Information and Computation 208 (2010) 397–416

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Continuation semantics for the Lambek–Grishin calculus[☆]

Raffaella Bernardi[a,*], Michael Moortgat[b]

[a] *Faculty of Computer Science, Free University of Bozen-Bolzano, P.zza Domenicani 3, 39100, Bolzano, Italy*
[b] *Department of Linguistics OTS, Trans 10, 3512 JK, Utrecht, The Netherlands*

ARTICLE INFO

Article history:
Received 6 January 2008
Revised 22 May 2009
Available online 4 December 2009

ABSTRACT

Categorial grammars in the tradition of Lambek 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. 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** derivations, based on Curien and Herbelin's lambda mu comu calculus. We discuss continuation-passing-style (CPS) translations mapping **LG** derivations to proofs/terms of Intuitionistic Multiplicative Linear Logic — the categorial system **LP** which serves as the logic for natural language meaning assembly. We show how LG, thus interpreted, associates sentences with quantifier phrases with the appropriate range of meanings, thus overcoming the expressive limitations of asymmetric categorial grammars in this area.

1. Background

In two groundbreaking papers written some 50 years ago [17,18], Jim Lambek initiated the 'parsing-as-deduction' style of linguistic analysis by setting himself the task of finding "an effective rule (or algorithm) for distinguishing sentences from nonsentences" which would work "not only for the formal languages of interest to the mathematical logician, but also for natural languages". The method consists in assigning types to the words of the language under investigation "in such a way that the grammatical correctness of a sentence can be determined by a computation on these types". The vocabulary of the type system has a small set of atomic types, and operations of multiplication, left and right division.

$$\begin{array}{lll} A, B & ::= & p \qquad\qquad \text{atoms: } s \text{ (sentence)}, np \text{ (noun phrase)}, \ldots \\ & & \mid A \otimes B \mid A\backslash B \mid B/A \qquad \text{product, left division, right division} \end{array} \tag{1}$$

[☆] The material in this paper was presented in a course at the European Summer School in Logic, Language and Information, Dublin 2007. We thank the participants for their feedback. Special thanks to Peter Selinger for helpful discussion on the duality between call-by-value and call-by-name, to Philippe de Groote for bringing Curien and Herbelin's work to our attention, and to Chris Barker and Ken Shan for sharing their views on continuation semantics with us. The paper benefited from the comments of two anonymous referees. All remaining errors are our own.

[*] Corresponding author.
E-mail addresses: bernardi@inf.unibz.it (R. Bernardi), M.J.Moortgat@uu.nl (M. Moortgat).

0890-5401/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2009.11.005

$$\frac{}{A \Rightarrow A}\ (\mathrm{Ax}) \qquad \frac{\Delta \Rightarrow A \quad \Gamma[A] \Rightarrow B}{\Gamma[\Delta] \Rightarrow B}\ (\mathrm{Cut})$$

$$\frac{\Delta \Rightarrow A \quad \Gamma[B] \Rightarrow C}{\Gamma[\Delta \circ (A\backslash B)] \Rightarrow C}\ (\backslash L) \qquad \frac{A \circ \Gamma \Rightarrow B}{\Gamma \Rightarrow A\backslash B}\ (\backslash R)$$

$$\frac{\Delta \Rightarrow A \quad \Gamma[B] \Rightarrow C}{\Gamma[(B/A) \circ \Delta] \Rightarrow C}\ (/L) \qquad \frac{\Gamma \circ A \Rightarrow B}{\Gamma \Rightarrow B/A}\ (/R)$$

$$\frac{\Gamma[A \circ B] \Rightarrow C}{\Gamma[A \otimes B] \Rightarrow C}\ (\otimes L) \qquad \frac{\Gamma \Rightarrow A \quad \Delta \Rightarrow B}{\Gamma \circ \Delta \Rightarrow A \otimes B}\ (\otimes R)$$

Fig. 1. Sequent presentation of **NL**, the Syntactic Calculus of Lambek (1961).

For computations on these types, Lambek proposes a Syntactic Calculus. The minimal version of this calculus is known as **NL**. It is presented in the [18] paper, and consists of the preorder laws for derivability ($A \vdash A$ plus transitivity: from $A \vdash B$ and $B \vdash C$, derive $A \vdash C$) together with the residuation laws of (2) establishing the relation between the product operation and the two divisions.

$$A \vdash C/B \quad \text{iff} \quad A \otimes B \vdash C \quad \text{iff} \quad B \vdash A\backslash C \qquad \text{residuation laws} \tag{2}$$

To obtain a decision algorithm for sentencehood, Lambek reformulates the Syntactic Calculus in the Gentzen sequent format of Fig. 1, obtaining what is in effect a logic without *any* structural rules: grammatical material cannot be duplicated or erased without affecting well formedness (absence of Contraction and Weakening); moreover, structural rules affecting word order and constituent structure (Commutativity and Associativity) are unavailable. Sequents are of the form $\Gamma \Rightarrow A$ with A a formula from (1) and Γ a (non-empty) tree with formulas at the yield; $\circ$ is the tree-building operation corresponding to the product connective $\otimes$. $\Gamma[\Delta]$ stands for an antecedent tree structure Γ with a distinguished substructure Δ. The cut rule can be shown to be admissible; backwards chaining cut-free proof search then immediately produces a decision procedure for theoremhood.

From the minimal system **NL** a number of extensions can be obtained by relaxing the sensitivity for word order and/or constituent structure: the system **L** [17] results from the addition of a structural rule of Associativity; the Lambek–van Benthem calculus **LP** [29] adds both Associativity and Commutativity. The latter system, in retrospect, can be seen to coincide with Intuitionistic Multiplicative Linear Logic (without units).

Semantics for **NL** and its relatives can be studied from a variety of perspectives. In this paper, we look at a structural and a computational interpretation. For the *structural* semantics we adopt the approach originally developed within the context of relevance logic. Models, in this interpretation, are based on frames $(W, R_\otimes)$, where W is the set of 'grammatical resources', and $R_\otimes$ a ternary relation of grammatical composition — the fusion relation of relevance logic, or the 'Merge' relation in the vocabulary of generative grammar. The valuation V assigns subsets of W to the type formulas, respecting the truth conditions of (3) for complex formulas.

$$\begin{array}{l} x \Vdash A \otimes B \text{ iff } \exists yz.R_\otimes xyz \text{ and } y \Vdash A \text{ and } z \Vdash B \\ y \Vdash C/B \text{ iff } \forall xz.(R_\otimes xyz \text{ and } z \Vdash B) \text{ implies } x \Vdash C \\ z \Vdash A\backslash C \text{ iff } \forall xy.(R_\otimes xyz \text{ and } y \Vdash A) \text{ implies } x \Vdash C \end{array} \tag{3}$$

The basic soundness/completeness result [11] then states that $A \vdash B$ is provable in **NL** iff $V(A) \subseteq V(B)$ for all frames F and valuations V. The theorems of **NL**, in this sense, capture grammatical *invariants*: principles of grammatical organization that hold no matter what the properties of the Merge relation are. For the extensions with Associativity and/or Commutativity, frame constraints corresponding to these structural rules restrict the interpretation of the Merge relation.

The second type of interpretation is a *computational* one, along the lines of the Curry–Howard formulas-as-types program. Under this interpretation, originally introduced in [29], categorial derivations are associated with λ terms, representing instructions for meaning assembly. The problem of parsing a sequence of words $w_1, \ldots, w_n$ with lexical types $A_1, \ldots, A_n$ as a phrase of type B is now represented by sequents of the form of (4), where the yield of the antecedent structure Γ are formulas A_i labeled with distinct variables x_i. The derivation showing that Γ is a well-formed structure of type B is associated with a lambda term t: a program with parameters $x_1, \ldots, x_n$ indicating how the meaning of the conclusion B is assembled out of the meanings of the constituent parts A_i.

$$\underbrace{\begin{array}{ccc} w_1 & \cdots & w_n \\ \vdots & & \vdots \\ x_1 : A_1 & \cdots & x_n : A_n \end{array}}_{\Gamma} \quad \Rightarrow \quad t : B \tag{4}$$

