Continuation Semantics for Symmetric Categorial Grammar

Raffaella Bernardi[1] and Michael Moortgat[2,⋆]

[1] Free University of Bozen-Bolzano, Italy
bernardi@inf.unibz.it
[2] Utrecht Institute of Linguistics OTS, The Netherlands
moortgat@let.uu.nl

Abstract. Categorial grammars in the tradition of Lambek [1,2] are asymmetric: sequent statements are of the form $\Gamma \Rightarrow A$, where the succedent is a single formula A, the antecedent a structured configuration of formulas $A_1, \ldots, A_n$. The absence of structural context in the succedent makes the analysis of a number of phenomena in natural language semantics problematic. A case in point is scope construal: the different possibilities to build an interpretation for sentences containing generalized quantifiers and related expressions. In this paper, we explore a symmetric version of categorial grammar based on work by Grishin [3]. In addition to the Lambek product, left and right division, we consider a dual family of type-forming operations: coproduct, left and right difference. Communication between the two families is established by means of structure-preserving distributivity principles. We call the resulting system **LG**. We present a Curry-Howard interpretation for **LG**$(/, \backslash, \oslash, \oslash)$ derivations. Our starting point is Curien and Herbelin's sequent system for $\lambda\mu$ calculus [4] which capitalizes on the duality between logical implication (i.e. the Lambek divisions under the formulas-as-types perspective) and the difference operation. Importing this system into categorial grammar requires two adaptations: we restrict to the subsystem where linearity conditions are in effect, and we refine the interpretation to take the left-right symmetry and absence of associativity/commutativity into account. We discuss the continuation-passing-style (CPS) translation, comparing the call-by-value and call-by-name evaluation regimes. We show that in the latter (but not in the former) the types of **LG** are associated with appropriate denotational domains to enable a proper treatment of scope construal.

1 Background

Lambek-style categorial grammars offer an attractive computational perspective on the principle of compositionality: under the Curry-Howard interpretation,

⋆ We thank Chris Barker and Ken Shan for sharing their view on continuation semantics with us at an earlier presentation of Lambek-Grishin calculus at the workshop Proof Theory at the Syntax-Semantics Interface (LSA Institute, Harvard/MIT, July 2005). Special thanks to Peter Selinger, for helpful discussion on the duality between call-by-value and call-by-name during GEOCAL'06 (Marseille-Luminy, February 2006) and to Philippe de Groote for bringing Curien and Herbelin's work to our attention. All errors remain our own.

D. Leivant and R. de Queiroz (Eds.): WoLLIC 2007, LNCS 4576, pp. 53–71, 2007.

derivations are associated with instructions for meaning assembly. In natural language semantics, scope construal of generalized quantifier expressions presents an ideal testing ground to bring out the merits of this approach. Scope construal exemplifies a class of phenomena known as *in situ* binding. An *in situ* binder syntactically occupies the position of a phrase of type A; semantically, it binds an A-type variable in that position within a context of type B, producing a value of type C as a result. The inference pattern of (1) (from [5]) schematically captures this behaviour in the format of a sequent rule. The challenge is to solve the equation for the type alias $q(A, B, C)$ in terms of the primitive type-forming operations.

$$\frac{\Delta[x : A] \Rightarrow N : B \qquad\qquad \Gamma[y : C] \Rightarrow M : D}{\Gamma[\Delta[z : q(A, B, C)]] \Rightarrow M[y := (z\ \lambda x.N)] : D}\ . \tag{1}$$

It is a poignant irony that precisely in the area of scope construal, the performance of the original Lambek calculus (whether in its associative or non-associative incarnation) is disappointing. For a sentence-level generalized quantifier (GQ) phrase, we have $A = np$, $B = C = s$ in (1). The type-forming operations available to define $q(np, s, s)$ are the left and right slashes. A first problem is the lack of *type uniformity*. Given standard modeltheoretic assumptions about the interpretation of the type language, an assignment $s/(np\backslash s)$ to a GQ phrase is associated with an appropriate domain of interpretation (a set of sets of individuals), but with such a type a GQ is syntactically restricted to subject positions: for phrase-internal GQ occurrences, context-dependent extra lexical type assignments have to be postulated. Second, this lexical ambiguity strategy breaks down as soon as one considers *non-local* scope construal, where the distance between the GQ occurrence and the sentential domain where it establishes its scope can be unbounded.

The solutions that have been proposed in the type-logical literature we consider suboptimal. The type-shifting approach of Hendriks [6] and the multimodal accounts based on wrapping operations of Morrill and co-workers [7,8] each break the isomorphic relation between derivations and terms that is at the heart of the Curry-Howard interpretation. Hendriks introduces a one-to-many dichotomy between syntactic and semantic derivations. Morrill makes the opposite choice: a multiplicity of syntactically distinct implicational operations which collapse at the semantic level.

The approach we develop in the sections below sticks to the *minimal* categorial logic: the pure logic of residuation. We overcome the expressive limitations of the Lambek calculi by lifting the single succedent formula restriction and move to a symmetric system where the Lambek connectives (product, left and right division) coexist with a dual family (coproduct, right and left difference). The communication between these two families is expressed in terms of Grishin's [3] distributivity principles. Figure 1 schematically presents the outline of the paper. In §2 we present **LG** in algebraic format and discuss the symmetries that govern the vocabulary of type-forming operations. In §3 we present a 'classical' term language for the **LG** type system, and we discuss how a term τ of type A is obtained as the Curry-Howard image of an **LG** sequent derivation π. In §4 we then study the

CPS interpretation of types and terms, comparing the dual call-by-value $\lceil\cdot\rceil$ and call-by-name $\lfloor\cdot\rfloor$ regimes. Under the CPS interpretation, the classical terms for **LG** derivations are transformed into terms of the simply typed lambda calculus — the terms that code proofs in positive intuitionistic logic. The $\lambda_{\rightarrow}$ terms thus obtained adequately reflect NL meaning composition, and (unlike the terms for Multiplicative Linear Logic or its categorial equivalent **LP**) they are obtained in a structure-preserving way. In §5 we illustrate the approach with a discussion of scope construal. We investigate under what conditions the lexical constants of the original Lambek semantics can be lifted to the call-by-value $[\![\cdot]\!]$ and/or call-by-name $[\![\cdot]\!]$ level, and study how the $\lambda_{\rightarrow}$ terms one obtains after this transformation and β normalisation encode the different possibilities for scope construal. In the concluding section §6, we point to some directions for future work.

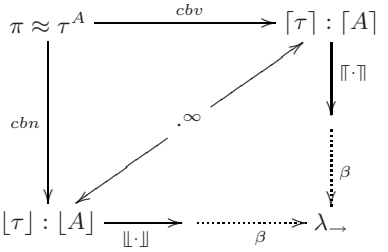


Fig. 1. Outline of the paper

Relation to previous work. Lambek [9] was the first paper to bring Grishin's work under the attention of a wider public. Lambek's bilinear systems are both stronger and weaker than what we propose here: they have hard-wired associativity for $\otimes, \oplus$, which means that control over constituent structure is lost; in addition, only half of the Grishin laws are taken into account ($G1$, $G3$ in Figure 2), an omission that precludes the account of non-peripheral scope construal presented here. De Groote [10] introduced $\lambda\mu$ calculus and continuations into the linguistic discussion of scope construal; Barker and Shan, in a series of papers ([11,12,13] among others), have been advocating this approach for a variety of semantic phenomena. We discuss the relation of our proposal to theirs in §6. Duality between the call-by-value and call-by-name evaluation strategies has been obtained in [14,4,15], among others. Our starting point is the Curien/Herbelin system because, in contrast to the other cited works, it has implication and difference as primitive operations.

