A Lambek Calculus with Dependent Types*

Zhaohui Luo†

Royal Holloway, Univ of London
zhaohui.luo@hotmail.co.uk

In this note, we discuss how to introduce dependent types into the Lambek calculus [7] (or, in general, an ordered calculus [17] and extensions such as those found in the studies of categorial grammars [12, 13]). One of the motivations to introduce dependent types in syntactical analysis is to facilitate a closer correspondence between syntax and semantics, especially when modern type theories are used in formal semantics [14, 9, 10, 2]. We then hope to establish a uniform basis for NL analysis: from automated syntactical analysis to logical reasoning in proof assistants based on type theories and formal semantics.

Dependent types have been used in various contexts in computational linguistics (see, for example, [14, 3, 11] among others). In [15] that formalises the Lambek calculus in (a proof assistant that implements) Martin-Löf's type theory, Ranta discussed the idea of introducing type dependency into directed types and gave an inspiring example to represent directed types of quantifiers, although the paper did not study a formal treatment of such an extension. De Groote et al [4] studied how to extend the underlying type system for ACGs by (intuitionistic) dependent product types so that type families can be used to represent syntactic categories indexed by linguistic features.

Combining resource sensitive types (such as those in linear logic [5]) with dependent types has been an interesting but difficult research topic and many researchers have worked on this, mainly with motivations to apply such calculi to programming and verification problems in computer science (see early work such as [1] and recent developments such as [6, 16]). However, how such a combination should be done is still widely open, on the one hand, and very much depends on the motivations in their applications, on the other.

We present a Lambek calculus with dependent types. Besides the type constructors in the Lambek calculus [7], we shall introduce directed types for dependent products (Π^r and Π^l) and dependent sums ($\Sigma^\sim$ and Σ°). Also considered is how to introduce type universes into the calculus. These are some of the type constructors in a modern type theory found useful in formal semantics and, therefore, their introduction at the syntactic level helps to facilitate a closer syntax-semantics tie as mentioned above. A focus of the current note is to formulate the rules for these types in ordered contexts so that their meanings are correctly captured.

1 The Lambek Calculus

Introducing dependent types into a Lambek calculus, we consider a calculus with contexts having two parts:

$$\Gamma;\ \Delta$$

where Γ is an intuitionistic context (whose variables can be used for any times in a term) and Δ is a Lambek context (or ordered context). Types may only depend on ordinary variables in Γ,

*Extended abstract for TYPES 2015, Tallinn, 2015.

†Partially supported by grants from Royal Academy of Engineering and CAS/SAFEA International Partnership Program.

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

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

Figure 1: Rules for directed Lambek types B/A.

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

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

Figure 2: Directed Π^r-types.

but not on Lambek variables in Δ.[1] Since we are going to introduce dependent types in which objects may occur, we need the following equality typing rule which says that computationally equal types have the same objects:

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

Contexts of the above form obey the following validity rules where, in the last two rules, $x\notin FV(\Gamma,\Delta)$:

$$\frac{}{*;*\ valid} \qquad \frac{\Gamma;\Delta\ valid \quad \Gamma;\ *\vdash A\ type}{(\Gamma, x{:}A);\ \Delta\ valid} \qquad \frac{\Gamma;\Delta\ valid \quad \Gamma;\ *\vdash A\ type}{\Gamma;\ (\Delta, x{:}A)\ valid}$$

For variables, we have

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

We can now present the directed types in the Lambek calculus. The rules for the directed types B/A are given in Figure 1. The rules for $A\setminus B$ are symmetric with term constructors such as $\setminus x{:}A.b$ and are omitted (so are the rules for ordered conjunctions).

2 Dependent Lambek Types

Directed Dependent Products. Dependent product types (Π-types) are split into directed dependent products (Π^r and Π^l). The rules for Π^r-types are given in Figure 2. The rules for Π^l-types, omitted here, are symmetric with term constructors $\lambda^l x{:}A.b$ and $app^l(a,f)$.

[1] This is a design choice of the current notes; it follows all the existing work (so far) on introducing dependent types into resource sensitive calculi.

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

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

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

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

Figure 3: Rules for $\Sigma^{\sim}$-types.