In [29], meaning composition is studied from the perspective of **LP**: the class of terms λ**LP** in one-to-one correspondence with the derivations in this logic are the *linear* λ terms, with exactly the antecedent x_i as free variables. For the syntactically more sensitive systems **(N)L**, the interpretation takes the form of a *translation* $(\cdot)'$ mapping the source logic types and proofs to types/proofs of the target logic **LP**. At the type level, we stipulate $p' = \widetilde{p}$ for the atoms of the source logic, for example, $np' = e$ and $s' = t$, with e and t target logic types denoting individuals and truth values, respectively. For complex types $(A/B)' = (B\backslash A)' = B' \to A'$,[1] where $\to$ is the non-directional linear implication of **LP**. At the level of proofs, **(N)L** sequents $A_1, \ldots, A_n \Rightarrow B$ have their antecedent formulas A_i labeled with distinct variables x_i of type A_i', the image in **LP** of the source logic types A_i. Out of these variables, an λ**LP** term t of type B' is computed by means of the rules in (5). We give the $(\backslash L)$ and $(\backslash R)$ rules; the $(/L)$ and $(/R)$ rules have the same term annotation since $(A/B)' = (B\backslash A)'$.

$$\frac{}{x : A \Rightarrow x : A}\ (\text{Ax}) \qquad \frac{\Delta \Rightarrow t : A \quad \Gamma[x : A] \Rightarrow u : B}{\Gamma[\Delta] \Rightarrow u[x := t] : B}\ (\text{Cut})$$

$$\frac{\Delta \Rightarrow t : A \quad \Gamma[x : B] \Rightarrow u : C}{\Gamma[\Delta \circ y : (A\backslash B)] \Rightarrow u[x := (y\, t)] : C}\ (\backslash L) \qquad \frac{x : A \circ \Gamma \Rightarrow t : B}{\Gamma \Rightarrow \lambda x.t : A\backslash B}\ (\backslash R) \tag{5}$$

1.1. Expressive limitations

The semantic expressivity of **(N)L** is inversely proportional to the syntactic discrimination of these systems, in the sense that $\mathbf{NL}' \subset \mathbf{L}' \subset \lambda\mathbf{LP}$: dropping Associativity and/or Commutativity means that certain recipes for meaning assembly that are available at the level of λ**LP** can no longer be obtained as the $(\cdot)'$ image of **(N)L** derivations. The expressive limitations of **(N)L** make themselves clearly felt in the area of natural language semantics that deals with quantifier phrases (QPs) — phrases such as 'nobody', 'every linguist', 'a logician', 'a typo' in the examples below. Model-theoretically, these phrases denote sets of sets of individuals, the interpretation for type $(e \to t) \to t$ in the target logic. Proof-theoretically, they behave as in (7).

a *Nobody* stayed.
b *Every linguist* met A LOGICIAN (in Amsterdam). (6)
c Jim saw that *a typo* occurred on EVERY PAGE.

$$\frac{\Delta[x : np] \Rightarrow t : s_1 \qquad \Gamma[y : s_2] \Rightarrow u : s_3}{\Gamma[\Delta[z : QP]] \Rightarrow u[y := (z\ \lambda x.t)] : s_3} \tag{7}$$

We can paraphrase the inference in (7) as follows. A QP acts locally as a simple *np* within an enveloping sentential context Δ, the scope *domain* of the QP. In (7) we label the scope domain as s_1, to distinguish it from s_2. The recipe associated with s_2 is the result of applying z, the parameter for the QP meaning, to a term denoting (the characteristic function of) a set of individuals. This term is obtained by abstracting over the *np* hypothesis in the scope domain s_1.

Matching rule (7) with the examples can be non-deterministic: in (b) and (c), there are different choices for identifying the scope domain Δ, maybe within a broader context Γ, leading to scope *ambiguities* in the interpretation. For (b), do we have to check whether $\{x \mid \text{every linguist met } x\} \in [\text{a logician}]$, or whether $\{x \mid x \text{ met a logician}\} \in [\text{every linguist}]$? For (c), does its interpretation involve a check whether $\{x \mid x \text{ occurred on every page}\} \in [\text{a typo}]$, or whether $\{x \mid \text{Jim saw } x \text{ occurred on every page}\} \in [\text{a typo}]$?

The challenge is to find a type for QP in **(N)L** that would behave as indicated in (7). The solution $s_2/(np\backslash s_1)$ restricts the QP to occur in subject positions, italicized in (6), and does not cover the occurrences in small caps. In **NL**, there is no type for the small caps occurrences such that its $(\cdot)'$ image would be $(e \to t) \to t$. In **L**, these occurrences would be covered by $(s_1/np)\backslash s_2$, but that type allows a QP to take scope only when it occupies the right periphery of its domain. Similarly, $s_2/(np\backslash s_1)$ is restricted to take scope from a left-peripheral position. That means that **L** cannot associate (c) with a reading where 'a typo' has scope at the main clause level ('there is a typo x such that Jim thinks x occurred on every page'), and that in (b), 'a logician' cannot outscope the adverb 'in Amsterdam'.

Summarizing the above, we identify the following problems in associating **(N)L** derivations involving QPs with λ**LP** terms coding their meaning.

Type uniformity. In **(N)L**, different syntactic occurrences of QPs require different type assignments. Instead, we would like to have a single type assignment in line with the uniform semantic contribution of the QPs.

Scope flexibility. In **(N)L**, the possibilities for scope construal in environments with multiple QPs are limited by peripherality conditions. Instead, we would like to realize the derivational non-determinism of (7) in full generality.

What we say here about the behavior of QPs applies to a wider range of phenomena known in the linguistic literature as *in situ* binding; in general, these phenomena involve the binding of an A-type hypothesis in a domain of type B, producing a C-type result. In the type-logical literature, various analyses of *in situ* phenomena have been proposed that address the expressivity issues discussed above. Solutions in the tradition of Multimodal TLG [4,24,25] postulate a richer inventory of syntactic operations to put phrases together, with *wrapping* operations in addition to the usual concatenation. Flexible Montague Grammar [15] relaxes the mapping from the syntactic source logic to **LP**. In this approach $(\cdot)'$ is modeled as a relation instead of a function; a source logic type is associated with a *set* of **LP** types, related by type-shifting postulates.

The approach we will develop below differs from the above in sticking to the *minimal* categorial logic: the pure logic of residuation **NL**. We overcome its expressive limitations by dropping the restriction that the succedent must consist of a single

[1] We restrict our attention to the interpretation for the division types /, \ in this paper.

$$\begin{tikzcd}
\lambda\mathbf{LG} \arrow[r, "cbv"] \arrow[d, "cbn"'] & \lceil\tau\rceil : \lceil A\rceil \arrow[d, "\llbracket\cdot\rrbracket"] \arrow[dl, dotted, leftrightarrow, ".^{\infty}" description] \\
\lfloor\tau\rfloor : \lfloor A\rfloor \arrow[r, "\lfloor\!\lfloor\cdot\rfloor\!\rfloor"'] & \lambda\mathbf{LP}
\end{tikzcd}$$

Fig. 2. Outline of the paper.

formula and moving to **LG**: a symmetric system where the Lambek connectives (product, left and right division) coexist with a dual family: coproduct ($\oplus$), right ($\oslash$) and left ($\obslash$) difference. The communication between these two families is expressed in terms of the distributivity principles of [14].

1.2. Outline of the paper

Fig. 2 schematically presents the outline of the sections that follow. In Section 2, we present the syntax and relational semantics of **LG** and discuss its symmetries at the level of types and proofs. In Section 3.1, we present a term language λ**LG** which is the Curry–Howard image of **LG** derivations. In Section 3.2, we then study the mapping from λ**LG** to our target logic for meaning composition λ**LP**. The mapping takes the form of a continuation-passing-style (CPS) translation, which can be executed with a call-by-value $\lceil\cdot\rceil$ or a call-by-name $\lfloor\cdot\rfloor$ strategy, related by duality. In Section 4 we turn to the discussion of scope construal. We show how the interpretation of lexical constants with simple **NL** types can be systematically lifted to the CPS level in terms of mappings $\llbracket\cdot\rrbracket$, $\lfloor\!\lfloor\cdot\rfloor\!\rfloor$. We propose $(s \oslash s) \obslash np$ as **LG** type assignment for QPs and show how it solves the problems we identified above with scope construal in the asymmetric systems **(N)L**. In the concluding section, we point to some directions for future work.