2 The Lambek-Grishin Calculus

The calculus that we will use in this paper is presented in Figure 2. We refer to this system as **LG**. In (2), one finds the extended vocabulary of **LG**: the familiar Lambek operators $\otimes, \backslash, /$ are complemented with a family $\oplus, \oslash, \otimes$. In verbal communication, we pronounce $B\backslash A$ as 'B under A', A/B as 'A over B', $B \otimes A$

as 'B from A' and $A \oslash B$ as 'A less B'. As usual under the formulas-as-types perspective, we can view the expressions of (2) as *types* with $\otimes, \backslash, /, \oplus, \oslash, \otimes$ as type-forming operations, or as logical *formulas*, with $\otimes, \backslash, /, \oplus, \oslash, \otimes$ as connectives.

$$\begin{array}{lll} A, B ::= p \mid & & \text{atoms: } s \text{ sentence, } np \text{ noun phrases, } \ldots \\ \quad A \otimes B \mid B \backslash A \mid A / B \mid & & \text{product, left vs right division (TYPES)} \\ & & \text{tensor, left vs right implication (FORMULAS)} \\ \quad A \oplus B \mid A \oslash B \mid B \otimes A & & \text{coproduct, right vs left difference (TYPES)} \\ & & \text{cotensor, left vs right coimplication (FORMULAS)} \end{array} \tag{2}$$

The **LG** type system exhibits rich symmetries, discussed in full in [16]. In the present paper, two kinds of mirror symmetry will play an important role. The first $\cdot^{\bowtie}$ is internal to the $\otimes$ and $\oplus$ families and is order-preserving; the second $\cdot^{\infty}$ relates the $\otimes$ and $\oplus$ families and is order-reversing. We have $p^{\bowtie} = p = p^{\infty}$ and the (bidirectional) translation tables of (3).

$$\bowtie \frac{\begin{array}{cccc} C/D & A \otimes B & B \oplus A & D \otimes C \end{array}}{\begin{array}{cccc} D \backslash C & B \otimes A & A \oplus B & C \oslash D \end{array}} \quad \infty \frac{\begin{array}{cccc} C/B & A \otimes B & A \backslash C \end{array}}{\begin{array}{cccc} B \otimes C & B \oplus A & C \oslash A \end{array}} \tag{3}$$

(PRE-ORDER)		$A \leq A$	$\dfrac{A \leq B \quad B \leq C}{A \leq C}$
(RESIDUATION)		$A \leq C/B$ iff $A \otimes B \leq C$ iff $B \leq A \backslash C$	
(DUAL RESIDUATION)		$C \oslash B \leq A$ iff $C \leq A \oplus B$ iff $A \otimes C \leq B$	
(GRISHIN INTERACTION)			
	$(G1)$	$(A \otimes B) \otimes C \leq A \otimes (B \otimes C)$	$C \otimes (B \oslash A) \leq (C \otimes B) \oslash A$ $(G3)$
	$(G2)$	$C \otimes (A \otimes B) \leq A \otimes (C \otimes B)$	$(B \oslash A) \otimes C \leq (B \otimes C) \oslash A$ $(G4)$

Fig. 2. LG: symmetric Lambek calculus with Grishin interaction principles

Algebraically, the Lambek operators $/, \otimes, \backslash$ form a residuated triple; likewise, the $\otimes, \oplus, \oslash$ family forms a dual residuated triple. The *minimal* symmetric categorial grammar consists of just the preorder axioms (reflexivity and transitivity of $\leq$) together with these (dual) residuation principles.[1]

Grishin's interaction principles. The minimal symmetric system is of limited use if one wants to address the linguistic problems discussed in the introduction. In a system with just the (dual) residuation principles, for every theorem of the (non-associative) Lambek calculus, one also has its image under $\cdot^{\infty}$: $A \leq B$ iff

[1] For a comprehensive overview (from a Display Logic perspective) of the substructural space of which **LG** is an inhabitant, the reader can consult [17]. De Groote and Lamarche [18] present sequent calculus and proof nets for a negation-tensor-par formulation of Classical Non-associative Lambek Calculus, of which **LG** is the subsystem given by the polarities of the operators in (2).

$B^\infty \leq A^\infty$. Interaction between the $\otimes$ and the $\oplus$ family, however, is limited to gluing together theorems of the two families via cut. This limited interaction means that a formula from the $\oplus$ family which is trapped in a $\otimes$ context (or vice versa) will be inaccessible for logical manipulation.

The interaction principles proposed in [3] address this situation. Consider first $G1$ and $G2$ in Fig 2. On the lefthand side of the inequality, a coimplication $A \oslash B$ is hidden as the first or second coordinate of a product. The postulates invert the dominance relation between $\otimes$ and $\oslash$, raising the subformula A to a position where it can be shifted to the righthand side by means of the dual residuation principle. $G3$ and $G4$ are the images of $G1$ and $G2$ under $\cdot^{\bowtie}$. Similarly, a left or right implication trapped within a $\oplus$ context can be liberated by means of the $\cdot^\infty$ images in (4). Combined with transitivity, the Grishin postulates take the form of inference rules ($G1$: from $A \oslash (B \otimes C) \leq D$ conclude $(A \oslash B) \otimes C \leq D$, etc.)

$$\begin{array}{llll} (G3)^\infty & A\backslash(B \oplus C) \leq (A\backslash B) \oplus C & (C \oplus B)/A \leq C \oplus (B/A) & (G1)^\infty; \\ (G4)^\infty & A\backslash(C \oplus B) \leq C \oplus (A\backslash B) & (B \oplus C)/A \leq (B/A) \oplus C & (G2)^\infty. \end{array} \tag{4}$$

The Grishin laws manifest themselves in many forms. The key observation for their usefulness in the analysis of scope construal lies in the fact that $(B \oslash C) \oslash A \leq C/(A\backslash B)$ is a theorem of **LG**. This means that a Lambek type $s/(np\backslash s)$ is derivable from a $(s \oslash s) \oslash np$ type; what can be done with the former can also be done with the latter, but the coimplication type also has non-local capabilities thanks to the Grishin interactions.

Apart from the interaction principles $G1$–$G4$ (and their duals) which will be at the heart of our analysis of scope construal, Grishin considers other options for generalizing Lambek calculus. The reader is referred to [16] for a discussion of these options and their potential linguistic uses. Also in [16] one finds a decision procedure for **LG** based on the monotonicity laws for the type-forming operations, together with the residuation principles and the Grishin principles in rule form.

3 Proofs and Terms

The term language we will use for **LG** derivations is a directional version of Curien/Herbelin's classical $\overline{\lambda}\mu\tilde{\mu}$ terms which takes the $\cdot^{\bowtie}$ symmetry into account. The term language distinguishes terms, coterms (contexts) and commands. We give the syntax of the term language in (5). For terms, we use x, M, N; for coterms (contexts) α, K, L; commands are cuts $M * L$ between a term M and a coterm L. We overload the notation, writing $x \setminus M$ versus $M \,/\, x$ for the left and right abstraction constructs; similarly for coabstraction. As discussed in §1, the familiar lambda abstraction of $\lambda_{\rightarrow}$ will be reinstalled as a result of the CPS transformation on the terms of (5).

$$
\begin{array}{lrl}
 & x \in \mathsf{Term}^A & \text{if } x \in \mathsf{Var}^A \\
\text{(L ABSTRACTION)} & x\backslash M \in \mathsf{Term}^{B\backslash A} & \text{if } x \in \mathsf{Var}^B, M \in \mathsf{Term}^A \\
\text{(R ABSTRACTION)} & M/x \in \mathsf{Term}^{A/B} & \text{if } x \in \mathsf{Var}^B, M \in \mathsf{Term}^A \\
\text{(L COAPPLICATION)} & K \prec M \in \mathsf{Term}^{B \oslash A} & \text{if } K \in \mathsf{CoTerm}^B, M \in \mathsf{Term}^A \\
\text{(R COAPPLICATION)} & M \succ K \in \mathsf{Term}^{A \oslash B} & \text{if } K \in \mathsf{CoTerm}^B, M \in \mathsf{Term}^A \\
\text{(R SHIFT)} & \mu\alpha.(x * K) \in \mathsf{Term}^B & \text{if } \alpha \in \mathsf{CoVar}^B, x \in \mathsf{Var}^A, K \in \mathsf{CoTerm}^A \\
 & \alpha \in \mathsf{CoTerm}^A & \text{if } \alpha \in \mathsf{CoVar}^A \\
\text{(L APPLICATION)} & M \ltimes K \in \mathsf{CoTerm}^{B\backslash A} & \text{if } K \in \mathsf{CoTerm}^A, M \in \mathsf{Term}^B \\
\text{(R APPLICATION)} & K \rtimes M \in \mathsf{CoTerm}^{A/B} & \text{if } K \in \mathsf{CoTerm}^A, M \in \mathsf{Term}^B \\
\text{(L COABSTR)} & \alpha \oslash K \in \mathsf{CoTerm}^{B \oslash A} & \text{if } \alpha \in \mathsf{CoVar}^B, K \in \mathsf{CoTerm}^A \\
\text{(R COABSTR)} & K \oslash \alpha \in \mathsf{CoTerm}^{A \oslash B} & \text{if } \alpha \in \mathsf{CoVar}^B, K \in \mathsf{CoTerm}^A \\
\text{(L SHIFT)} & \widetilde{\mu}x.(M * \alpha) \in \mathsf{CoTerm}^A & \text{if } x \in \mathsf{Var}^A, M \in \mathsf{Term}^B, \alpha \in \mathsf{CoVar}^B
\end{array}
\tag{5}
$$

As in the case of the Lambek calculus, for **LG** we are interested in the resource-sensitive sublanguage. This means that the (co)abstraction and (co)application cases are subject to a linearity condition: the (co)variable bound in a (co) abstraction occurs free exactly once in the body; in the (co)application case the sets of free (co)variables of the term and coterm involved are disjoint. Our use of cut is restricted to patterns $x*K$ $(M*\alpha)$ in the shift right (left) construct, where μ $(\widetilde{\mu})$ obeys the single-bind restriction.