Type Universes. In type theory, a type universe is a type whose objects are (names of) types. For instance, common nouns can be considered to correspond to types (rather than predicates), as in formal semantics in modern type theories [14, 8]. In a similar fashion, we may introduce a universe CN of common nouns:

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

where T_{CN} maps any common noun to a type (we often omit T_{CN} and just write A for $T_{CN}(A)$). (CN is closed under several type constructors including the directed dependent sum types below.)

Example 2.1. *Here is a simple example in syntactic analysis (as in categorial grammar). Consider the following sentence (1):*

(1) Every student works.

The words in the sentence can be given the following types:

(2) $every : \Pi^r X{:}CN.\ S/(X\setminus S)$

(3) $student : CN$

(4) $works : human\setminus S$

where S is the type of sentences and student is a subtype of human (and, hence by contravariance, $human\setminus S$ is a subtype of $student\setminus S$). It is then straightforward to derive that

$$app^r(every, student)\ works\ :\ S$$

In other words, (1) is a sentence.

Directed Dependent Sums Dependent sum types (Σ-types) are split into reverse dependent sums ($\Sigma^{\sim}$) and concatenation dependent sums (Σ°). The rules for $\Sigma^{\sim}$-types are given in Figure 3, while the rules for Σ°-types are symmetric and omitted. We remark that the universe CN is closed under $\Sigma^{\sim}$ and Σ°. For example, directed dependent sum types may be used to analyse modified common nouns when the modifying adjectives are intersective or subsective. To illustrate this with an example, assuming that $B : A\setminus S$, let's use $\Sigma^{\sim}(A,B)$ to abbreviate $\Sigma^{\sim}x{:}A.(x\ B)$ and $\Sigma^{\circ}(A,B)$ to abbreviate $\Sigma^{\circ}x{:}A.(x\ B)$. Now, with $diligent : human\setminus S$, we can use $\Sigma^{\sim}(student, diligent)$ to describe the modified common noun *diligent student*, and $\Sigma^{\circ}(student, diligent)$ to analyse *student who is diligent*.

References

[1] I. Cervesato and F. Pfenning. A linear logical framework. *Information and Computation*, 179, 2002.

[2] S. Chatzikyriakidis and Z. Luo. Natural language reasoning in Coq. *J. of Logic, Language and Information*, 23(4), 2014.

[3] P. de Groote and S. Maarek. Type-theoretic extensions of abstract categorial grammars. *Proc. of the Workshop on New Directions in Type-theoretic Grammars. ESSLLI 2007*, 2007.

[4] P. de Groote, S. Maarek, and R. Yoshinaka. On two extensions of abstract categorial grammars. *LPAR 2007, LNAI 4790*, 2007.

[5] J.-Y. Girard. Linear logic. *Theoret. Comput. Sci.*, 50, 1987.

[6] N. Krishnaswami, P. Pradic, and N. Benton. Integrating dependent and linear types. *POPL 2015*, 2015.

[7] J. Lambek. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3), 1958.

[8] Z. Luo. Common nouns as types. In D. Bechet and A. Dikovsky, editors, *Logical Aspects of Computational Linguistics (LACL'2012). LNCS 7351*, 2012.

[9] Z. Luo. Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy*, 35(6):491–513, 2012.

[10] 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*, pages 177–188, 2014.

[11] S. Martin and C. Pollard. A dynamic categorial grammar. *Formal Grammar 2014*, 2014.

[12] M. Moortgat. Categorial type logic. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 3–43. Elsevier, 1997.

[13] G. Morrill. *Categorial Grammar: Logical Syntax, Semantics, and Processing*. CUP, 2011.

[14] A. Ranta. *Type-Theoretical Grammar*. Oxford University Press, 1994.

[15] A. Ranta. Syntactic calculus with dependent types. *J of Logic, Language and Information*, 7, 1998.

[16] M. Vákár. A categorical semantics for linear logical frameworks. *FoSSaCS 2015*, 2015.

[17] D. Walker. Substructural type systems. In B. Pierce, editor, *Advanced Topics in Types and Programming Languages*, pages 3–43. MIT, 2005.