1.3. Relation to previous work

Lambek [19] 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: stronger in assuming hard-wired Associativity for $\otimes$, $\oplus$; weaker in that our proposal uses a more comprehensive set of distributivity principles. Continuations were put on the linguistic agenda in [2] and [10]; the latter places the discussion explicitly in the context of the $\lambda\mu$ calculus. Barker and Shan, in a series of papers, [3–5,28] among others, have worked out continuation-based analyses for a variety of semantic phenomena. We comment on the relation of our proposal to theirs in Section 5. Duality between the call-by-value and call-by-name evaluation strategies has been obtained in [9,27,31], 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. Lambek–Grishin calculus

In a remarkable paper written in 1983, Grishin [14] has proposed a framework for systematically generalizing the Lambek calculus. The generalization has two components. First of all, the vocabulary of type-forming operations is made symmetric: see the extended set of formulas in (8), where in addition to the familiar $\otimes, \backslash, /$ (product, left and right division), one finds a dual family $\oplus$, $\oslash$, $\obslash$: coproduct, right and left difference.[2] The second component of the generalization consists in adding distributivity principles for the interaction between the $\otimes$ and the $\oplus$ families. We discuss these two components in turn.

$$\begin{array}{lll l} A, B & ::= & p \mid & \text{atoms: } s \text{ sentence, } np \text{ noun phrase, } \ldots \\ & & A \otimes B \mid B\backslash A \mid A/B \mid & \text{product, left versus right division} \\ & & A \oplus B \mid A \oslash B \mid B \obslash A & \text{coproduct, right versus left difference} \end{array} \tag{8}$$

2.1. Symmetry

The minimal symmetric categorial grammar (which we will refer to as $\mathbf{LG}_{\emptyset}$) is given by the preorder laws for the derivability relation, together with the residuation and dual residuation principles of (10).

$$A \vdash A; \qquad \text{from } A \vdash B \text{ and } B \vdash C \text{ infer } A \vdash C \qquad \text{preorder} \tag{9}$$

[2] A little pronunciation dictionary: read $B\backslash A$ as 'B under A', A/B as 'A over B', $B \obslash A$ as 'B from A' and $A \oslash B$ as 'A less B'.

$\sigma \setminus \rho$	$?\otimes$	$\otimes ?$	$\obslash ?$	$?\oslash$
$\obslash ?$	$Q1$	$Q4$		
$?\oslash$	$Q2$	$Q3$		
$?\otimes$			$P1$	$P4$
$\otimes ?$			$P2$	$P3$

Fig. 3. Structure-preserving interaction postulates $A^{\sigma}B^{\rho}C \vdash B^{\rho}A^{\sigma}C$.

$$\begin{array}{lll} A \vdash C/B \text{ iff } A \otimes B \vdash C \text{ iff } B \vdash A\backslash C & \text{residuated triple} \\ B \obslash C \vdash A \text{ iff } C \vdash B \oplus A \text{ iff } C \oslash A \vdash B & \text{dual residuated triple} \end{array} \tag{10}$$

The residuation patterns give rise to two kinds of symmetry, which we write $\cdot^{\bowtie}$ and $\cdot^{\infty}$. For atomic types p, $p^{\bowtie} = p = p^{\infty}$. For complex types the definitions are given by the bidirectional translation tables in (11). The tables succinctly represent a list of definitional equations $(C/D)^{\bowtie} = D^{\bowtie}\backslash C^{\bowtie}$, $(D\backslash C)^{\bowtie} = C^{\bowtie}/D^{\bowtie}, \dots$

$$\bowtie \quad \frac{C/D \quad A\otimes B \quad B\oplus A \quad D\obslash C}{D\backslash C \quad B\otimes A \quad A\oplus B \quad C\oslash D} \qquad\qquad \infty \quad \frac{C/B \quad A\otimes B \quad A\backslash C}{B\obslash C \quad B\oplus A \quad C\oslash A} \tag{11}$$

The two symmetries and their composition (in either order) are involutive operations; moreover we have $A^{\bowtie\infty\bowtie} = A^{\infty}$ and $A^{\infty\bowtie\infty} = A^{\bowtie}$. Together with the identity, then, $\cdot^{\bowtie}$, $\cdot^{\infty}$ and their composition constitute Klein's four-group, the smallest non-cyclic Abelian group. With respect to the derivability relation, $\cdot^{\bowtie}$ is order-preserving, $\cdot^{\infty}$ is order-reversing

$$A^{\bowtie} \vdash B^{\bowtie} \text{ iff } A \vdash B \text{ iff } B^{\infty} \vdash A^{\infty} \tag{12}$$

The connections between residuation theory and substructural logics have been well studied; see [12] for a recent survey. We review some properties that are useful for an understanding of the rest of the paper. Following the notation of [14], for connectives $* \in \{/, \otimes, \backslash, \obslash, \oplus, \oslash\}$, we write $A^{?\,*}B$ for $B*A$ and $A^{*\,?}B$ for $A*B$.

2.2. Compositions

Consider the operations $?\otimes$ and $?\backslash$, i.e. multiplication to the left and left division. The composition $(?\otimes)(?\backslash)$ is contracting: it yields the familiar application law $A \otimes (A\backslash B) \vdash B$. The composition $(?\backslash)(?\otimes)$ is expanding: $B \vdash A\backslash(A \otimes B)$. Together with the $\cdot^{\bowtie}$ and $\cdot^{\infty}$ symmetries, we obtain the patterns in (13). The columns of (13) are related by $\cdot^{\infty}$; the rows by $\cdot^{\bowtie}$.

$$\begin{array}{ll} A \otimes (A\backslash B) \vdash B \vdash A\backslash(A \otimes B) & (B/A) \otimes A \vdash B \vdash (B \otimes A)/A \\ (B \oplus A) \oslash A \vdash B \vdash (B \oslash A) \oplus A & A \obslash (A \oplus B) \vdash B \vdash A \oplus (A \obslash B) \end{array} \tag{13}$$

2.3. Monotonicity

The operations $\sigma \in \{?\otimes, \otimes ?, ?\oplus, \oplus ?, ?/, \backslash ?, ?\oslash, \obslash ?\}$ are isotone with respect to the derivability relation; the operations $\rho \in \{/?, ?\backslash, \oslash ?, ?\obslash\}$ are antitone; i.e. we have the following monotonicity inferences.

$$\frac{A \vdash B}{C^{\sigma}A \vdash C^{\sigma}B} \qquad \frac{A \vdash B}{C^{\rho}B \vdash C^{\rho}A} \tag{14}$$

Given the preorder laws (9) and the (dual) residuation principles (10), one easily derives (13) and the inference rules in (14). Conversely, given (13) and (14), one can derive (10).

2.4. Distributivity principles

The minimal symmetric system $\mathbf{LG}_{\emptyset}$ by itself does not offer us the kind of expressivity needed to address the problems discussed in Section 1. The attraction of Grishin's work derives from the fact that he develops a systematic schema for extending the minimal symmetric system by means of extra postulates. Combinatorially, there is a total of sixteen such extensions, configured in groups of four.

The 16 cells of the matrix of Fig. 3 represent inequalities of the form $A^{\sigma}B^{\rho}C \leq B^{\rho}A^{\sigma}C$, with σ and ρ taken from the set $\{?\otimes, \otimes ?, \obslash ?, ?\oslash\}$. Eight of these choices pick operations from the same family: they give rise to same-sort Associativity and Commutativity principles for $\otimes$ and $\oplus$; we have left these cells blank as they are of no concern for our purposes. The remaining eight are *interaction* principles that relate the $\otimes$ and $\oplus$ families. They are referred to as *weak* (or linear) distributivity principles in [8], because no material is copied (contrast $a \times (b + c) = (a \times b) + (a \times c)$ in arithmetic). We call them *structure-preserving* distributivities, because they leave intact the linear order and bracketing structure information encoded in our non-associative, non-commutative type-forming operations.

$$\begin{tikzcd}
 & \mathbf{LG}_\emptyset + (18) + (15) & & \\
\mathbf{LG}_\emptyset + (18) \arrow[ur] & & \mathbf{LG}_\emptyset + (15) \arrow[ul] & (= \mathbf{LG}) \\
 & \mathbf{LG}_\emptyset \arrow[ul] \arrow[ur] & &
\end{tikzcd}$$

Fig. 4. The Lambek–Grishin hierarchy.