The dualities we discussed for the type system extend to the term language: (6) and (7). The latter acts on the directional constructs; identity otherwise.

$$
\begin{array}{rclcrcl}
x^\infty & = & \alpha & & \alpha^\infty & = & x; \\
(x \setminus M)^\infty & = & M^\infty \oslash \alpha & & (K \oslash \alpha)^\infty & = & x \setminus K^\infty; \\
(M / x)^\infty & = & \alpha \oslash M^\infty & & (\alpha \oslash K)^\infty & = & K^\infty / x; \\
(M \succ K)^\infty & = & K^\infty \ltimes M^\infty & & (M \ltimes K)^\infty & = & K^\infty \succ M^\infty; \\
(K \prec M)^\infty & = & M^\infty \rtimes K^\infty & & (K \rtimes M)^\infty & = & M^\infty \prec K^\infty; \\
(\mu\beta.(x * K))^\infty & = & \widetilde{\mu}y.(K^\infty * \alpha) & & (\widetilde{\mu}y.(M * \alpha))^\infty & = & \mu\beta.(x * M^\infty).
\end{array}
\tag{6}
$$

$$
\begin{array}{rlrl}
(M \succ K)^\bowtie = K^\bowtie \prec M^\bowtie & & (M \ltimes K)^\bowtie = K^\bowtie \rtimes M^\bowtie; \\
(x \setminus M)^\bowtie = M^\bowtie / x & & (K \oslash \alpha)^\bowtie = \alpha \oslash K^\bowtie.
\end{array}
\tag{7}
$$

3.1 LG Sequent Calculus

In Lambek calculus, sequents are statements $\Gamma \Rightarrow B$, where Γ is a binary tree with formulas $A_1,\ldots,A_n$ at the yield, and B is a single formula. The structure-building operation which puts together the antecedent tree is the counterpart of the $\otimes$ logical operation. In **LG**, sequents $\Gamma \Rightarrow \Delta$ can have structures both in the antecedent and in the succedent. The sequent interpunction (which we write $\cdot \circ \cdot$) is the structural counterpart of $\otimes$ in the antecedent, and of $\oplus$ in the succedent. Notice that in the absence of associativity, $\circ$ is a binary operation.

In the rules below, we decorate **LG** derivations with the terms of (5). We distinguish sequents $\Gamma \xrightarrow{M} \Delta[B]$ and cosequents $\Gamma[A] \xrightarrow{K} \Delta$ with proof term M and

coterm K respectively. A sequent (cosequent) has precisely one active succedent (antecedent) formula. The active formula is unlabeled. The passive antecedent (succedent) formulas are labeled with distinct variables x_i (covariables α_i).

For the axiomatic case, we distinguish two versions, depending on whether the succedent or the antecedent is the active formula. The rules ($\leftharpoonup$) and ($\rightharpoonup$) make it possible to shift the focus from antecedent to succedent or vice versa. These rules are in fact restricted cuts, where one of the premises is axiomatic (Axiom or Co-Axiom).

$$\frac{}{x : A \xrightarrow{x} A}\ \text{Ax} \qquad \frac{}{A \xrightarrow{\alpha} \alpha : A}\ \text{Co-Ax} \tag{8}$$

$$\frac{\Gamma[A] \xrightarrow{K} \Delta[\alpha : B]}{\Gamma[x : A] \xrightarrow{\mu\alpha.(x * K)} \Delta[B]}\ (\leftharpoonup) \qquad \frac{\Gamma[x : A] \xrightarrow{M} \Delta[B]}{\Gamma[A] \xrightarrow{\widetilde{\mu}x.(M * \alpha)} \Delta[\alpha : B]}\ (\rightharpoonup) \tag{9}$$

Let us now consider the sequent left and right rules for the connectives. We restrict attention to the (co)implication fragment, i.e. we only cater for $\otimes$ and $\oplus$ in their 'structural' form $\circ$ as antecedent resp. succedent punctuation. The rules of use for the (co)implications are given in (10): these are two-premise rules, introducing an implication (coimplication) in the antecedent (succedent). Notice that we find the $\cdot^{\bowtie}$ and $\cdot^{\infty}$ symmetries here at the level of the inference rules, with $\cdot^{\infty}$ ($\cdot^{\bowtie}$) relating pairs of rules in the horizontal (vertical) dimension.

$$\frac{B \xrightarrow{K} \Delta \quad \Delta' \xrightarrow{M} \Gamma[A]}{\Delta' \xrightarrow{M \succ K} \Gamma[(A \oslash B) \circ \Delta]}\ (\oslash R) \qquad \frac{\Delta \xrightarrow{M} B \quad \Gamma[A] \xrightarrow{K} \Delta'}{\Gamma[\Delta \circ (B \backslash A)] \xrightarrow{M \ltimes K} \Delta'}\ (\backslash L)$$

$$\begin{array}{ccc} (\oslash R) & \leftarrow \cdot^{\infty} \rightarrow & (\backslash L) \\ \uparrow & & \uparrow \\ \cdot^{\bowtie} & & \cdot^{\bowtie} \\ \downarrow & & \downarrow \\ (\oslash R) & \leftarrow \cdot^{\infty} \rightarrow & (/L) \end{array}$$

$$\frac{B \xrightarrow{K} \Delta \quad \Delta' \xrightarrow{M} \Gamma[A]}{\Delta' \xrightarrow{K \prec M} \Gamma[\Delta \circ (B \oslash A)]}\ (\oslash R) \qquad \frac{\Delta \xrightarrow{M} B \quad \Gamma[A] \xrightarrow{K} \Delta'}{\Gamma[(A/B) \circ \Delta] \xrightarrow{K \rtimes M} \Delta'}\ (/L) \tag{10}$$

The rules of proof for the (co)implications are given in (10): these are one-premise rules, introducing an implication (coimplication) in the succedent (antecedent).

$$\frac{x : B \circ \Gamma \xrightarrow{M} \Delta[A]}{\Gamma \xrightarrow{x \setminus M} \Delta[B \backslash A]}\ (\backslash R) \qquad \frac{\Gamma[A] \xrightarrow{K} \Delta \circ \alpha : B}{\Gamma[A \oslash B] \xrightarrow{K \oslash \alpha} \Delta}\ (\oslash L)$$

$$\begin{array}{ccc} (\backslash R) & \leftarrow \cdot^{\infty} \rightarrow & (\oslash L) \\ \uparrow & & \uparrow \\ \cdot\bowtie & & \cdot\bowtie \\ \downarrow & & \downarrow \\ (/R) & \leftarrow \cdot^{\infty} \rightarrow & (\oslash L) \end{array} \tag{11}$$

$$\frac{\Gamma \circ x : B \xrightarrow{M} \Delta[A]}{\Gamma \xrightarrow{M \,/\, x} \Delta[A/B]}\ (/R) \qquad \frac{\Gamma[A] \xrightarrow{K} \alpha : B \circ \Delta}{\Gamma[B \oslash A] \xrightarrow{\alpha \oslash K} \Delta}\ (\oslash L)$$

Observe that to prove the soundness of the coimplication (implication) rules of proof from the algebraic presentation, one uses the Grishin interaction principles to move the B subformula upwards through the $\otimes$ context ($\oplus$ context), and then shifts it to the succedent (antecedent) part via the residuation principles. The Grishin interaction principles, in other words, are *absorbed* in these rules of proof. We illustrate this in (12) for $(\backslash R)$, writing $\Gamma^{\bullet}$ (Δ°) for the formula equivalent of an antecedent (succedent) structure. The vertical dots abbreviate a succession of Grishin interaction steps.

$$\begin{array}{c} \dfrac{B \otimes \Gamma^{\bullet} \leq \Delta^{\circ}[A]}{\Gamma^{\bullet} \leq B \backslash \Delta^{\circ}[A]} \\ \vdots \\ \Gamma^{\bullet} \leq \Delta^{\circ}[B \backslash A] \end{array} \tag{12}$$

As indicated in §2, we will use the formula $(B \oslash C) \oslash A$ to do the work of the *in situ* binder schema $q(A, B, C)$. (Alternatively, we could have used its $\cdot^{\bowtie}$ dual $A \oslash (C \oslash B)$.) The (qL) and (qR) rules of Fig 3 have the status of derived inference rules. We will use them in §5 to present proofs and terms in a more compact format. In §A we give a worked-out derivation of $(B \oslash C) \oslash A \Rightarrow C/(A \backslash B)$, together with further abbreviatory conventions. The reader may want to check that a cut of (qR) against (qL) can be rewritten with cuts on the subformulae A, B, C, as required: $\texttt{cobind}(M^A, K^B, \beta^C) * \texttt{bind}(x^A, N^B, L^C) \longrightarrow_{\beta} M * \widetilde{\mu} x.(\mu \beta.(N * K) * L)$. One should keep in mind that (qL) and (qR) are short-cuts, i.e. ways of abbreviating a sequence of n inference steps as a one-step inference. For some theorems of **LG**, one cannot take a short-cut: their derivation requires the individual inference rules for the connectives involved. The type transition $(B \oslash C) \oslash A \Rightarrow ((D \backslash B) \oslash (D \backslash C)) \oslash A$ is an example. Its derivation is given in [16].

