A Lambek Calculus with Dependent Types

Zhaohui Luo
Dept of Computer Science
Royal Holloway, Univ of London

This talk

- Background and motivation
  - Categorial grammars and Montague semantics
  - NL semantics in Modern TTs – syntactic counterpart?
- Lambek dependent types
  - Dependent types in resource sensitive TTs – previous work
  - Dependent types in Lambek calculi
- Future work

Categorial Grammars and Montague Semantics

- Montague semantics (MG in early 70's)
  - MG is based on the simple TT (Church 1940)
  - Dominating semantic framework in the last four decades
- Categorial grammars
  - Early work (Ajdukiewicz 1935, Bar-Hillel 1953)
  - CG as logic (Lambek 1958)
  - "Lambek = linear – exchange"
    - "linear = intuitionistic – (weakening + contraction)"
- Close correspondence between syntax and semantics
  - Lambek CG ------ Montague semantics
  - Implementations (eg, the Grail system by Moot)

The Lambek calculus

- Presented as an ND type system
  - c.f. (Polakow-Pfenning 1999)
- Rules for / and \ (forget Γ for the moment)

$$(/\text{-F})\quad \frac{\Gamma;\ *\vdash A\ type \quad \Gamma;\ *\vdash B\ type}{\Gamma;\ *\vdash B/A\ type} \qquad (/\text{-I})\quad \frac{\Gamma;\ (\Delta, x{:}A)\vdash b : B \quad \Gamma;\ *\vdash B/A\ type}{\Gamma;\ \Delta\vdash /x{:}A.b : B/A}$$

$$(/\text{-E})\quad \frac{\Gamma;\ \Delta_1\vdash f : B/A \quad \Gamma;\ \Delta_2\vdash a : A}{\Gamma;\ (\Delta_1,\Delta_2)\vdash f\ a : B} \qquad (/\text{-C})\quad \frac{\Gamma;\ (\Delta_1, x{:}A)\vdash b : B \quad \Gamma;\ \Delta_2\vdash a : A}{\Gamma;\ (\Delta_1,\Delta_2)\vdash (/x{:}A.b)\ a = [a/x]b : B}$$

$$(\backslash\text{-F})\quad \frac{\Gamma;\ *\vdash A\ type \quad \Gamma;\ *\vdash B\ type}{\Gamma;\ *\vdash A\setminus B\ type} \qquad (\backslash\text{-I})\quad \frac{\Gamma;\ (x{:}A, \Delta)\vdash b : B \quad \Gamma;\ *\vdash A\setminus B\ type}{\Gamma;\ \Delta\vdash \backslash x{:}A.b : A\setminus B}$$

$$(\backslash\text{-E})\quad \frac{\Gamma;\ \Delta_1\vdash f : A\setminus B \quad \Gamma;\ \Delta_2\vdash a : A}{\Gamma;\ (\Delta_2,\Delta_1)\vdash a\ f : B} \qquad (\backslash\text{-C})\quad \frac{\Gamma;\ (x{:}A, \Delta_1)\vdash b : B \quad \Gamma;\ \Delta_2\vdash a : A}{\Gamma;\ (\Delta_2,\Delta_1)\vdash a\ (\backslash x{:}A.b) = [a/x]b : B}$$

Lambek CG and Montague semantics: example

John works hard.

	Types in Lambek CG	Types in Montague sem
John	e	e
works	e \ s	e → t
hard	(e \ s) \ (e \ s)	(e → t) → (e → t)

Note: Phrases as terms (c.f., contextual strings)

- Let Γ be John : e, works : e\s, hard : (e\s)\(e\s)
- $\Gamma \vdash$ John (works hard) : s

Semantics in Modern TTs (MTT-semantics)

- Examples of MTTs
  - Martin-Löf's TT, Coq's CIC_p, UTT, … …
- MTT-semantics of NLs
  - Early work (Ranta 1994)
  - Recent development into a full-blown alternative to Montague semantics, with various advantages
  - E.g., CNs as types and subtyping
- MTT-semantics: both model- and proof-theoretic
  - Model-theoretic – rich type structure with wide coverage
  - Proof-theoretic – inferential understanding and practical reasoning (eg, in Coq)

- Question:

  CG ---------- Montague semantics

  ??? ---------- MTT-semantics

  Dependent types in resource sensitive calculi?
- Motivation (among others)

  Uniform basis for NL analysis
  - automated syntactical analysis
  - logical reasoning in proof assistants with MTT-semantics

A Lambek calculus with dependent types

- Extension of the Lambek calculus (recall \, / and •)
- Add directed dependent types
  - Directed dependent product types Π^r/Π^l
  - Directed dependent sum types $\Sigma^\sim/\Sigma^o$
- Add intuitionistic Π and Σ
  - C.f. (de Groote et al. 2007)
  - Arguments of Π in syntactic analysis are usually "omitted".
- Add universes S (of sentences) and CN (of common nouns).

Resource sensitive dependent types

- Previous work on linear TTs
  - Linear LF (Pfenning et al. 2002)
  - Recent work (Vákár 2015, Krishnaswami et al. 2015)
- Introducing dependent types into Lambek calculi
  - Contexts with two parts:

    $\Gamma; \Delta$

    intuitionistic context Γ and Lambek context Δ.
  - Judgements:
    - Types : $\Gamma; * \vdash A$ type $\Gamma; * \vdash A = B$
    - Objects: $\Gamma; \Delta \vdash a : A$ $\Gamma; \Delta \vdash a = b : A$

    Note: Types are only dependent on intuitionistic variables in Γ.

- Equality typing (conversion rule)

$$\frac{\Gamma; \Delta \vdash a : A \quad \Gamma; \ * \vdash A = B}{\Gamma; \Delta \vdash a : B}$$

- Variables

$$\frac{\Gamma, x{:}A, \Gamma'; \ * \ \mathit{valid}}{\Gamma, x{:}A, \Gamma'; \ * \vdash x : A} \qquad \frac{\Gamma; \ y{:}A \ \mathit{valid}}{\Gamma; \ y{:}A \vdash y : A}$$

Directed dependent product types Π^r/Π^l

$$(\Pi^r\text{-F})\ \frac{\Gamma;\ *\vdash A\ type \quad \Gamma, x{:}A;\ *\vdash B\ type}{\Gamma;\ *\vdash \Pi^r x{:}A.B\ type} \qquad (\Pi^r\text{-I})\ \frac{\Gamma, x{:}A;\ \Delta\vdash b : B \quad \Gamma;\ *\vdash \Pi^r x{:}A.B\ type}{\Gamma;\ \Delta\vdash \lambda^r x{:}A.b : \Pi^r x{:}A.B}$$

$$(\Pi^r\text{-E})\ \frac{\Gamma;\ \Delta\vdash f : \Pi^r x{:}A.B \quad \Gamma;\ *\vdash a : A}{\Gamma;\ \Delta\vdash app^r(f, a) : [a/x]B} \qquad (\Pi^r\text{-C})\ \frac{\Gamma, x{:}A;\ \Delta\vdash b : B \quad \Gamma;\ *\vdash a : A \quad \Gamma;\ *\vdash \Pi^r x{:}A.B\ type}{\Gamma;\ \Delta\vdash app^r(\lambda^r x{:}A.b,\ a) = [a/x]b : [a/x]B}$$

$$(\Pi^l\text{-F})\ \frac{\Gamma;\ *\vdash A\ type \quad \Gamma, x{:}A;\ *\vdash B\ type}{\Gamma;\ *\vdash \Pi^l x{:}A.B\ type} \qquad (\Pi^l\text{-I})\ \frac{\Gamma, x{:}A;\ \Delta\vdash b : B \quad \Gamma;\ *\vdash \Pi^l x{:}A.B\ type}{\Gamma;\ \Delta\vdash \lambda^l x{:}A.b : \Pi^l x{:}A.B}$$

$$(\Pi^l\text{-E})\ \frac{\Gamma;\ \Delta\vdash f : \Pi^l x{:}A.B \quad \Gamma;\ *\vdash a : A}{\Gamma;\ \Delta\vdash app^l(a,\ f) : [a/x]B} \qquad (\Pi^l\text{-C})\ \frac{\Gamma, x{:}A;\ \Delta\vdash b : B \quad \Gamma;\ *\vdash a : A \quad \Gamma;\ *\vdash \Pi^l x{:}A.B\ type}{\Gamma;\ \Delta\vdash app^l(a,\ \lambda^l x{:}A.b) = [a/x]b : [a/x]B}$$