Consider first the group of postulates $P1$–$P4$. In (15) we spell out the inequality $A^\sigma B^\rho C \vdash B^\rho A^\sigma C$ for these combinations. On the left side of the turnstile, one finds a $\otimes$ formula with a difference formula ($B \obslash C$ or $C \oslash B$) as its first or second coordinate. The Grishin principles rewrite this configuration in such a way that the difference operations $\obslash$, $\oslash$ become the main connective. The four instances in (15) are obtained by closing the choice of the operations σ, ρ under the $\cdot^{\bowtie}$ symmetry.[3]

$$\begin{array}{llll}
(P1) & (B \obslash C) \otimes A \vdash B \obslash (C \otimes A) & A \otimes (C \oslash B) \vdash (A \otimes C) \oslash B & (P3) \\
(P2) & A \otimes (B \obslash C) \vdash B \obslash (A \otimes C) & (C \oslash B) \otimes A \vdash (C \otimes A) \oslash B & (P4)
\end{array} \qquad (15)$$

To see the potential usefulness of these postulates for the linguistic applications we have in mind, suppose the lexicon contains a word with a type-assignment that has one of the difference operations as its main connective, say $B \obslash C$. Without interaction principles, when we use this word in building a phrase, $\obslash$ would be *trapped* in its $\otimes$ context:

$$A_1 \otimes \cdots A_i \otimes (B \obslash C) \otimes A_{i+2} \cdots A_n \vdash D$$

By repeated application of $P1$ and $P2$, we can move the $B^{\obslash ?}$ component upward through the $\otimes$ context until it becomes the main connective. At that point, the dual residuation principle is applicable, allowing B to move to the right side of the turnstile.

The images of (15) under $\cdot^\infty$ are given in (16). Their role is dual: on the right side of the turnstile, they can rewrite a configuration where a left or right slash is trapped within a $\oplus$ context into a configuration where the B subformula can be shifted to the left side by means of the residuation principles.

$$\begin{array}{llll}
(P1') & (A \oplus C)/B \vdash A \oplus (C/B) & B\backslash(C \oplus A) \vdash (B\backslash C) \oplus A & (P3') \\
(P2') & (C \oplus A)/B \vdash (C/B) \oplus A & B\backslash(A \oplus C) \vdash A \oplus (B\backslash C) & (P4')
\end{array} \qquad (16)$$

One easily checks that the forms in (16) and those in (15) are interderivable. In (17) on the left, $P1'$ is derived from $P1$; on the right $P1$ is derived from $P1'$. The proofs are related by the $\cdot^\infty$ Symmetry.

$$\begin{array}{c}
(A \oplus C)/B \vdash (A \oplus C)/B \\ \hline
((A \oplus C)/B) \otimes B \vdash A \oplus C \\ \hline
A \obslash (((A \oplus C)/B) \otimes B) \vdash C \\ \hline
(A \obslash ((A \oplus C)/B)) \otimes B \vdash C \\ \hline
A \obslash ((A \oplus C)/B) \vdash C/B \\ \hline
(A \oplus C)/B \vdash A \oplus (C/B)
\end{array}\ P1 \quad \overset{\infty}{\longleftrightarrow} \quad
\begin{array}{c}
B \obslash (C \otimes A) \vdash B \obslash (C \otimes A) \\ \hline
C \otimes A \vdash B \oplus ((B \obslash (C \otimes A)) \\ \hline
C \vdash (B \oplus ((B \obslash (C \otimes A)))/A \\ \hline
C \vdash B \oplus ((B \obslash (C \otimes A))/A) \\ \hline
B \obslash C \vdash (B \obslash (C \otimes A))/A \\ \hline
(B \obslash C) \otimes A \vdash B \obslash (C \otimes A)
\end{array}\ P1' \qquad (17)$$

The alternative option for interaction discussed by Grishin is given by the group of postulates $Q1$–$Q4$. The realization of the schema $A^\sigma B^\rho C \vdash B^\rho A^\sigma C$ is given in (18). Notice that, abstracting from the propositional variables used, $Q1$–$Q4$ and $P1$–$P4$ are converses of each other: whereas the $P1$–$P4$ postulates *lift* an embedded difference operation ($\obslash ?$ or $? \oslash$) out of a $\otimes$ context, the $Q1$–$Q4$ postulates *lower* the difference operation into the $\otimes$ context.

$$\begin{array}{llll}
(Q1) & A \obslash (C \otimes B) \vdash (A \obslash C) \otimes B & (B \otimes C) \oslash A \vdash B \otimes (C \oslash A) & (Q3) \\
(Q2) & (C \otimes B) \oslash A \vdash (C \oslash A) \otimes B & A \obslash (B \otimes C) \vdash B \otimes (A \obslash C) & (Q4)
\end{array} \qquad (18)$$

The general picture that emerges is a landscape where the minimal symmetric Lambek calculus $\mathbf{LG}_\emptyset$ can be extended either with $P1$–$P4$ or with their converses, or with the combination of the two. It is shown in [6] that each of these choices is conservative with respect to $\mathbf{LG}_\emptyset$, but that the combination is not: with (18) + (15) Associative/Commutative perturbations for $\otimes$/$\oplus$ become derivable. For the analysis of scope construal in Section 4, we use the extension with (15), and in the remainder of the paper we refer to this combination as **LG**.

[3] Earlier presentations of [14] such as [19,13] have been incomplete in only giving the 'mixed associativity' instances $P1$ and $P3$. The full set of postulates is essential for our linguistic applications.

Axiom

$$A \vdash A$$

Monotonicity principles

$$\frac{A \vdash B \qquad C \vdash D}{\begin{array}{c} A \otimes C \vdash B \otimes D; \\ A/D \vdash B/C; \\ D\backslash A \vdash C\backslash B \end{array}} \qquad \frac{A \vdash B \qquad C \vdash D}{\begin{array}{c} A \oplus C \vdash B \oplus D; \\ A \oslash D \vdash B \oslash C; \\ D \obslash A \vdash C \obslash B \end{array}}$$

(Dual) residuation principles

$$\begin{array}{c} B \vdash A\backslash C \\ \hline\hline A \otimes B \vdash C \\ \hline\hline A \vdash C/B \end{array} \begin{array}{l} rp \\ rp \end{array} \qquad \begin{array}{c} C \oslash A \vdash B \\ \hline\hline C \vdash B \oplus A \\ \hline\hline B \obslash C \vdash A \end{array} \begin{array}{l} drp \\ drp \end{array}$$

Grishin distributivity principles

$$\frac{B \obslash (C \otimes A) \vdash D}{(B \obslash C) \otimes A \vdash D} \quad \frac{(A \otimes C) \oslash B \vdash D}{A \otimes (C \oslash B) \vdash D} \quad \frac{D \vdash (A \oplus C)/B}{D \vdash A \oplus (C/B)} \quad \frac{D \vdash B\backslash(C \oplus A)}{D \vdash (B\backslash C) \oplus A}$$

$$\frac{B \obslash (A \otimes C) \vdash D}{A \otimes (B \obslash C) \vdash D} \quad \frac{(C \otimes A) \oslash B \vdash D}{(C \oslash B) \otimes A \vdash D} \quad \frac{D \vdash (C \oplus A)/B}{D \vdash (C/B) \oplus A} \quad \frac{D \vdash B\backslash(A \oplus C)}{D \vdash A \oplus (B\backslash C)}$$

Fig. 5. LG in cut-free form.

2.5. Relational semantics

We have seen in (3) that from the modal logic perspective, the binary type-forming operation $\otimes$ is interpreted as an existential modality with respect to a ternary accessibility relation $R_\otimes$ ('Merge'). The residual / and \ operations are the corresponding universal modalities. For the coproduct $\oplus$ and its residuals, the dual situation obtains: $\oplus$ here is the universal modality interpreted with respect to an accessibility relation $R_\oplus$; the co-implications are the corresponding existential modalities.

$$\begin{array}{l} x \Vdash A \oplus B \text{ iff } \forall yz.R_\oplus xyz \text{ implies } (y \Vdash A \text{ or } z \Vdash B) \\ y \Vdash C \oslash B \text{ iff } \exists xz.R_\oplus xyz \text{ and } z \nVdash B \text{ and } x \Vdash C \\ z \Vdash A \obslash C \text{ iff } \exists xy.R_\oplus xyz \text{ and } y \nVdash A \text{ and } x \Vdash C \end{array} \tag{19}$$

Soundness and completeness for the relational semantics of $\mathbf{LG}_\emptyset$ and its extensions with the Grishin distributivity principles is established in [16]. In the canonical model, worlds are construed as *weak filters*, i.e. sets of formulas closed under derivability.[4] Completeness for the minimal system $\mathbf{LG}_\emptyset$ does not impose any restrictions on the interpretation of the $R_\otimes$ and $R_\oplus$ relations. For $\mathbf{LG}_\emptyset$ extended with the Grishin interaction principles, one imposes the frame constraints corresponding to the set of postulates one wishes to adopt, (15) or (18), and one shows that in the canonical model these constraints are satisfied.