4 Interpretation: Continuation Semantics

We turn now to an interpretation for **LG** derivations in the continuation-passing-style (CPS). In the semantics of programming languages, CPS interpretation has

$$\frac{C \xrightarrow{L} \Delta \quad \Gamma[x:A] \xrightarrow{N} B}{\Gamma[(B \oslash C) \otimes A] \xrightarrow{\texttt{bind}(x,N,L)} \Delta}\ (qL) \quad \triangleq \quad \frac{\dfrac{C \xrightarrow{L} \Delta \qquad \Gamma[x:A] \xrightarrow{N} B}{\Gamma[x:A] \xrightarrow{N \succ L} (B \oslash C) \circ \Delta}\ (\oslash R)}{\Gamma[A] \xrightarrow{\widetilde{\mu}x.((N \succ L) * \gamma)} \gamma : (B \oslash C), \Delta}\ \rightleftharpoons$$

$$\frac{}{\Gamma[(B \oslash C) \otimes A] \xrightarrow{\gamma \otimes (\widetilde{\mu}x.((N \succ L) * \gamma))} \Delta}\ (\otimes L)$$

$$\frac{B \xrightarrow{K} \Delta' \circ \beta : C \quad \Gamma \xrightarrow{M} \Delta[A]}{\Gamma \xrightarrow{\texttt{cobind}(M,K,\beta)} \Delta[\Delta' \circ ((B \oslash C) \otimes A)]}\ (qR) \quad \triangleq \quad \frac{\dfrac{B \xrightarrow{K} \Delta' \circ \beta : C}{B \oslash C \xrightarrow{K \oslash \beta} \Delta'}\ (\oslash L) \qquad \Gamma \xrightarrow{M} \Delta[A]}{\Gamma \xrightarrow{(K \oslash \beta) \prec M} \Delta[\Delta' \circ ((B \oslash C) \otimes A)]}\ (\otimes R)$$

Fig. 3. Derived inference rules for $(B \oslash C) \otimes A \approx q(A, B, C)$

been a fruitful strategy to make explicit (and open to manipulation) aspects of computation that remain implicit in a direct interpretation. In the direct interpretation, a function simply returns a value. Under the CPS interpretation, functions are provided with an extra argument for the continuation of the computation. This explicit continuation argument is then passed on when functions combine with each other. Key concepts, then, are "computation", "continuation" and "value" and they way they relate to each other for different evaluation strategies.

Curien and Herbelin [4] develop the CPS interpretation for a classical system with an implication and a difference operation; call-by-value (cbv) $\lceil\cdot\rceil$ and call-by-name (cbn) $\lfloor\cdot\rfloor$ regimes are related by the duality between the implication and difference operations. For **LG** we refine the Curien/Herbelin continuation semantics to accommodate the left/right symmetry. We first consider the effect of the CPS interpretation on the level of *types*, comparing a call-by-value (cbv) and a call-by-name (cbn) regime; then we define the CPS interpretation on the level of the *terms* of (5).

Types: call-by-value. The target type language has a distinguished type R of responses, products and functions; all functions have range R. For each type A of the source language, the target language has values $V_A = \lceil A \rceil$, continuations $K_A = R^{V_A}$ (functions from V_A to R) and computations $C_A = R^{K_A}$ (functions from K_A to R).[2] Notice that given the canonical isomorphism $A \times B \to C \cong A \to (B \to C)$, one can also think of a $V_{A \backslash B}$ as a function from A values to B computations. For p atomic, $\lceil p \rceil = p$. In (13), the $\lceil\cdot\rceil$ translations for the (co)implications are related in the vertical dimension by left/right symmetry $\cdot^{\bowtie}$ and in the horizontal dimension by arrow reversal $\cdot^{\infty}$: right difference is dual to left division, left difference dual to right division.

[2] In the schemas (13) and (14) we use exponent notation for function spaces, for comparison with [4]. In the text, we usually shift to the typographically more convenient arrow notation, compare A^B versus $B \to A$.

$$\begin{array}{rclcrcl} \lceil A\backslash B\rceil & = & R^{\lceil A\rceil\times R^{\lceil B\rceil}} & & \lceil B\oslash A\rceil & = & \lceil B\rceil\times R^{\lceil A\rceil};\\ \lceil B/A\rceil & = & R^{R^{\lceil B\rceil}\times\lceil A\rceil} & & \lceil A\obslash B\rceil & = & R^{\lceil A\rceil}\times\lceil B\rceil. \end{array} \tag{13}$$

Types: call-by-name. Under the call-by-name regime, for each type A of the source language, the target language has continuations $K_A = \lfloor A\rfloor$ and computations $C_A = R^{K_A}$. The call-by-name interpretation $\lfloor\cdot\rfloor$ is obtained as the composition of the $\cdot^\infty$ duality map and the $\lceil\cdot\rceil$ interpretation: $\lfloor A\rfloor \triangleq \lceil A^\infty\rceil$. For atoms, $\lfloor p\rfloor = \lceil p^\infty\rceil = p$. For the (co)implications, compare the cbv intepretation (left) with the cbn interpretation (right) in (14).

$$\begin{array}{rclcrcl} \lceil A\backslash B\rceil & = & R^{\lceil A\rceil\times R^{\lceil B\rceil}} & & R^{\lfloor A\rfloor\times R^{\lfloor B\rfloor}} & = & \lfloor B\oslash A\rfloor;\\ \lceil B\oslash A\rceil & = & \lceil B\rceil\times R^{\lceil A\rceil} & & \lfloor B\rfloor\times R^{\lfloor A\rfloor} & = & \lfloor A\backslash B\rfloor;\\ \lceil B/A\rceil & = & R^{R^{\lceil B\rceil}\times\lceil A\rceil} & & R^{R^{\lfloor B\rfloor}\times\lfloor A\rfloor} & = & \lfloor A\obslash B\rfloor;\\ \lceil A\obslash B\rceil & = & R^{\lceil A\rceil}\times\lceil B\rceil & & R^{\lfloor A\rfloor}\times\lfloor B\rfloor & = & \lfloor B/A\rfloor. \end{array} \tag{14}$$

Notice that for the call-by-name regime, the starting point is the level of continuations, not values as under call-by-value. Let's take the definition of $B\oslash A$ by means of example. For call-by-value, one starts from $\lceil B\oslash A\rceil$ (i.e., $V_{B\oslash A}$) that is a pair $V_B\times K_A$; hence its continuation is $K_{B\oslash A} = (V_B\times K_A)\to R$ and its computation is $C_{B\oslash A} = ((V_B\times K_A)\to R)\to R$. On the other hand, the call-by-name interpretation starts at the level of continuations: $\lfloor B\oslash A\rfloor = (K_A\times C_B)\to R$ and from this the computation is obtained as usual, viz. $C_{B\oslash A} = ((K_A\times C_B)\to R)\to R$, hence obtaining a higher order function than the one computed under the call-by-value strategy. This difference will play an important role in the linguistic application of the two strategies.

Terms: cbv versus cbn. Given the different CPS *types* for left and right (co) implications, we can now turn to their interpretation at the *term* level. In (15), we give the cbv interpretation of terms, in (16) of coterms. We repeat the typing information from (5) to assist the reader. The call-by-name regime is the composition of call-by-value and arrow reversal: $\lfloor\cdot\rfloor \triangleq \lceil\cdot^\infty\rceil$. This CPS interpretation of terms is set up in such a way that for sequents with yield $A_1,\ldots,A_n\Rightarrow B$, the cbv interpretation represents the process of obtaining a B *computation* from $A_1,\ldots,A_n$ *values*; the cbn interpretation takes $A_1,\ldots,A_n$ *computations* to a B computation. See Propositions 8.1 and 8.3 of [4].

