Lambek Grammars as Combinatory Categorial Grammars

GERHARD JÄGER, *Zentrum für Allgemeine Sprachwissenschaft (ZAS), Jägerstr. 10/11, 10117 Berlin, Germany.*
E-mail: jaeger@zas.gwz-berlin.de

Abstract

We propose a combinatory reformulation of the product free version of the categorial calculus **LL**, i.e. the associative Lambek calculus that admits empty premises. We prove equivalence of the combinatory with the standard Natural Deduction presentation of **LL**. The result offers a new perspective on the relation between the type logical and the combinatory branch of the Categorial Grammar research program.

Keywords: Categorial Grammars, Lambek Calculus, CCG

1 The Lambek calculus

1.1 Sequent presentation

In his seminal paper [7], Joachim Lambek introduced the (associative) Lambek calculus, a type logical extension of Bar Hillel's [3] Basic Categorial Grammars. Lambek gives two equivalent proof theories for his calculus, an axiomatic one and a Gentzen style sequent calculus. We will restrict attention to the product-free fragment of this calculus. There the categorial slashes "/" and "\" are the only logical constants. The sequent calculus consists of the rules of use and of proof for them, together with the identity axiom scheme for arbitrary types. Being a subsystem of positive implicational intuitionistic logic, the history of a proof in the Lambek calculus can be recorded in λ-terms, using the Curry-Howard correspondence.

Definition 1.1 (**LL**: Sequent presentation)

$$\frac{}{x : A \to x : A}\, id$$

$$\frac{X \to M : A \qquad Y, x : B, Z \to N : C}{Y, y : B/A, X, Z \to N[x \leftarrow (yM)] : C}\, /L \qquad \frac{X, x : A \to M : B}{X \to \lambda x.M : B/A}\, /R$$

$$\frac{X \to M : A \qquad Y, x : B, Z \to N : C}{Y, X, y : A \backslash B, Z \to N[x \leftarrow (yM)] : C}\, \backslash L \qquad \frac{x : A, X \to M : B}{X \to \lambda x.M : A \backslash B}\, \backslash R$$

Lambek originally added the additional constraint that the premises of each sequent be non-empty. We omit this requirement as an essentially superfluous complication, thus arriving at the product free version of the calculus **LL**.

In [7] it is proved that the Cut rule is admissible in the Lambek calculus. The proof carries over to the version of **LL** considered here without complications.

1.2 Natural Deduction

Alternatively, **LL** can be presented in a Natural Deduction format, i.e. the rules of use for the implicational slashes can be replaced by explicit Modus Ponens rules. Here Curry-Howard labels mirror the proof history directly.

Definition 1.2 (**LL**: Natural Deduction)

$$\frac{}{x : A \to x : A}\,id$$

$$\frac{X \to M : B/A \qquad Y \to N : A}{X, Y \to (MN) : B}\,/E \qquad \frac{X, x : A \to M : B}{X \to \lambda x.M : B/A}\,/I$$

$$\frac{X \to M : A \qquad Y \to N : A \setminus B}{X, Y \to (NM) : B}\,\setminus E \qquad \frac{x : A, X \to M : B}{X \to \lambda x.M : A \setminus B}\,\setminus I$$

It is easy to show that the two inference formats are equivalent. Even stronger, sequent derivations and ND derivations generate the same set of proof terms.

2 Combinatory Categorial systems

The Combinatory branch of Categorial grammar was initiated by the work of [1]. As in Basic Categorial Grammars, CCG (Combinatory Categorial Grammar) makes use of the identity axiom and the slash elimination rules [/E] and [\E], but it does without the slash introduction rules [/I] and [\I]. The Basic Categorial core is extended by other schemes of inference though. Most works in the CCG tradition assume some—possibly restricted—version of Type Raising **T** and (generalized) Function Compositions **B** (for ease of comparison, we employ a sequent style notation here). As in the systems discussed above, the history of a derivation can be recorded in a label in CCG; labels are terms in the closure of the set of typed variables under the set of combinators.