2.6. Decidable proof search

The axiomatization we have considered so far contains the rule of transitivity/cut (from $A \vdash B$ and $B \vdash C$ conclude $A \vdash C$). In the presence of *expanding* type transitions, transitivity is an undesirable rule from a proof search perspective. The presentation of **LG** in Fig. 5 consists of the identity axiom together with the (dual) residuation principles (10), the monotonicity rules (14), and the Grishin postulates (15) in rule form. It is shown in [22] that adding transitivity to the rules in Fig. 5 does not increase the set of derivable theorems: every derivation that makes use of transitivity/cut can be transformed in a cut-free derivation. Backward-chaining cut-free proof search on the basis of Fig. 5 provides a decision procedure for **LG**: the monotonicity rules reduce a problem $A \vdash B$ to smaller problems by removing a matching pair of connectives; the residuation rules and the Grishin interaction rules provide a finite number of alternative forms for $A \vdash B$ which one can try for pattern-matching with the monotonicity rules.

The formulation of Fig. 5 is a close relative of Display Logic; see [13] for a comprehensive view on substructural systems from a Display Logic perspective. In Display Logic every logical connective has a matching structural connective (not just for $\otimes$ and $\oplus$ as in the Gentzen sequent calculus). Structural connectives are introduced by explicit rewriting steps; the Grishin rules

[4] Allwein and Dunn [1] develop richer models for a hierarchy of substructural logics, accommodating both the lattice operations and (co)product and (co)implications.

$$\frac{}{\Gamma, x : A \vdash x : A \mid \Delta}\, Ax_R \qquad \frac{}{\Gamma \mid \alpha : A \vdash \alpha : A, \Delta}\, Ax_L$$

$$\frac{c : (\Gamma \vdash \alpha : A, \Delta)}{\Gamma \vdash \mu\alpha.c : A \mid \Delta}\, \mu \qquad \frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \mid \tilde{\mu}x.c : A \vdash \Delta}\, \tilde{\mu}$$

$$\frac{\Gamma \vdash v : A \mid \Delta \quad \Gamma \mid e : A \vdash \Delta}{\langle v \mid e \rangle : (\Gamma \vdash \Delta)}\, Cut$$

$$\frac{\Gamma, x : A \vdash t : B \mid \Delta}{\Gamma \vdash \lambda x.t : A \to B \mid \Delta} \to R \qquad \frac{\Gamma \vdash v : A \mid \Delta \quad \Gamma \mid e : B \vdash \Delta}{\Gamma \mid v \cdot e : A \to B \vdash \Delta} \to L$$

$$\frac{\Gamma \mid e : A \vdash \Delta \quad \Gamma \vdash v : B \mid \Delta}{\Gamma \vdash e \cdot v : B - A \mid \Delta}\, {-R} \qquad \frac{\Gamma \mid e : B \vdash \beta : A, \Delta}{\Gamma \mid \tilde{\lambda}\beta.e : B - A \vdash \Delta}\, {-L}$$

Fig. 6. Curien/Herbelin: $\bar{\lambda}\mu\tilde{\mu}$ terms for classical sequent calculus.

$A \to B$	$A \backslash B$	B/A		$A - B$	$A \oslash B$	$B \obslash A$
$v \cdot e$	$v \backslash e$	e / v	contexts	$\tilde{\lambda}\beta.e$	$\tilde{\lambda}^r\beta.e$	$\tilde{\lambda}^l\beta.e$
$\lambda x.v$	$\lambda^l x.v$	$\lambda^r x.v$	terms	$e \cdot v$	$v \oslash e$	$e \obslash v$

Fig. 7. Non-directional $\bar{\lambda}\mu\tilde{\mu}$ constructs and their directional λ**LG** counterparts.

and residuation principles are then expressed as structural rules. The presentation of **LG** of Fig. 5 is entirely formula-based; the distinction between a 'logical' and a 'structural' occurrence of a type-forming operation is determined by its polarity: on the left of the turnstile, principal occurrences of $\otimes$, $\oslash$, and $\obslash$ are structural, which means the (dual) residuation and distributivity rules are applicable to these occurrences; on the right of the turnstile, principal occurrences of $\otimes$, $\oslash$, and $\obslash$ are logical, which means these occurrences must be introduced by the monotonicity rules. For the complementary set of connectives $\oplus$, /, \, the dual situation obtains.[5]

3. Proofs and terms

Let us turn to the computational semantics of **LG** derivations. In Section 1, we saw how for **(N)L** this semantics is obtained by a translation $(\cdot)'$ into **LP** derivations, i.e. proofs of Intuitionistic Multiplicative Linear Logic and the associated linear lambda terms. Here we do the same for **LG**. We first present a term language to label **LG** derivations. We then give a mapping of the **LG** proof terms into linear lambda terms via a continuation-passing-style (CPS) translation. The mapping can be realized in two ways, reflecting call-by-value and call-by-name evaluation strategies. These two options are related by the $\cdot^{\infty}$ symmetry.

Our point of departure is the calculus of Fig. 5, with now *labeled* formulae as the basic declarative units. In the case of **(N)L**, input formulas were labeled with variables; the focus for the construction of the proof terms built out of these variables was exclusively on the unique *succedent* formula. In the case of the symmetric calculus **LG** this is no longer true: we need to supplement the rules of Fig. 5 with an explicit mechanism to keep track of the formula that is in focus, on the left or on the right of the turnstile; the remaining inactive input formulas are labeled with variables, and inactive output formulas with covariables.

3.1. *λ**LG**: proof terms for **LG** derivations*

We will derive the computational semantics for **LG** from the $\bar{\lambda}\mu\tilde{\mu}$ calculus of [9], a term language isomorphic to proofs in the implication/co-implication fragment of classical sequent calculus *LK*. We give the sequent rules and the associated $\bar{\lambda}\mu\tilde{\mu}$ terms in Fig. 6.

The $\bar{\lambda}\mu\tilde{\mu}$ term language makes a distinction between three types of expressions: *terms* ($x, v, \ldots$), evaluation *contexts* ($\alpha, e, \ldots$), and *commands c*. Terms and contexts are assigned to sequents with a single formula in focus; the focused formula gives the type of a proof. It is marked off from the passive formulas by means of the bar notation. If the focus is on a succedent formula, it is labeled with a term; if it is on an antecedent formula, it is labeled with a context

terms	v	$::=$	$\mu\alpha.c \mid x \mid \lambda x.v \mid e \cdot v$
evaluation contexts	e	$::=$	$\tilde{\mu}x.c \mid \alpha \mid v \cdot e \mid \tilde{\lambda}\beta.e$
commands	c	$::=$	$\langle v \mid e \rangle$

The axiomatic sequent comes in two forms, depending on whether one focuses on an antecedent or on a succedent formula. The cut rule provides an evaluation context with a term of the required type; it results in an unfocused sequent (a

[5] A proper Display Logic formulation of **LG** and of the term calculus discussed in the next section can be found in [23].