$$\begin{array}{rrl} A & \lceil x\rceil = \lambda k.k\ x & x:A\\ B\backslash A & \lceil x\setminus M\rceil = \lambda k.(k\ \lambda\langle x,\beta\rangle.\lceil M\rceil\ \beta) & x:B,\ M:A\\ A/B & \lceil M\,/\,x\rceil = \lambda k.(k\ \lambda\langle\beta,x\rangle.\lceil M\rceil\ \beta) & x:B,\ M:A\\ B\oslash A & \lceil M\succ K\rceil = \lambda k.(\lceil M\rceil\ \lambda y.(k\ \langle y,\lceil K\rceil\rangle)) & M:A,\ K:B\\ A\obslash B & \lceil K\prec M\rceil = \lambda k.(\lceil M\rceil\ \lambda y.(k\ \langle\lceil K\rceil,y\rangle)) & M:A,\ K:B\\ B & \lceil\mu\alpha.(x*K)\rceil = \lambda\alpha.(\lceil K\rceil\ x) & \alpha:B,\ x,K:A \end{array} \tag{15}$$

$$\begin{array}{rrl}
A & \lceil\alpha\rceil = \alpha & \alpha : A \\
B \oslash A & \lceil\alpha \oslash K\rceil = \lambda\langle\alpha, x\rangle.(\lceil K\rceil\ x) & \alpha : B,\ K : A \\
A \oslash B & \lceil K \oslash \alpha\rceil = \lambda\langle x, \alpha\rangle.(\lceil K\rceil\ x) & \alpha : B,\ K : A \\
B\backslash A & \lceil M \ltimes K\rceil = \lambda k.(\lceil M\rceil\ \lambda x.(k\ \langle x, \lceil K\rceil\rangle)) & M : B,\ K : A \\
A/B & \lceil K \rtimes M\rceil = \lambda k.(\lceil M\rceil\ \lambda x.(k\ \langle\lceil K\rceil, x\rangle)) & M : B,\ K : A \\
A & \lceil\widetilde{\mu}x.(M * \alpha)\rceil = \lambda x.(\lceil M\rceil\ \alpha) & x : A,\ \alpha, M : B
\end{array} \tag{16}$$

5 Application: Scope Construal

In this section we turn to the linguistic application. Our aim is twofold. First we show that a type assignment $(s \oslash s) \oslash np$ for generalized quantifier phrases solves the problems with $s/(np\backslash s)$ mentioned in §1: the type $(s \oslash s) \oslash np$ uniformly appears in positions that can be occupied by ordinary noun phrases, and it gives rise to ambiguities of scope construal (local and non-local) in constructions with multiple GQ and/or multiple choices for the scope domain. Second, we relate the CPS interpretation to the original interpretation for Lambek derivations by defining translations $[\![\cdot]\!], [\![\cdot]\!]$ lifting the lexical constants from the type they have in the original Lambek semantics to the type required by the cbv or cbn level. To realize this second aim, we assume that the result type R is the type of truth values. For the rest our modeltheoretic assumptions are standard. The domain of interpretation for np values is E (the set of individuals), for s values it is $\{0, 1\}$ (the set of truth values). Out of E and $\{0, 1\}$ one then constructs complex domains in terms of function spaces and Cartesian products. In the case of the original Lambek semantics (or Montague grammar) these domains of interpretation are obtained indirectly, via a mapping between syntactic and semantic types where $np' = e$, $s' = t$ and $(A\backslash B) = (B/A) = A' \to B'$, with $\mathrm{Dom}_e = E$ and $\mathrm{Dom}_t = \{0, 1\}$.

We start with a fully worked-out example of a 'S goes to NP VP' combination, 'Alice left'. The official sequent derivation is given in (17). Consulting the dictionary of Table 1, we fill in the lexical items alice and left of type np and $np\backslash s$ respectively for the x and y parameters of the proof term. Call the resulting term M. We now compare the CPS transformation of M under the cbv and cbn execution regimes as defined in (15) and (16).

$$\dfrac{\dfrac{x : np \xrightarrow{x} np \quad s \xrightarrow{\alpha} \alpha : s}{x : np \circ np\backslash s \xrightarrow{x \ltimes \alpha} \alpha : s}\ (\backslash L)}{x : np \circ y : np\backslash s \xrightarrow{\mu\alpha.(y * (x \ltimes \alpha))} s}\ (\leftrightharpoons) \tag{17}$$

Consider first cbv, on the left in Figure 4. Recall that under the cbv regime a sequent with yield $A_1, \ldots, A_n \Rightarrow B$ maps the A_i values to a B computation. A value of type $np\backslash s$ ($V_{np\backslash s}$), as we see in Table 1, is a function taking a pair of an np value and an s continuation to the result type R, i.e. $V_{np} \times K_s \to R$.

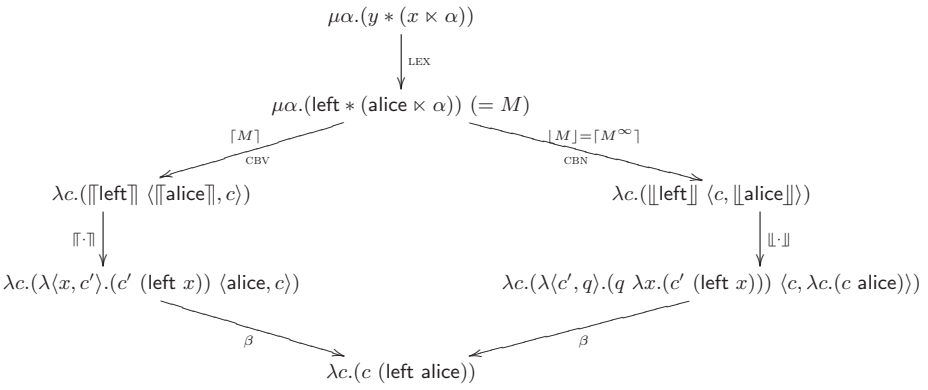


Fig. 4. 'Alice left': cbv versus cbn

Combining V_{np} and $V_{np\backslash s}$ we obtain an s computation, i.e. $(s \to R) \to R$, by giving the $V_{np\backslash s}$ function the pair it requires, and abstracting over the K_s component. This is what we see in $\lceil M \rceil = \lambda c.(\llbracket \mathsf{left} \rrbracket\ \langle \llbracket \mathsf{alice} \rrbracket, c \rangle)$. The next step is to relate the cbv CPS interpretation to the original semantics of the Lambek calculus. In the original semantics, 'Alice' and 'left' would be interpreted in terms of constants of type e and $e \to t$ respectively. The mapping $\llbracket \cdot \rrbracket$ in Table 2 produces terms of type V_{np} and $V_{np\backslash s}$ from constants alice and left of type e and $e \to t$. Substituting these in $\lceil M \rceil$ (and β conversion) gives $\lambda c.(c\ (\mathsf{left}\ \mathsf{alice}))$. Combined with the trivial s continuation (the identity function on $\{0, 1\}$) one obtains $(\mathsf{left}\ \mathsf{alice})$.

On the right in Figure 4 we have the cbn interpretation. Under the cbn regime, a sequent with yield $A_1, \ldots, A_n \Rightarrow B$ maps A_i *computations* to a B computation. We obtain the cbn interpretation by duality: $\lfloor M \rfloor = \lceil M^\infty \rceil = \lceil \widetilde{\mu} x.((x \succ \mathsf{alice}) * \mathsf{left}) \rceil$, i.e. we read off the cbn interpretation from the mirror image derivation $(np \circ np \backslash s \Rightarrow s)^\infty$, which is $s \Rightarrow s \oslash np \circ np$. For the lexical items involved, Table 1 gives the *continuation* types $\lfloor A \rfloor$ corresponding to the source language types A. To obtain the required types for A *computations*, we take functions from these continuations into R. Specifically, the verb $\lfloor\!\lfloor \mathsf{left} \rfloor\!\rfloor$ in this case is interpreted as a function taking a pair $K_s \times C_{np}$ into R; the noun phrase argument $\lfloor\!\lfloor \mathsf{alice} \rfloor\!\rfloor$ also is of type C_{np}, i.e. $(V_{np} \to R) \to R$. Combining these produces a result of type C_s again by abstracting over the K_s component of the pair given to the verb:

Table 1. Lexical entries. $\lceil A \rceil$ is a value of type A; $\lfloor A \rfloor$ is a continuation of type A.