Definition 2.1 (Type Raising)

$$\frac{X \to M : A}{X \to \mathbf{T}_{>}(M) : B/(A \setminus B)}\,\mathbf{T}_{>} \qquad \frac{X \to M : A}{X \to \mathbf{T}_{<}(M) : (B/A) \setminus B}\,\mathbf{T}_{<}$$

Definition 2.2 (Function Composition)

$$\frac{X \to M : A/B \qquad Y \to N : C_{n_l} \setminus \cdots \setminus B/D_1/\cdots/D_{n_r}}{XY \to \mathbf{B}_{>}(M, N) : C_{n_l} \setminus \cdots \setminus A/D_1/\cdots/D_{n_r}}\,\mathbf{B}_{>}$$

$$\frac{X \to N : C_{n_l} \setminus \cdots \setminus B/D_1/\cdots/D_{n_r} \qquad M : B \setminus A \to Y}{XY \to \mathbf{B}_<(N,M) : C_{n_l} \setminus \cdots \setminus A/D_1/\cdots/D_{n_r}}\ \mathbf{B}_<$$

It is easy to see that the well-known schemes of ordinary function composition

$$A/B, B/C \to A/C$$

and crossed composition

$$A/B, C \setminus B \to C \setminus A$$

(and their mirror images) are special cases of the more general operations given above.

Next to these combinatory inference schemes, CCGs might contain empty categories, i.e. axioms of the form

$$\frac{}{\to M : A}$$

for some term M and type A. They may be considered as 0-place combinators.

3 Combinatory presentation of LL

Even though both branches of modern Categorial grammar are based on the same basic intuitions, the formal properties of the the systems differ considerably. This is most obvious when we consider issues of weak generative capacity: Since [9] it is known that Lambek grammar recognize exactly the context free languages (and the proof carries over directly to **LL**), while the generative capacity of CCGs is mildly context sensitive (cf. [5]). So the two approaches seem to be irreconcilable.

Recent developments in type logical grammar have changed this rigid picture. The use of multi-modality (see for instance [8]) increases the generative capacity of type logical grammars in principle to Turing completeness (cf. [4]), so there is no principal obstacle to simulate CCGs in the type logical framework. [6] contains some general results based on this idea.

In this section, we will pursue the opposite strategy. It will be shown that under a slight generalization of the notion of function composition as it is used in the CCG tradition, **LL** can equivalently be reformulated as a combinatory system.

First some terminology: we will call the first operand of forward function composition $\mathbf{B}_>$ and the second operand of backward composition $\mathbf{B}_<$ ("M" in the schemes above) the *functor* and the other operand ("N") the *argument* of the composition operation. In the schemes given above, the argument is always matched with an outermost argument place of of the functor (the rightmost one in the case of $\mathbf{B}_>$ and the leftmost one in the case of $\mathbf{B}_<$). The notion of function composition to be employed here generalizes this aspect: $\mathbf{B}_>$ can target any forward looking (and $\mathbf{B}_<$ any backward looking) argument slot of the functor. To keep track of the argument slot to be addressed, we use a superscript notation. Generalized Function Composition is subject to certain constraints; certain instances are only licit if one or both of the operands are closed, i.e. do not contain free variables.

DEFINITION 3.1 (Generalized Function Composition)

$$\frac{X \to M : A/B_1/\cdots/B_i \qquad Y \to N : C_{n_l} \backslash \cdots \backslash C_1 \backslash B_1/D_1/\cdots/D_{n_r}}{XY \to \mathbf{B}^i_>(M,N) : C_{n_l} \backslash \cdots \backslash C_1 \backslash A/D_1/\cdots/D_{n_r}/B_2/\cdots/B_i}\ \mathbf{B}^i_>$$

$i = 1$ or $n_l = 0$.
If $i > 1$, then N is closed.
If $n_l > 0$ then M is closed.

$$\frac{X \to N : C_{n_l} \backslash \cdots \backslash C_1 \backslash B_1/D_1/\cdots/D_{n_r} \qquad Y \to M : B_i \backslash \cdots \backslash B_1 \backslash A}{XY \to \mathbf{B}^i_<(N,M) : B_i \backslash \cdots \backslash B_2 \backslash C_{n_l} \backslash \cdots \backslash C_1 \backslash A/D_1/\cdots/D_{n_r}}\ \mathbf{B}^i_<$$

$i = 1$ or $n_r = 0$.
If $i > 1$, then N is closed.
If $n_r > 0$ then M is closed.

Furthermore we assume any type instance of the identity combinator $\mathbf{I}$:

Definition 3.2 (Identity Combinator)

$$\frac{}{\to \mathbf{I}^A_> : A/A}\ \mathbf{I}^A_>$$

$$\frac{}{\to \mathbf{I}^A_< : A \backslash A}\ \mathbf{I}^A_<$$

Finally, an appropriate combinatory reformulation of $\mathbf{LL}$ would require generalizations of the "swapping" rule $A \backslash (B/C) \Leftrightarrow (A \backslash B)/C$. Since this rule is reversible, it induces an equivalence relation on types. We thus suppress the swapping rule an instead tacitly consider each type in a derivation as a representative of its equivalence class induced by swapping.