Axiom and co-axiom, Cut

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

$$\frac{\Gamma \xrightarrow{v} A \mid \quad \mid A \xrightarrow{e} \Delta}{\Gamma \xrightarrow{\langle v|e\rangle} \Delta}\ \text{Cut}$$

Focusing rules

$$\frac{\Gamma \xrightarrow{c} \beta : B}{\Gamma \xrightarrow{\mu\beta.c} B \mid}\ \mu \qquad \frac{x : A \xrightarrow{c} \Delta}{\mid A \xrightarrow{\widetilde{\mu}x.c} \Delta}\ \widetilde{\mu}$$

Logical rules

$$\frac{\Gamma \xrightarrow{v} A \mid \quad \mid B \xrightarrow{e} \Delta}{\mid A\backslash B \xrightarrow{v\,\backslash\, e} \Gamma\backslash\Delta}\ \backslash L \qquad \frac{\Gamma \xrightarrow{v} A \mid \quad \mid B \xrightarrow{e} \Delta}{\Gamma \oslash \Delta \xrightarrow{v \oslash e} A \oslash B \mid}\ \oslash R$$

$$\frac{\Gamma \xrightarrow{v} A \mid \quad \mid B \xrightarrow{e} \Delta}{\mid B/A \xrightarrow{e\,/\,v} \Delta/\Gamma}\ /L \qquad \frac{\Gamma \xrightarrow{v} A \mid \quad \mid B \xrightarrow{e} \Delta}{\Delta \obslash \Gamma \xrightarrow{e \obslash v} B \obslash A \mid}\ \obslash R$$

$$\frac{y : B \otimes \Gamma \xrightarrow{v} A \mid}{\Gamma \xrightarrow{\lambda^l y.v} B\backslash A \mid}\ \backslash R \qquad \frac{\mid A \xrightarrow{e} \Delta \oplus \beta : B}{\mid A \oslash B \xrightarrow{\widetilde{\lambda}^r \beta.e} \Delta}\ \oslash L$$

$$\frac{\Gamma \otimes y : B \xrightarrow{v} A \mid}{\Gamma \xrightarrow{\lambda^r y.v} A/B \mid}\ /R \qquad \frac{\mid A \xrightarrow{e} \beta : B \oplus \Delta}{\mid B \obslash A \xrightarrow{\widetilde{\lambda}^l \beta.e} \Delta}\ \obslash L$$

Structural rules (command c unaffected):
(dual) residuation rules and Grishin distributivities of Figure 5

Fig. 8. λ**LG**.

command). A command can be turned into a term by non-deterministically selecting a succedent formula as the focus (the μ rule), or into an evaluation context by putting an antecedent formula in focus (the $\widetilde{\mu}$ rule). The type-forming operations considered here are implication and co-implication (difference). Their rules are fully symmetric. An implication introduced in the antecedent (the $\to L$ rule) is associated with an evaluation context. The $\to R$ rule produces a term to put into such an applicative context. Symmetrically, a co-implication introduced in the succedent by means of the $-R$ rule yields a term. An evaluation context for such a difference term is produced by the $-L$ rule.

We will refer to the term language coding **LG** proofs as λ**LG**. To adapt the $\overline{\lambda}\mu\widetilde{\mu}$ calculus to λ**LG**, we have to restrict and refine it. As for the restriction: the classical sequent calculus in correspondence with the $\overline{\lambda}\mu\widetilde{\mu}$ terms has an additive resource management, with implicit Contraction and Weakening. For **LG**, we need a multiplicative resource management. The term formation rules for λ**LG**, then, are subject to the restriction that for the rules combining a term v and a context e the sets of free (co)variable occurrences of v and e are disjoint. The (co)variable-binding rules have exactly one free (co)variable occurrence in their scope.

The refinement concerns the fact that also Commutativity and Associativity are lacking in **LG**. As a result, the non-directional (co-)implication types and their associated terms are split in two directional versions. Fig. 7 gives our notation. Also here, we have restrictions on the term formation rules, resulting from the absence of Associativity/Commutativity. Let v^* be the bracketed string of free variable occurrences associated with v. One can form a left abstraction term $\lambda^l x.v$ provided $v^* = (x\sigma)$, i.e. the bound variable has to be the leftmost free variable of v^*; $(\lambda^l x.v)^*$ then is σ. Similarly for right abstraction terms $\lambda^r x.v$ where the bound variable has to be the rightmost free variable of v^*, and for the left and right co-abstraction cases, where the binding concerns co-variables. The μ and $\widetilde{\mu}$ binders do not impose positional restrictions on the (co)variable they bind.

The sequent rules in Curry–Howard correspondence with the λ**LG** language are given in Fig. 8. We restrict attention here to the (co)implication fragment, with formulas taken from the grammar $\mathcal{F} ::= \mathcal{A} \mid \mathcal{F}/\mathcal{F} \mid \mathcal{F}\backslash\mathcal{F} \mid \mathcal{F} \oslash \mathcal{F} \mid \mathcal{F} \obslash \mathcal{F}$.

Like $\overline{\lambda}\mu\widetilde{\mu}$, λ**LG** distinguishes three kinds of sequent, corresponding to the distinction between terms ($x, v, \ldots$), contexts ($\alpha, e, \ldots$), commands (c). The sequents associated with terms/contexts have a distinguished formula in focus; the focused formula is marked off by means of the bar, and gives the type of the proof. As for notation: the proof term associated with the focused formula appears as a superscript of the derivability symbol (which we write as $\vdash$ or as a long arrow to accommodate the terms as they grow bigger).

- terms (focus right): $\Gamma \xrightarrow{v} B \mid$
- contexts (focus left): $\mid A \xrightarrow{e} \Delta$
- commands (unfocused): $\Gamma \xrightarrow{c} \Delta$

Whereas in the Associative/Commutative calculus for $\overline{\lambda}\mu\widetilde{\mu}$, formulas can be freely taken out of their antecedent/succedent context to be put in focus, in λ**LG** we have to explicitly *display* a formula by means of the (dual) residuation rules and Grishin interactions of Fig. 5, so that the original structural context of the formula is preserved. The grammar for structural contexts ($\Gamma, \Delta, \ldots$) is given below. Structures are built out of labeled formulas. We distinguish input (antecedent) and output (succedent) structures. A formula leaf of an input (output) structure is labeled with a variable (co-variable)

$$
\begin{array}{rcl}
\text{input} & \mathcal{S}^\bullet & ::= \ \mathcal{V}ar : \mathcal{F} \mid \mathcal{S}^\bullet \otimes \mathcal{S}^\bullet \mid \mathcal{S}^\circ \obslash \mathcal{S}^\bullet \mid \mathcal{S}^\bullet \oslash \mathcal{S}^\circ \\[1ex]
\text{output} & \mathcal{S}^\circ & ::= \ \mathcal{C}ovar : \mathcal{F} \mid \mathcal{S}^\circ \oplus \mathcal{S}^\circ \mid \mathcal{S}^\bullet \backslash \mathcal{S}^\circ \mid \mathcal{S}^\circ / \mathcal{S}^\bullet
\end{array}
$$

We comment on the components of Fig. 8. In the absence of Weakening, the (co)axiom sequents cannot have unused material. In the absence of Contraction, the two-premise rules cannot duplicate material: structural contexts are multiplicatively merged. The cut rule has the special instances where one of the premises is axiomatic; these give the converses of the focusing rules $\mu^{-1}, \widetilde{\mu}^{-1}$ as derived rules of inference. The slash left and co-slash right rules are the Monotonicity rules of Fig. 5 in their labeled form. The slash right and co-slash left rules are half of the (dual) residuation laws, operating at the formula level. The (dual) residuation rules in general and the Grishin distributivities operate on unfocused sequents.

As a sample derivation, in (20) on the left we compute the term for the transition from $(s \oslash s) \obslash np$, the type we'll use for QP expressions, to the Lambek type for QP subjects $s/(np\backslash s)$: $q : (s \oslash s) \obslash np \xrightarrow{v} s/(np\backslash s) \mid$.