WORD	TYPE	ALIAS	$\lceil \cdot \rceil$ CBV	$\lfloor \cdot \rfloor$ CBN
alice, lewis	np		$\lceil np \rceil$	$\lfloor np \rfloor$
left	$np\backslash s$	iv	$R^{\lceil np \rceil \times R^{\lceil s \rceil}}$	$\lfloor s \rfloor \times R^{\lfloor np \rfloor}$
teases	$(np\backslash s)/np$	tv	$R^{R^{\lceil iv \rceil} \times \lceil np \rceil}$	$R^{\lfloor np \rfloor} \times \lfloor iv \rfloor$
thinks	$(np\backslash s)/s$	tvs	$R^{R^{\lceil iv \rceil} \times \lceil s \rceil}$	$R^{\lfloor s \rfloor} \times \lfloor iv \rfloor$
somebody	$s/(np\backslash s)$	su	$R^{R^{\lceil s \rceil} \times \lceil iv \rceil}$	$R^{\lfloor iv \rfloor} \times \lfloor s \rfloor$

Table 2. Lifting lexical constants: cbv regime

$$
\begin{aligned}
\lceil\!\lceil \mathsf{alice} \rceil\!\rceil &= \mathsf{alice} \\
\lceil\!\lceil \mathsf{lewis} \rceil\!\rceil &= \mathsf{lewis} \\
\lceil\!\lceil \mathsf{left} \rceil\!\rceil &= \lambda\langle x, c\rangle.(c\ (\mathsf{left}\ x)) \\
\lceil\!\lceil \mathsf{teases} \rceil\!\rceil &= \lambda\langle v, y\rangle.(v\ \lambda\langle x, c\rangle.(c\ ((\mathsf{teases}\ y)\ x))) \\
\lceil\!\lceil \mathsf{somebody} \rceil\!\rceil &= \lambda\langle c, v\rangle.(\exists\ \lambda x.(v\ \langle x, c\rangle))
\end{aligned}
$$

$\lfloor M \rfloor = \lambda c.(\lfloor\!\lfloor \mathsf{left} \rfloor\!\rfloor\ \langle c, \lfloor\!\lfloor \mathsf{alice} \rfloor\!\rfloor\rangle)$. The mapping $\lfloor\!\lfloor \cdot \rfloor\!\rfloor$ in Table 3 produces terms of type $\lfloor np \rfloor \to R$ and $\lfloor np\backslash s \rfloor \to R$ (i.e. computations) from constants alice and left of type e and $e \to t$. Substituting these in $\lfloor M \rfloor$ (and β conversion) gives the same result as what we had under the cbv regime.

Table 3. Lifting lexical constants: cbn regime

$$
\begin{aligned}
\lfloor\!\lfloor \mathsf{alice} \rfloor\!\rfloor &= \lambda c.(c\ \mathsf{alice}) \\
\lfloor\!\lfloor \mathsf{lewis} \rfloor\!\rfloor &= \lambda c.(c\ \mathsf{lewis}) \\
\lfloor\!\lfloor \mathsf{left} \rfloor\!\rfloor &= \lambda\langle c, q\rangle.(q\ \lambda x.(c\ (\mathsf{left}\ x))) \\
\lfloor\!\lfloor \mathsf{teases} \rfloor\!\rfloor &= \lambda\langle q, \langle c, q'\rangle\rangle.(q'\ \lambda x.(q\ \lambda y.(c\ ((\mathsf{teases}\ y)\ x)))) \\
\lfloor\!\lfloor \mathsf{somebody} \rfloor\!\rfloor &= \lambda\langle v, c\rangle.(\exists\ \lambda x.(v\ \langle c, \lambda c'.(c'\ x)\rangle))
\end{aligned}
$$

The reader is invited to go throught the same steps for 'Alice teases Lewis'. The term for the derivation is $\mu\alpha.(\mathsf{teases} * ((\mathsf{alice} \ltimes \alpha) \rtimes \mathsf{lewis}))$ $(= M)$, with the CPS interpretations in (18). The variable v is of type $K_{np\backslash s}$, a verb phrase continuation. Consulting the cbv and cbn dictionaries of Tables 2 and 3, we can substitute the required lambda terms for the lexical constants. After this substitution and β reduction, the cbv and cbn interpretations converge on $\lambda c.$ $(c\ ((\mathsf{teases}\ \mathsf{lewis})\ \mathsf{alice}))$.

$$
\begin{aligned}
\lceil M \rceil &= \lambda c.(\lceil\!\lceil \mathsf{teases} \rceil\!\rceil\ \langle \lambda v.(v\ \langle \lceil\!\lceil \mathsf{alice} \rceil\!\rceil, c\rangle), \lceil\!\lceil \mathsf{lewis} \rceil\!\rceil\rangle); \\
\lfloor M \rfloor &= \lambda c.(\lfloor\!\lfloor \mathsf{teases} \rfloor\!\rfloor\ \langle \lfloor\!\lfloor \mathsf{lewis} \rfloor\!\rfloor, \langle c, \lfloor\!\lfloor \mathsf{alice} \rfloor\!\rfloor\rangle\rangle).
\end{aligned}
\tag{18}
$$

In Fig 5 we highlight the type structure of the CPS interpretations for this sentence, showing that (i) call-by-value produces terms consisting of function applications of values to pairs of values and continuations (left tree), whereas (ii) call-by-name produces terms consisting of the application of computation to pairs of computations and continuation types. The observed difference will be relevant for the interpretation of generalized quantifiers expressions to which we now turn.

Scope construal: simple subject GQ. In Table 1, one finds the CPS image under cbv and cbn of a Lambek-style $s/(np\backslash s)$ type assignment for a GQ expression such as 'somebody'. The corresponding lexical recipes for the cbv and cbn regimes is given in Tables 2 and 3, respectively. We leave it as an exercise for the reader to work through a derivation with these types/terms and to verify that a type assignment $s/(np\backslash s)$ is restricted to subject position and to local scope, as

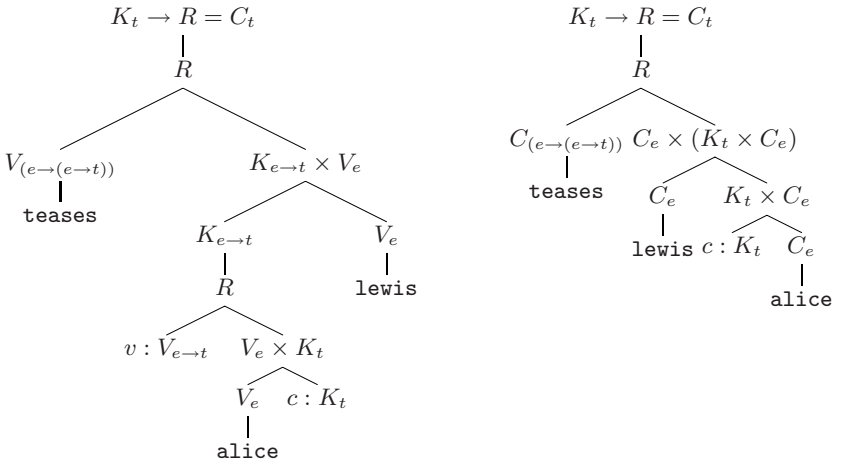


Fig. 5. Type schemas for cbv (left) versus cbn (right)

we saw in §1. Let us turn then to derivations with the type assignment we have proposed: $(s \oslash s) \oslash np$ (alias: gq) — a type assignment that will accommodate both local and non-local scope construals. In (19) we compute the term for the derivation of 'somebody left', using the abbreviatory conventions discussed in the Appendix (e.g., the $\boxed{1}$ in step 2. stands for the proof term computed at step 1.); in (20) its CPS transformation under cbv and cbn. ($z : V_s$, $q : C_{np}$, $y : K_{s\oslash s}$)

$$1.\quad \mu\alpha.(\mathsf{left}{*}(x \ltimes \alpha)) \qquad 2.\quad \mu\beta.(\mathsf{somebody}{*}\mathtt{bind}(x, \boxed{1}, \beta))) \tag{19}$$

$$\boxed{2} = N \qquad \begin{array}{l} \lceil N \rceil = \lambda c.((\llbracket \mathsf{left} \rrbracket\ \langle \pi^2 \llbracket \mathsf{somebody} \rrbracket, \lambda z.(\pi^1 \llbracket \mathsf{somebody} \rrbracket\ \langle z, c\rangle)\rangle)); \\ \lfloor N \rfloor = \lambda c.((\lVert \mathsf{somebody} \rVert\ \lambda\langle q, y\rangle.(y\ \langle c, \lambda c'.(\lVert \mathsf{left} \rVert\ \langle c', q\rangle)\rangle))). \end{array} \tag{20}$$

The difference between cbv and cbn regimes observed above with respect to $\llbracket\cdot\rrbracket, \lVert\cdot\rVert$ in the term in (18) turns out to be of particular interest here. In (21) we give V_{gq} and K_{gq} for the cbv and cbn interpretations respectively. For cbv, this is a pair of an $s\oslash s$ continuation and an np value. This np value would have to be realised as the variable bound by the (logical) constant $\exists$ (of type $(e \to t) \to t$) in the $\llbracket\cdot\rrbracket$ translation. Such a binding relation cannot be established from the $K_{s\oslash s}$ component of the pair. For cbn, the situation is different: $\lfloor gq \rfloor$ has the required structure to specify the lifted lexical recipe of (22). $\mathcal{Q}$ is a function taking a pair of a np computation and an $(s \oslash s)$ continuation to the result type R. In the body of the term, we apply $\mathcal{Q}$ to the C_{np} term $\lambda k.(k\ x)$, where x is the e type variable bound by $\exists$, and to the closed term $\lambda\langle c, p\rangle.(p\ c)$ obtained by applying

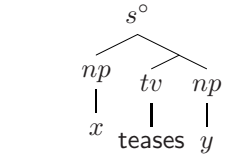
the C_s variable p to the K_s variable c. In combination with $[\![\mathsf{left}]\!]$ given above, $\lfloor N \rfloor$ simplifies to $\lambda c.(\exists\ \lambda x.(c\ (\mathsf{left}\ x)))$ as required. From here on, we stay with the cbn regime.