Let us say that a sequent $X \to A$ is derivable in $\mathbf{LL}_C$ iff it is derivable by using only the identity axiom $[id]$ and the combinatory schemes $\mathbf{I}^A_>, \mathbf{I}^A_<, \mathbf{B}^i_>$, and $\mathbf{B}^i_<$ for arbitrary types A and natural numbers i.

Theorem 3.3
A sequent $X \to A$ is derivable in $\mathbf{LL}_C$ if and only if it is derivable in $\mathbf{LL}$.

Proof: Let us start with introducing some syntactic sugar. Following [10], we write $\lambda^l x.M$ in Curry-Howard terms rather than $\lambda x.M$ if the main functor of the type of the whole term is "$\backslash$"; otherwise we write $\lambda^r x.M$. Furthermore, if M has type $B \backslash A$ and N has type B, we write (NM) rather than (MN). If we talk about the rightmost or leftmost free variable occurrence in a term, we assume the linearization induced by this modified notation. This notation has the obvious advantage that the sequence of free variables in a derivable proof term directly mirrors the sequence of premises of the corresponding proof (a feature that is shared by the combinatory system).

Wansing ([10], p. 70) proves the following lemma.

Lemma 3.4
A term M is a proof term of an $\mathbf{LL}$-proof iff

1. Every λ binds exactly one variable occurrence.
2. For every subterm $\lambda^l x.N$ of M, x is the leftmost free variable occurrence in N.

3. For every subterm $\lambda^r x.N$ of M, x is the rightmost free variable occurrence in N.

Now we can start with the proof of the *if*-direction. Let a labeled $\mathbf{LL}_C$-proof be given. We define a translation from $\mathbf{LL}_C$ proof terms to Curry-Howard terms of the ND calculus.

DEFINITION 3.5 (Translation $\mathbf{LL}_C \to$ ND)

1. $(\mathbf{I}^A_>)^\# = \lambda^r x.x : A/A$
2. $(\mathbf{B}^i_>(M, N))^\# =$
   $\lambda^r y_i \ldots y_2 \lambda^l x_{n_l} \ldots x_1 \lambda^r z_{n_r} \ldots z_1.(M^\# y_i \ldots y_2(x_1 \ldots x_{n_l} N^\# z_{n_r} \ldots z_1))$
3. $(\mathbf{I}^A_<)^\# = \lambda^l x.x : A \setminus A$
4. $(\mathbf{B}^i_<(N, M))^\# =$
   $\lambda^l y_i \ldots \lambda^l y_2 \lambda^r z_{n_r} \ldots z_1 \lambda^l x_{n_l} \ldots x_1.((x_1 \ldots x_{n_l} N^\# z_{n_r} \ldots z_1) y_2 \ldots y_i M^\#)$
   where the types of M, N are as above and the types of x_k, y_k, and z_k are C_k, B_k, and D_k respectively.

It is easy to see that the translation of every well-formed combinatory proof term is a derivable ND term, so every derivation in $\mathbf{LL}_C$ can be transformed into a Natural Deduction proof.

As a consequence of this, adding the combinatory rules to the ND calculus does not extend the notion of derivability of the calculus. We can thus work with a hybrid calculus that makes use both of ND and of combinatory rules without transcending the bounds of $\mathbf{LL}$. At the level of terms, we work with a hybrid system as well; the new term language contains application, abstraction, and the combinators from $\mathbf{LL}_C$. It is easy to see that a term of the hybrid term language is a derivable proof term if and only if it obeys both the constraints on the applicability of the combinators and Wansing's constraints on abstraction.

As next step we define a reduction relation on this hybrid term language.

DEFINITION 3.6 (Reduction)