We omit the term labeling for the intermediate steps, as it is completely determined (up to alphabetic variants) by the labeling at the (co)axiom leaves and the rules applied. For legibility, the structural occurrences of the operations $\otimes, \oplus$ and their residuals are set off by dots. A derivation consists of logical steps, operating on formulas that have been brought in focus, interspersed with blocks of structural reasoning operating on unfocused sequents.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{z}{np \vdash np \mid} \quad \overset{\alpha}{\mid s \vdash s}}{\mid (np\backslash s) \vdash (np \cdot \backslash \cdot s)}\,\backslash L}{(np\backslash s) \vdash (np \cdot \backslash \cdot s)}\,\widetilde{\mu}^{-1}}{(np \cdot \otimes \cdot (np\backslash s)) \vdash s}\,\mathit{rp}}{(np \cdot \otimes \cdot (np\backslash s)) \vdash s \mid}\,\mu \qquad \overset{\gamma}{\mid s \vdash s}}{((np \cdot \otimes \cdot (np\backslash s)) \cdot \oslash \cdot s) \vdash (s \oslash s) \mid}\,\oslash R}{((np \cdot \otimes \cdot (np\backslash s)) \cdot \oslash \cdot s) \vdash (s \oslash s)}\,\mu^{-1}}{(np \cdot \otimes \cdot (np\backslash s)) \vdash ((s \oslash s) \cdot \oplus \cdot s)}\,\mathit{drp}}{np \vdash (((s \oslash s) \cdot \oplus \cdot s) \cdot / \cdot (np\backslash s))}\,\mathit{rp}}{np \vdash ((s \oslash s) \cdot \oplus \cdot (s \cdot / \cdot (np\backslash s)))}\,P1'}{\mid np \vdash ((s \oslash s) \cdot \oplus \cdot (s \cdot / \cdot (np\backslash s)))}\,\widetilde{\mu}}{\mid ((s \oslash s) \obslash np) \vdash (s \cdot / \cdot (np\backslash s))}\,\obslash L}{((s \oslash s) \obslash np) \vdash (s \cdot / \cdot (np\backslash s))}\,\widetilde{\mu}^{-1}}{(((s \oslash s) \obslash np) \cdot \otimes \cdot (np\backslash s)) \vdash s}\,\mathit{rp}}{(((s \oslash s) \obslash np) \cdot \otimes \cdot (np\backslash s)) \vdash s \mid}\,\mu}{((s \oslash s) \obslash np) \vdash (s/(np\backslash s)) \mid}\,/R
\qquad
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{z}{np \vdash np \mid} \quad \overset{\alpha}{\mid s \vdash s}}{\mid (np\backslash s) \vdash (np \cdot \backslash \cdot s)}\,\backslash L}{(np \cdot \otimes \cdot (np\backslash s)) \vdash s \mid}\,\leftrightharpoons \qquad \overset{\gamma}{\mid s \vdash s}}{((np \cdot \otimes \cdot (np\backslash s)) \cdot \oslash \cdot s) \vdash (s \oslash s) \mid}\,\oslash R}{\mid np \vdash ((s \oslash s) \cdot \oplus \cdot (s \cdot / \cdot (np\backslash s)))}\,\rightleftharpoons}{\mid ((s \oslash s) \obslash np) \vdash (s \cdot / \cdot (np\backslash s))}\,\obslash L}{(((s \oslash s) \obslash np) \cdot \otimes \cdot (np\backslash s)) \vdash s \mid}\,\leftrightharpoons}{((s \oslash s) \obslash np) \vdash (s/(np\backslash s)) \mid}\,/R
\tag{20}
$$

$$
\lambda^r y.(\mu\gamma.\langle\, q \mid \widetilde{\lambda}^l \beta.(\widetilde{\mu} z.\langle\, (\mu\alpha.\langle\, y \mid (z \setminus \alpha)\,\rangle \oslash \gamma) \mid \beta \,\rangle)\,\rangle)
$$

3.1.1. Derived rules of inference: shifting focus

For legibility, we will use a more compact derivation format, obtained by compiling away the structural part in derived inference rules for *shifting* the focus. Notice first that the cut rule has two special instances, where the left premise is an instance of Ax, or the right premise of Co-Ax. Such cuts give rise to the *defocusing* rules below, the converses of the μ and $\widetilde{\mu}$ focusing rules.

$$\dfrac{|\ A \xrightarrow{e} \Delta}{x : A \xrightarrow{\langle x|e\rangle} \Delta}\ \widetilde{\mu}^{-1} \qquad \dfrac{\Gamma \xrightarrow{v} B\ |}{\Gamma \xrightarrow{\langle v|\beta\rangle} \beta : B}\ \mu^{-1} \tag{21}$$

With the aid of (21), we obtain the four focus shifting rules below. Vertical dots stand for a sequence of *structural* rule applications (residuation laws, Grishin distributivities) rewriting the conclusion of the defocusing rules to the premise of the focusing rules. Compare the derivation with explicit structural steps in (20) on the left with the equivalent derivation on the right, which hides the structural steps in the focus shifting rules.

$$\dfrac{\begin{array}{c}\dfrac{\Gamma \xrightarrow{v} B\ |}{\Gamma \xrightarrow{\langle v|\beta\rangle} \beta : B}\ \mu^{-1}\\ \vdots \\ x : A \xrightarrow{\langle v|\beta\rangle} \Delta\end{array}}{|\ A \xrightarrow{\widetilde{\mu} x.\langle v|\beta\rangle} \Delta}\ \widetilde{\mu} \quad\rightsquigarrow\quad \dfrac{\Gamma \xrightarrow{v} B\ |}{|\ A \xrightarrow{\widetilde{\mu} x.\langle v|\beta\rangle} \Delta}\ \rightleftharpoons \qquad\qquad \dfrac{\begin{array}{c}\dfrac{|\ B \xrightarrow{e} \Delta}{y : B \xrightarrow{\langle y|e\rangle} \Delta}\ \widetilde{\mu}^{-1}\\ \vdots \\ x : A \xrightarrow{\langle y|e\rangle} \Delta'\end{array}}{|\ A \xrightarrow{\widetilde{\mu} x.\langle y|e\rangle} \Delta'}\ \widetilde{\mu} \quad\rightsquigarrow\quad \dfrac{|\ B \xrightarrow{e} \Delta}{|\ A \xrightarrow{\widetilde{\mu} x.\langle y|e\rangle} \Delta'}\ \Leftarrow$$

$$\dfrac{\begin{array}{c}\dfrac{|\ B \xrightarrow{e} \Delta}{y : B \xrightarrow{\langle y|e\rangle} \Delta}\ \widetilde{\mu}^{-1}\\ \vdots \\ \Gamma \xrightarrow{\langle y|e\rangle} \alpha : A\end{array}}{\Gamma \xrightarrow{\mu\alpha.\langle y|e\rangle} A\ |}\ \mu \quad\rightsquigarrow\quad \dfrac{|\ B \xrightarrow{e} \Delta}{\Gamma \xrightarrow{\mu\alpha.\langle y|e\rangle} A\ |}\ \leftrightharpoons \qquad\qquad \dfrac{\begin{array}{c}\dfrac{\Gamma \xrightarrow{v} B\ |}{\Gamma \xrightarrow{\langle v|\beta\rangle} \beta : B}\ \mu^{-1}\\ \vdots \\ \Gamma' \xrightarrow{\langle v|\beta\rangle} \alpha : A\end{array}}{\Gamma' \xrightarrow{\mu\alpha.\langle v|\beta\rangle} A\ |}\ \mu \quad\rightsquigarrow\quad \dfrac{\Gamma \xrightarrow{v} B\ |}{\Gamma' \xrightarrow{\mu\alpha.\langle v|\beta\rangle} A\ |}\ \Rightarrow$$

Derived rules of inference: Grishin distributivities The rules $(/, \backslash R')$ and $(\oslash, \obslash L')$ below are derived rules of inference obtained by composing the original $(/, \backslash R)$ and $(\oslash, \obslash L)$ rules with the structural steps licensed by the Grishin distributivities. In the antecedent (succedent), the notation $\Gamma\{\Delta\}$ picks out an input (output) substructure Δ along the input (output) branch of $\oslash$, $\obslash$ $(/, \backslash)$ structures. (Example: consider an antecedent structure $\Gamma_1 \obslash (\Gamma_2 \oslash \Gamma_3)$. Writing $\Box$ for the hole, it matches the $\Gamma\{\Delta\}$ notation with (i) $\Gamma = \Box$, $\Delta = \Gamma_1 \obslash (\Gamma_2 \oslash \Gamma_3)$, (ii) $\Gamma = \Gamma_1 \obslash \Box$, $\Delta = \Gamma_2 \oslash \Gamma_3$, or (iii) $\Gamma = \Gamma_1 \obslash (\Box \oslash \Gamma_3)$, $\Delta = \Gamma_2$.)

$$\dfrac{\Gamma'\{y : B \otimes \Gamma\} \xrightarrow{v} A\ |}{\Gamma'\{\Gamma\} \xrightarrow{\lambda^l y.v} B\backslash A\ |}\ \backslash R' \quad \dfrac{|\ A \xrightarrow{e} \Delta'\{\Delta \oplus \beta : B\}}{|\ A \oslash B \xrightarrow{\widetilde{\lambda}^r \beta.e} \Delta'\{\Delta\}}\ \oslash L' \quad \dfrac{\Gamma'\{\Gamma \otimes y : B\} \xrightarrow{v} A\ |}{\Gamma'\{\Gamma\} \xrightarrow{\lambda^r y.v} A/B\ |}\ /R' \quad \dfrac{|\ A \xrightarrow{e} \Delta'\{\beta : B \oplus \Delta\}}{|\ B \obslash A \xrightarrow{\widetilde{\lambda}^l \beta.e} \Delta'\{\Delta\}}\ \obslash L' \tag{22}$$