Directed dependent sum types $\Sigma^{\sim}/\Sigma^{o}$

$$(\Sigma^{\sim}\text{-F}) \quad \frac{\Gamma;\ *\vdash A\ type \quad \Gamma, x{:}A;\ *\vdash B\ type}{\Gamma;\ *\vdash \Sigma^{\sim} x{:}A.B\ type}$$

$$(\Sigma^{\sim}\text{-I}) \quad \frac{\Gamma; *\vdash a : A \quad \Gamma; \Delta \vdash b : [a/x]B}{\Gamma; \Delta \vdash pair^{\sim}(b, a) : \Sigma^{\sim} x{:}A.B}$$

$$(\Sigma^{\sim}\text{-E}) \quad \frac{\Gamma; \Delta \vdash p : \Sigma^{\sim} x{:}A.B \quad \Gamma, x{:}A;\ \Delta', y{:}B \vdash e : C \quad \Gamma;\ *\vdash C\ type}{\Gamma;\ (\Delta, \Delta') \vdash \mathbf{let}\ pair^{\sim}(y, x) = p\ \mathbf{in}\ e\ :\ C} \quad (FV(\Delta) \cap FV(\Delta') = \emptyset)$$

$$(\Sigma^{\sim}\text{-C}) \quad \frac{\Gamma; *\vdash a : A \quad \Gamma; \Delta \vdash b : [a/x]B \quad \Gamma, x{:}A;\ \Delta', y{:}B \vdash e : C \quad \Gamma;\ *\vdash C\ type}{\Gamma;\ (\Delta, \Delta') \vdash \mathbf{let}\ pair^{\sim}(y, x) = pair^{\sim}(b, a)\ \mathbf{in}\ e\ =\ [a/x, b/y]e\ :\ C} \quad (FV(\Delta) \cap FV(\Delta') = \emptyset)$$

(Note: Rules for Σ^{o} are symmetric and omitted.)

Universes S and CN

- Universe S of sentences

$$\frac{}{*; * \vdash S\ type} \qquad \frac{\Gamma;\ * \vdash A : S}{\Gamma;\ * \vdash T_S(A)\ type}$$

- Universe CN of common nouns

$$\frac{}{*; * \vdash CN\ type} \qquad \frac{\Gamma;\ * \vdash A : CN}{\Gamma;\ * \vdash T_{CN}(A)\ type}$$

Note: CN is closed under $\Sigma^{\sim}/\Sigma^{o}$

Examples of Lambek CG with dependent types

(1) John works hard.

	Lambek CG with dependent types	**MTT semantics**
John	Man (≤ Human)	Man (≤ Human)
works	Human \ S	Human → Prop
hard	ΠA:CN.(A\S)\(A\S)	ΠA:CN.(A→Prop)→(A→Prop)

John (works hard) : S

[John works hard] : Prop

Examples (2)

(2) Every student works.

	Lambek CG with dependent types	**MTT semantics**
every	Π^r A:CN. S / (A \ S)	Π A:CN. (A→Prop)→Prop
student	CN (Student ≤ Human)	CN (Student ≤ Human)
works	Human \ S	Human → Prop

app^r(every, student) works : S

[Every student works] : Prop

Examples (3)

(3) diligent student

	Lambek CG with dependent types	**MTT semantics**
diligent	Human \ S	Human → Prop
student	CN (Student ≤ Human)	CN (Student ≤ Human)
diligent student	= $\tilde{\Sigma}$(student,diligent) : CN	= Σ(student,diligent) : CN

Note: $\tilde{\Sigma}$(student,diligent) abbreviates $\tilde{\Sigma}$x:student.(x diligent).

diligent student = $\tilde{\Sigma}$(student,diligent) : CN
[diligent student] = Σ(student,diligent) : CN

Future work

- Further development
  - CG based on Lambek dependent types
  - Meta-theory (expected OK)
  - Implementation (from syntactical analysis to semantics reasoning in proof assistants)
- More general studies on dependent types in resource sensitive frameworks
  - Types dependent on Lambek/linear variables?
  - What about universes – Vakar's question?