
- M. Gotham, M. Composing criteria of individuation in copredication. J of Semantics, 34(2). 2017.
- J. Groenendijk and M. Stokhof. Dynamic predicate logic. Ling & Phil, 14(1). 1991.
- I. Heim. The Semantics of Definite and Indefinite Noun Phrases. PhD thesis, University of Massachusetts. 1982.
- HoTT. Homotopy Type Theory: Univalent Foundations of Mathematics. The Univalent Foundations Program, Institute for Advanced Study, 2013.
- H. Kamp. Two theories about adjectives. In E. Keenan, editor, Formal Semantics of Natural Language. Cambridge Univ Press, 1975.
- H. Kamp. Events, instants and temporal reference. In R. Bäuerle, U. Egli, and A. von Stechow (eds.), Semantics from Different Points of View. Springer, 1979.
- H. Kamp. A theory of truth and semantic representation. In J. Groenendijk et al (eds.) Formal Methods in the Study of Language, pages 189–222, 1981.
- Y. Kubota and R. Levine. Type-Logical Syntax. MIT, 2020.

References (cited in the lecture slides)

- J. Lambek. The mathematics of sentence structure [J]. The American Mathematical Monthly, 65(3). 1958.
- D. Liebesman and O. Magidor. Copredication and Meaning Transfer. J. of Semantics, 40. 2023.
- Z. Luo. *Computation and Reasoning: A Type Theory for Computer Science.* Oxford University Press, 1994.
- Z. Luo. Coercive subtyping in type theory. Computer Science Logic (CSL96), LNCS 1258. 1997.
- 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. 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: Is It Model-theoretic, Proof-theoretic, or Both? Invited talk at Logical Aspects of Computational Linguistics 2014 (LACL 2014), Toulouse. LNCS 8535, pp177-188. 2014.
- Z. Luo. A Lambek Calculus with Dependent Types. TYPES 2015. Tallinn, May 2015.
- Z. Luo. Formal Semantics in Modern Type Theories (and Event Semantics in MTT-Framework). Invited talk at LACompLing 2018. Stockholm, 2018.

References (cited in the lecture slides)

- Z. Luo. Proof irrelevance in type-theoretical semantics. Logic and Algorithms in Computational Linguistics 2018 (LACompLing2018), Studies in Computational Intelligence , Springer, pages 1–15. 2019.
- Z. Luo. On Type-Theoretical Semantics of Donkey Anaphora. Logical Aspects of Computational Linguistics (LACL21). Montpellier (online), 2021. (An earlier version of the paper, titled "Donkey Anaphora: Type-Theoretic Semantics with Both Strong and Weak Sums", was read in the ESSLLI 2021 Workshop CSTFRS21.)
- Z. Luo. *Modern Type Theories: Their Development & Applications*. Tsinghua University Press. 2023. (In Chinese)
- 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 and S. Soloviev. Dependent event types. Manuscript. 2020.
- 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. An intuitionistic theory of types: Predicative part. In H. Rose and J. Shepherdson (eds). Logic Colloquium'73. 1975.
- P. Martin-Löf. *Intuitionistic Type Theory*. 1984.

References (cited in the lecture slides)

- U. Mönnich. Untersuchungen zu einer konstruktiven Semantik fur ein Fragment des Englischen. Habilitation, University of Tübingen, 1985.
- R. Montague. English as a formal language. In B. Visentini et al (eds.), Linguaggi nella societa e nella tecnica. Edizioni di Communita, pages 189-223, 1970.
- R. Montague. Formal philosophy. Yale Univ Press, 1974. (Collection edited by R. Thomason)
- M. Moortgat. Typelogical grammar. Stanford Encyclopedia of Philosophy, 2010.
- R. Moot and C. Retore. *The Logic of Categorial Grammar*. LNCS 6850. Springer, 2012.
- R. Oehrle. Term-labelled categorial type systems. Linguistics and Philosophy 17(6). 1994.
- T. Parsons. Some problems concerning the logic of grammatical modifiers. Synthese, 21(3/4), 1970.
- T. Parsons. *Events in the Semantics of English*. MIT, 1990.
- A. Prior. *Past, Present and Future*. OUP. 1967.
- J. Pustejovsky. The Generative Lexicon. MIT. 1995.

References (cited in the lecture slides)

- A. Ranta. *Type-Theoretical Grammar*. Oxford University Press, 1994.
- C. Retoré. The Montagovian generative lexicon λTyn: a type theoretical framework for natural language semantics. Proc of TYPES2013, pages 202–229, 2013.
- B. Russell. Introduction to Mathematical Philosophy. Allen and Unwin, 1919.
- G. Sundholm. Proof theory and meaning. In D. Gabbay and F. Guenthner (eds). Handbook of Philosophical Logic III: Alternatives to Classical Logic. 1986.
- G. Sundholm. Constructive Generalized Quantifiers. Synthese 79(1). 1989.
- R. Tanaka. Generalized quantifiers in dependent type semantics. Talk given at Ohio State University, 2015.
- J. van Benthem. *The Logic of Time*. 2nd edition, Kluwer. 1991.
- F. Veltman. Proof systems for Dynamic Predicate Logic. Manuscript, 2000.
- Y. Winter and J. Zwarts. Event semantics and abstract categorial grammar. In Conference on Mathematics of Language, pages 174-191. Springer, 2011.
- T. Xue. Theory and Implementation of Coercive Subtyping. PhD thesis, Royal Holloway, University of London, 2013.
- T. Xue and Z. Luo. Dot-types and their implementation. LACL'12, LNCS 7351. 2012.