$$\lceil gq \rceil\ =\ R^{\lceil s \rceil \times R^{\lceil s \rceil}} \times \lceil np \rceil \qquad \lfloor gq \rfloor\ =\ R^{R^{\lfloor np \rfloor} \times R^{\lfloor s \rfloor} \times R^{\lfloor s \rfloor}}. \tag{21}$$

$$[\![\mathsf{somebody}]\!]\ =\ \lambda\mathcal{Q}.(\exists\ \lambda x.(\mathcal{Q}\ \langle \lambda k.(k\ x), \lambda\langle c, p\rangle.(p\ c)\rangle)). \tag{22}$$

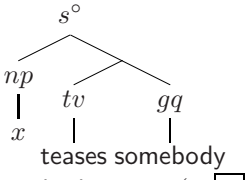
Ambiguous scope construal. First, compare 'Alice teases Lewis' with 'everyone teases somebody' involving two GQ expressions. Starting from $\boxed{1}\ \mu\alpha.(\mathsf{teases} * ((x \ltimes \alpha) \rtimes y))$, the derivation can be completed in two ways, depending on whether we first bind the object variable y, then the subject variable x, or vice versa. On the left the subject wide scope reading, on the right the object wide scope reading.

1.



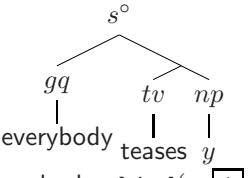
$\mu\alpha.(\mathsf{teases} * ((x \ltimes \alpha) \rtimes y))$

2a.



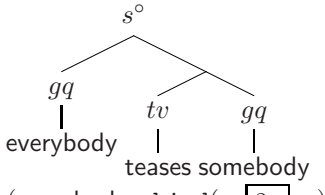
$\mu\beta.(\mathsf{somebody} * \mathtt{bind}(y, \boxed{1}, \beta))$

2b.



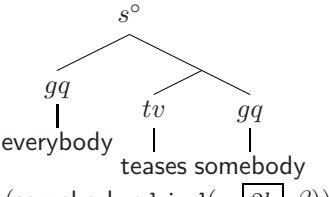
$\mu\gamma.(\mathsf{everybody} * \mathtt{bind}(x, \boxed{1}, \gamma))$

3a.



$\mu\gamma.(\mathsf{everybody} * \mathtt{bind}(x, \boxed{2a}, \gamma)))$

3b.



$\mu\beta.(\mathsf{somebody} * \mathtt{bind}(y, \boxed{2b}, \beta)))$

By applying the method described with previous examples, one obtains the $\forall/\exists$ reading for (3a) and the $\exists/\forall$ reading for (3b). First, the terms are interpreted by means of the definitions in (15) and (16) obtaining the following results:

$$\begin{array}{l} \lfloor 3a \rfloor = \lambda c.([\![\mathsf{evro.}]\!]\ \lambda\langle q, y\rangle.([\![\mathsf{smo.}]\!]\ \lambda\langle p, z\rangle.(y\ \langle c, \lambda c'.(z\ \langle c', \lambda c''.([\![\mathsf{teases}]\!]\ \langle p, \langle c'', q\rangle\rangle)\rangle)\rangle)\rangle)\rangle); \\ \lfloor 3b \rfloor = \lambda c.([\![\mathsf{smo.}]\!]\ \lambda\langle p, z\rangle.([\![\mathsf{evro.}]\!]\ \lambda\langle q, y\rangle.(z\ \langle c, \lambda c'.(y\ \langle c', \lambda c''.([\![\mathsf{teases}]\!]\ \langle p, \langle c'', q\rangle\rangle)\rangle)\rangle)\rangle)\rangle). \end{array} \tag{23}$$

where the variables p, q of type C_{np} are for the object and subject respectively. The y, z variables are $s \oslash s$ continuations, the (primed) c are s continuations. Secondly, the $[\![\cdot]\!]$ translation is applied, and the readings reduce to $\lambda c.(\forall\ \lambda x.(\exists\ \lambda y.(c\ ((\mathsf{teases}\ y)\ x))))$ and $\lambda c.(\exists\ \lambda y.(\forall\ \lambda x.(c\ ((\mathsf{teases}\ y)\ x))))$, respectively.

Local versus non-local scope. Consider the two readings for the sentence 'Alice thinks somebody left'. The ambiguity arises here from the fact that in this context the GQ can non-deterministically select the embedded or the main clause s as its scope domain. We give the terms for these two derivations (local (a) versus non-local (b) scope) in (24), reusing the components $\boxed{1}$ and $\boxed{2}$ of (19), and the cbn interpretations of these terms in (25). These can be further reduced to (26) via a lexical recipe $[\![\mathsf{thinks}]\!] = \lambda\langle p, \langle c, q\rangle\rangle.(q\ \lambda x.(c\ ((\mathsf{thinks}\ (p\ \lambda c.c))\ x)))$ expressed in terms of a constant thinks of type $t \to (e \to t)$.

$$\begin{array}{ll} a. & \mu\gamma.(\mathsf{thinks} * ((\mathsf{alice} \ltimes \gamma) \rtimes \boxed{2})); \\ b. & \mu\gamma.(\mathsf{somebody} * \mathtt{bind}(x, \mu\gamma'.(\mathsf{thinks} * ((\mathsf{alice} \ltimes \gamma') \rtimes \boxed{1})), \gamma)). \end{array} \tag{24}$$

$$\begin{array}{l} \lfloor a \rfloor = \lambda c.([\![\mathsf{thinks}]\!]\ \langle \lambda c'.([\![\mathsf{somebody}]\!]\ \lambda\langle q, y\rangle.(y\ \langle c', \lambda c''.([\![\mathsf{left}]\!]\ \langle c'', q\rangle)\rangle)), \langle c, [\![\mathsf{alice}]\!]\rangle\rangle); \\ \lfloor b \rfloor = \lambda c.([\![\mathsf{somebody}]\!]\ \lambda\langle q, y\rangle.(y\ \langle c, \lambda c'.([\![\mathsf{thinks}]\!]\ \langle \lambda c''.([\![\mathsf{left}]\!]\ \langle c'', q\rangle), \langle c', [\![\mathsf{alice}]\!]\rangle\rangle)\rangle)). \end{array} \tag{25}$$

$$\begin{array}{l} \lfloor a \rfloor \rightsquigarrow_\beta \lambda c.(c\ ((\mathsf{thinks}\ (\exists\ \mathsf{left}))\ \mathsf{alice})); \\ \lfloor b \rfloor \rightsquigarrow_\beta \lambda c.(\exists\ \lambda y.(c\ ((\mathsf{thinks}\ (\mathsf{left}\ y))\ \mathsf{alice}))). \end{array} \tag{26}$$

6 Conclusions, Further Directions

In this paper we have moved from asymmetric Lambek calculus with a single succedent formula to the symmetric Lambek-Grishin calculus, where both the antecedent and the succedent are formula structures, configured in terms of $\otimes$ and $\oplus$ respectively, and where the $\otimes$ and $\oplus$ environments can interact in a structure-preserving way. This move makes it possible to import into the field of natural language semantics the powerful tools of $\lambda\mu$-calculus. The main attraction of the proposed continuation semantics, in our view, lies in the fact that **LG** allows us to fully exploit the duality between Lambek's directional $/, \backslash$ implications and the corresponding directional $\oslash, \oslash$ difference operations, at the level of syntax and at the level of semantics. We thus restore Curry's original idea of an *isomorphism* between proofs and terms, rather than the weaker homomorphic view of standard Lambek (or Montagovian) semantics.

Our approach differs in a number of respects from the related work cited in §1. Abstracting away from the directionality issue, de Groote's original application of $\lambda\mu$ calculus to scope construal syntactically types generalized quantifier phrases as np with meaning representation $\mu\alpha^{K_e}(\exists\ \alpha)$. As a result, a sentence with multiple GQ phrases is associated with a *unique* parse/term; the multiple readings for that term are obtained as a result of the non-confluence of $\lambda\mu$ calculus, which is considered as a feature, not a bug. Our approach in contrast is true to the principle that multiple readings can only arise as the images of distinct proofs, given the Curry-Howard isomorphism between proofs and terms. Barker [19,11] uses a simplified continuation semantics, which lifts types A to $(A \to R) \to R$ 'externally', without applying the CPS transformation to the internal structure of complex types. This breaks the symmetry which is at the heart of our dual treatment of $/, \backslash$ vs $\oslash, \oslash$. The structural-rule account of scope flexibility in [12,13] suffers from commutativity problems.