$$
\begin{array}{c}
(M : A/B, N : B) \rightsquigarrow \mathbf{B}^1_>(M, N) \\
(N : B, M : B \setminus A) \rightsquigarrow \mathbf{B}^1_<(N, M) \\
\lambda^l x.x \rightsquigarrow \mathbf{I}_< \\
\lambda^l x.\mathbf{B}^i_>(M[x], N) \rightsquigarrow \mathbf{B}^i_>(\lambda^l x.M, N) \\
\lambda^l x.\mathbf{B}^i_>(M, N[x]) \rightsquigarrow \mathbf{B}^i_>(M, \lambda^l x.N) \\
\lambda^l x.\mathbf{B}^i_<(N[x], M) \rightsquigarrow \mathbf{B}^i_<(\lambda^l x.N, M) \\
\lambda^l x.\mathbf{B}^i_<(N, M[x]) \rightsquigarrow \mathbf{B}^{i+1}_<(N, \lambda^l x.M) \\
\lambda^r x.x \rightsquigarrow \mathbf{I}_> \\
\lambda^r x.\mathbf{B}^i_>(M[x], N) \rightsquigarrow \mathbf{B}^{i+1}_>(\lambda^r x.M, N) \\
\lambda^r x.\mathbf{B}^i_>(M, N[x]) \rightsquigarrow \mathbf{B}^i_>(M, \lambda^r x.N) \\
\lambda^r x.\mathbf{B}^i_<(N[x], M) \rightsquigarrow \mathbf{B}^i_<(\lambda^r x.N, M) \\
\lambda^r x.\mathbf{B}^i_<(N, M[x]) \rightsquigarrow \mathbf{B}^i_<(N, \lambda^r x.M)
\end{array}
$$

The reduction relation is generalized to subterms in the obvious way:
if $M \rightsquigarrow M'$, then $N[M/x] \rightsquigarrow N[M'/x]$.

Note that the output of each reduction step is a well-formed proof term provided the input is well-formed. So term reduction does not lead beyond the bounds of the hybrid combinatory/ND system. Furthermore, neither the sequence of free variables of a proof term nor the type of the term itself is affected by any reduction step. So input and output of a reduction step represent proofs of the same sequent. Note furthermore that any occurrence of application in a well-formed proof term matches the left hand side of a reduction step. The same holds for λ-abstraction, with the exception of configurations where a λ-abstraction takes immediate scope over another abstraction.

Given this, it is straightforward to show that every well-formed proof term of the hybrid system can be normalized to a pure combinatory term. A possible algorithm starts with replacing all instances of application by **B**. In the next step, the leftmost λ-operator that has no other λ in its scope is targeted by a reduction step (as pointed out in the last paragraph, this is always possible). Every λ-reduction step either reduces the number of combinators in the scope of the targeted λ-operator, or it removes it altogether. So after finitely many reduction steps, the number of λ-operators is reduced by one. Repeating this procedure eventually leads to a normal form term that neither contains λs nor applications, i.e. a pure combinatory term. So starting with an arbitrary ND-proof of a sequent $X \rightarrow A$, by normalizing its Curry-Howard term we end up with a $\mathbf{LL}_C$-proof term and thus an $\mathbf{LL}_C$-proof of this very sequent. $\dashv$

4 Conclusion

The present paper presented a novel Combinatory Categorial system making use of the identity combinator, a generalized version of function composition and—implicitly—swapping. As main result, it was proved that under the appropriate restriction of generalized function composition, this combinatory system is equivalent to the product free version of the calculus **LL**, the variant of the associative Lambek calculus that does without the restriction to empty premises.

At the present point, the significance of the result lies in the fact that is makes a connection between the two main branches of Categorial Grammar; it is thus a base for transferring concrete analyses from the Lambek tradition into the Combinatory tradition. Two directions for further research suggest themselves: Are there similar constructions for other type logical calculi like full **LL**, **L**, the non-associative Lambek calculus **NL** etc.? And, does the combinatory reformulation offer new insights about the Lambek calculus as such? Especially investigations of the proof theoretic complexity of type logical grammars might profit from the combinatory perspective.

References

[1] Anthony E. Ades and Mark J. Steedman. On the order of words. *Linguistics and Philosophy*, 4:517–558, 1982.

[2] Kazimierz Ajdukiewicz. Die syntaktische Konnexität. *Studia Philosophica*, 1:1–27, 1935.

[3] Yehoshua Bar-Hillel. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58, 1953.

[4] Bob Carpenter. The Turing-completeness of multimodal Categorial Grammars. unpublished ms., 1995.

[5] Aravind Joshi, K. Vijay-Shanker, and David Weir. The convergence of mildly context-sensitive grammar formalisms. In Tom Wasow and Peter Sells, editors, *Processing of Linguistic Structure*, pages 31–81. MIT Press, Cambridge (Mass.), 1990.

[6] Geert-Jan Kruijff and Jason Baldridge. Relating Categorial Type Logics and CCG through simulation. ms., University of Edinburgh, 2000.

[7] Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958.

[8] Michael Moortgat. Categorial type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 2, pages 93–178. Elesvier, MIT Press, 1997.

[9] Martin Pentus. Lambek grammars are context-free. In *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*. Montreal, 1993.

[10] Heinrich Theodor Wansing. *The Logic of Information Structure*. PhD thesis, Freie Universität Berlin, 1992.