The approach described here, like Hendriks's type-shifting approach, creates all combinatorial possibilities for scope construal. However, it is well known that, depending on the choice of particular lexical items, many of these construals will in fact be unavailable. Bernardi [20] uses the control modalities $\Diamond, \Box$ to calibrate the scopal behaviour of particular classes of GQ expressions. Adding $\Diamond, \Box$ and a pair of dually residuated modalities to **LG** is straightforward. In a follow-up paper, we plan to study the continuation semantics of these operations, relating them to the `shift` and `reset` constructs one finds in the theory of functional programming languages and that have been considered in [11,13].

Finally, the interpretation given here construes scopal ambiguities in a *static* setting. In a recent paper, de Groote [21] develops a continuation-based approach towards *dynamic* interpretation. A natural topic for further research would be to investigate how to incorporate this dynamic perspective in our setting, and how to extend the approach of [21] with the difference operations and the concomitant Grishin interaction principles.

References

1. Lambek, J.: The mathematics of sentence structure. American Mathematical Monthly 65, 154–170 (1958)
2. Lambek, J.: On the calculus of syntactic types. In: Jakobson, R. (ed.) Structure of Language and Its Mathematical Aspects. American Mathematical Society, 166–178 (1961)
3. Grishin, V.: On a generalization of the Ajdukiewicz-Lambek system. In: Studies in Nonclassical Logics and Formal Systems. Nauka, Moscow, pp. 315–334 [English translation in Abrusci and Casadio (eds.) Proceedings 5th Roma Workshop, Bulzoni Editore, Roma, 2002] (1983)
4. Curien, P., Herbelin, H.: Duality of computation. In: International Conference on Functional Programming (ICFP'00), pp. 233–243 [2005: corrected version] (2000)
5. Moortgat, M.: Generalized quantifiers and discontinuous type constructors. In: Bunt, H., van Horck, A. (eds.) Discontinuous Constituency, pp. 181–207. Walter de Gruyter, Berlin (1996)
6. Hendriks, H.: Studied flexibility. Categories and types in syntax and semantics. PhD thesis, ILLC, Amsterdam University (1993)
7. Morrill, G.: Discontinuity in categorial grammar. Linguistics and Philosophy. 18, 175–219 (1995)
8. Morrill, G., Fadda, M., Valentin, O.: Nondeterministic discontinuous Lambek calculus. In: Proceedings of the Seventh International Workshop on Computational Semantics (IWCS7), Tilburg (2007)
9. Lambek, J.: From categorial to bilinear logic. In: Schröder-Heister, K.D.P. (ed.) Substructural Logics, pp. 207–237. Oxford University Press, Oxford (1993)
10. de Groote, P.: Type raising, continuations, and classical logic. In: van Rooy, R., (ed.) Proceedings of the Thirteenth Amsterdam Colloquium, ILLC, Universiteit van Amsterdam, pp. 97–101 (2001)
11. Barker, C.: Continuations in natural language. In: Thielecke, H. (ed.) CW'04: Proceedings of the 4th ACM SIGPLAN continuations workshop, Tech. Rep. CSR-04-1, School of Computer Science, University of Birmingham, pp. 1–11 (2004)

12. Barker, C., Shan, C.: Types as graphs: Continuations in type logical grammar. Language and Information 15(4), 331–370 (2006)
13. Shan, C.: Linguistic side effects. PhD thesis, Harvard University (2005)
14. Selinger, P.: Control categories and duality: on the categorical semantics of the lambda-mu calculus. Math. Struct. in Comp. Science 11, 207–260 (2001)
15. Wadler, P.: Call-by-value is dual to call-by-name. In: ICFP, Uppsala, Sweden (August 2003)
16. Moortgat, M.: Symmetries in natural language syntax and semantics: the Lambek-Grishin calculus (this volume). In: Leivant, D., de Queiros, R. (eds.) WoLLIC'07. Proceedings 14th Workshop on Logic, Language, Information and Computation. LNCS, vol. 4576, Springer, Heidelberg (2007)
17. Goré, R.: Substructural logics on display. Logic Journal of IGPL 6(3), 451–504 (1997)
18. de Groote, P., Lamarche, F.: Classical non-associative Lambek calculus. Studia Logica 71, 335–388 (2002)
19. Barker, C.: Continuations and the nature of quantification. Natural language semantics 10, 211–242 (2002)
20. Bernardi, R.: Reasoning with Polarity in Categorial Type Logic. PhD thesis, Utrecht Institute of Linguistics OTS (2002)
21. de Groote, P.: Towards a Montagovian account of dynamics. In: Proceedings SALT 16, CLC Publications (2006)

A Shorthand Format for Sequent Derivations

As an example of the **LG** term assignment, (27) gives the derivation showing how one obtains a Lambek-style GQ type $C/(A\backslash B)$ from a $(B \oslash C) \otimes A$ source.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{z : A \xrightarrow{z} A \quad B \xrightarrow{\gamma} \gamma : B}{z : A \circ A\backslash B \xrightarrow{z \ltimes \gamma} \gamma : B}(\backslash L)}{C \xrightarrow{\alpha} \alpha : C \quad z : A \circ y : A\backslash B \xrightarrow{\mu\gamma.(y * (z \ltimes \gamma))} B}(\leftrightharpoons)}{z : A \circ y : A\backslash B \xrightarrow{(\mu\gamma.(y * (z \ltimes \gamma))) \succ \alpha} B \oslash C \circ \alpha : C}(\oslash R)}{A \circ y : A\backslash B \xrightarrow{\tilde{\mu}z.((\mu\gamma.(y * (z \ltimes \gamma))) \succ \alpha) * \beta)} \beta : B \oslash C \circ \alpha : C}(\leftrightharpoons)}{(B \oslash C) \otimes A \circ y : A\backslash B \xrightarrow{\beta \otimes (\tilde{\mu}z.((\mu\gamma.(y * (z \ltimes \gamma))) \succ \alpha) * \beta))} \alpha : C}(\otimes L)}{x : (B \oslash C) \otimes A \circ y : A\backslash B \xrightarrow{\mu\alpha.(x * (\beta \otimes (\tilde{\mu}z.((\mu\gamma.(y * (z \ltimes \gamma))) \succ \alpha) * \beta)))} C}(\leftrightharpoons)}{x : (B \oslash C) \otimes A \xrightarrow{(\mu\alpha.(x * (\beta \otimes (\tilde{\mu}z.((\mu\gamma.(y * (z \ltimes \gamma))) \succ \alpha) * \beta)))) \,/\, y} C/(A\backslash B)}(/R) \tag{27}$$

This example shows that except for the most simple derivations, the sequent tree format soon becomes unwieldy. Below we introduce a more user-friendly line format, which graphically highlights the tree structure of the antecedent and succedent parts. In the line format, each row has (a) a line number, (b) the (co)term for the currently active formula, and the antecedent (c) and succedent (d) structures in tree format. The cursor singles out the currently active formula. It takes the form $\cdot^{\bullet}$ in the antecedent, and $\cdot^{\circ}$ in the succedent. With $\boxed{n}$ we refer to the (co)term at line n. Compare (27) with the derivation for a sentence 'Somebody left' in line format.

LINE	(CO)TERM	LHS TREE	RHS TREE
1.	x	np | x	np°
2.	α	$s^\bullet$	s | α
3.	$\boxed{1} \ltimes \boxed{2}$	$np\ (np\backslash s)^\bullet$ | x	s | α
4.	$\mu\alpha.(\mathsf{left} * \boxed{3})$	$np\ (np\backslash s)$ | | x left	s°
5.	β	$s^\bullet$	s | β
6.	$\boxed{4} \succ \boxed{5}$	$np\ (np\backslash s)$ | | x left	$(s \oslash s)^\circ\ s$ | β
7.	$\widetilde{\mu}x.(\boxed{6} * \gamma)$	$np^\bullet\ (np\backslash s)$ | left	$(s \oslash s)\ s$ | | γ β
8.	$\gamma \oslash \boxed{7}$	$((s \oslash s) \oslash np)^\bullet\ (np\backslash s)$ | left	s | β
9.	$\mu\beta.(\mathsf{somebody} * \boxed{8})$	$((s \oslash s) \oslash np)\ (np\backslash s)$ | | somebody left	s°

We reduce the length of a derivation further using the (qL) and (qR) rules of inference discussed in the main text and *folding*: a sequence of n $(/, \backslash R)$ (respectively $(\oslash, \oslash L)$) one-premise inferences is folded into a one-step one-premise inference; a $(\leftrightharpoons)$ step (or $(\rightleftharpoons)$ respectively) followed by a sequence of n $(/, \backslash L)$ (respectively $(\oslash, \oslash R)$) two-premise inferences is folded into a one-step $n+1$ premise inference; an n premise inference with m non-lexical axiom premises is contracted to an $n-m$ premise inference. Where the succedent (antecedent) tree is just a point, we write the highlighted formula as the root of the antecedent (succedent) tree. The result of applying these conventions for the example sentence 'somebody left' is (19) in the main text.