Using the $(/, \backslash R')$ and $(\oslash, \obslash L')$ versions, the premise and conclusion of the focus shifting rules are related purely in terms of the (dual) residuation principles. Compare (20) with the derivation in (23) below where in the $(\obslash L')$ step, $s \oslash s$ commutes with $/$ to find its proper place in the succedent structure. In (20) the Grishin interaction takes place when the focus is shifted from antecedent *np* to succedent $s \oslash s$. In the remainder, we will freely use the compact derivation format.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{z}{np \vdash np\ |} \quad \overset{\alpha}{|\ s \vdash s}}{|\ (np\backslash s) \vdash (np \cdot \backslash \cdot s)}\ \backslash L}{(np \cdot \otimes \cdot (np\backslash s)) \vdash s\ |}\ \leftrightharpoons \qquad \overset{\gamma}{|\ s \vdash s}}{((np \cdot \otimes \cdot (np\backslash s)) \cdot \oslash \cdot s) \vdash (s \oslash s)\ |}\ \oslash R}{|\ np \vdash (((s \oslash s) \cdot \oplus \cdot s) \cdot / \cdot (np\backslash s))}\ \rightleftharpoons}{|\ ((s \oslash s) \obslash np) \vdash (s \cdot / \cdot (np\backslash s))}\ \obslash L'}{(((s \oslash s) \obslash np) \cdot \otimes \cdot (np\backslash s)) \vdash s\ |}\ \leftrightharpoons}{((s \oslash s) \obslash np) \vdash (s/(np\backslash s))\ |}\ /R' \tag{23}$$

3.2. Continuation-passing-style translation

We now provide a continuation-passing-style (CPS) translation that maps **LG** derivations and the associated λ**LG** proof terms to **LP** proofs/terms.

Table 1
CPS translations: types.

$\lceil B/A\rceil = \lceil A\backslash B\rceil$	$=$	$\lceil B\rceil^{\perp} \rightarrow \lceil A\rceil^{\perp}$	$\lfloor A \obslash B\rfloor = \lfloor B \oslash A\rfloor$	$=$	$\lfloor B\rfloor^{\perp} \rightarrow \lfloor A\rfloor^{\perp}$
$\lceil B \obslash A\rceil = \lceil A \oslash B\rceil$	$=$	$\lceil A\backslash B\rceil^{\perp}$	$\lfloor A/B\rfloor = \lfloor B\backslash A\rfloor$	$=$	$\lfloor B \oslash A\rfloor^{\perp}$
	$=$	$(\lceil B\rceil^{\perp} \rightarrow \lceil A\rceil^{\perp})^{\perp}$		$=$	$(\lfloor B\rfloor^{\perp} \rightarrow \lfloor A\rfloor^{\perp})^{\perp}$

Table 2
Values, continuations, computations.

$V^{\mathsf{CBV}}_{A\backslash B}$	$=$	$K^{\mathsf{CBV}}_{B} \rightarrow K^{\mathsf{CBV}}_{A}$	$K^{\mathsf{CBN}}_{B\oslash A}$	$=$	$C^{\mathsf{CBN}}_{B} \rightarrow C^{\mathsf{CBN}}_{A}$
$V^{\mathsf{CBV}}_{A\oslash B}$	$=$	$(K^{\mathsf{CBV}}_{B} \rightarrow K^{\mathsf{CBV}}_{A}) \rightarrow R$	$K^{\mathsf{CBN}}_{B\backslash A}$	$=$	$(C^{\mathsf{CBN}}_{B} \rightarrow C^{\mathsf{CBN}}_{A}) \rightarrow R$
K^{CBV}_{A}	$=$	$V^{\mathsf{CBV}}_{A} \rightarrow R$			
C^{CBV}_{A}	$=$	$K^{\mathsf{CBV}}_{A} \rightarrow R$	C^{CBN}_{A}	$=$	$K^{\mathsf{CBN}}_{A} \rightarrow R$

In the semantics of programming languages, CPS interpretation has 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 the way they relate to each other for different evaluation strategies.

CPS translations can be executed in a variety of ways. In this paper, we adopt the Plotkin-style translation of [20], which we extend with clauses for the difference operations $\oslash$, $\obslash$.[6] We first consider the effect of the CPS mapping at the level of **LG** *types*, comparing a call-by-value (cbv) and a call-by-name (cbn) regime; then we define the CPS translation at the level of the λ**LG** proof terms.

The target language has a distinguished type R (responses). We write $A^{\perp}$ as an abbreviation for $A \rightarrow R$. As in the direct interpretation for **(N)L**, the directional implications $/, \backslash$ and co-implications $\oslash$, $\obslash$ of the source language are mapped to the same interpretation in the non-directional target language **LP**. On the left in Table 1, one finds the cbv translation. For a source language type A, the cbv translation $\lceil\cdot\rceil$ produces a *value* of type A in the target language. For p atomic, $\lceil p\rceil = \widetilde{p}$. Implications $A\backslash B$ are mapped to functions that produce the response type R given *two* inputs: an A value, $\lceil A\rceil$, and a B continuation, i.e. a function from B values to R, $\lceil B\rceil \rightarrow R$. There are two essentially equivalent ways of expressing this idea without using conjunction in the target logic: the curried form $\lceil A\backslash B\rceil = \lceil A\rceil \rightarrow (\lceil B\rceil \rightarrow R) \rightarrow R$ (abbreviated: $\lceil A\rceil \rightarrow \lceil B\rceil^{\perp\perp}$) interprets $A\backslash B$ as a function from A values to B computations. Alternatively, taking first the B continuation and then the A value, one has the interpretation $(\lceil B\rceil \rightarrow R) \rightarrow (\lceil A\rceil \rightarrow R)$ given below. The interpretation of the difference operations $\oslash$, $\obslash$ respects the $(\cdot)^{\infty}$ duality in the sense that the interpretation of an $A \oslash B$ *value* is the same as that of an $A\backslash B$ *continuation*.

On the right in Table 1, one finds the cbn interpretation. The cbn translation $\lfloor\cdot\rfloor$ of a source language type A produces a *continuation* in the target language. The cbn regime is defined by duality: $\lfloor A\rfloor = \lceil A^{\infty}\rceil$.

In what follows, it will be handy to have a way of referring to values, continuations and computations without spelling out the types in full. In Table 2, we write V_A for values of type A, K_A for continuations, and C_A for computations.

The cbv translation of the proof terms of λ**LG** is given in Table 3. The translation takes terms to computations, and contexts to continuations. With respect to the $v \backslash e : A\backslash B$ and $v \oslash e : A \oslash B$ constructs, notice that the slash and the division operations process the term $v : A$ and the context $e : B$ in exactly the same way: $\lambda u.(\lceil v\rceil\ (u\ \lceil e\rceil))$. On the left side of the turnstile, this produces an $A\backslash B$ continuation; on the right side, an $A \oslash B$ computation, i.e. $\lambda k.(k\ \lceil v \backslash e\rceil)$.

For the cbn translation, we have $\lfloor\cdot\rfloor = \lceil\cdot^{\infty}\rceil$. For commands $\langle v|e\rangle^{\infty} = \langle e^{\infty}|v^{\infty}\rangle$. For terms and contexts, we have the mapping given in (24).

$$\begin{array}{rcl@{\qquad}rcl}
x^{\infty} & = & \alpha & \alpha^{\infty} & = & x \\
(\lambda^{l}x.v)^{\infty} & = & \widetilde{\lambda}^{r}\alpha.v^{\infty} & (\widetilde{\lambda}^{r}\alpha.e)^{\infty} & = & \lambda^{l}x.e^{\infty} \\
(\lambda^{r}x.v)^{\infty} & = & \widetilde{\lambda}^{l}\alpha.v^{\infty} & (\widetilde{\lambda}^{l}\alpha.e)^{\infty} & = & \lambda^{r}x.e^{\infty} \\
(v \oslash e)^{\infty} & = & e^{\infty} \backslash v^{\infty} & (v \backslash e)^{\infty} & = & e^{\infty} \oslash v^{\infty} \\
(e \obslash v)^{\infty} & = & v^{\infty} / e^{\infty} & (e / v)^{\infty} & = & v^{\infty} \obslash e^{\infty} \\
(\mu\beta.c)^{\infty} & = & \widetilde{\mu}y.c^{\infty} & (\widetilde{\mu}y.c)^{\infty} & = & \mu\beta.c^{\infty}
\end{array} \tag{24}$$

Table 4 specifies how te CPS translations relate proofs/terms in the source logic **LG** to proofs/terms in the target logic **LP** (cf Prop 8.1 and 8.3 of [9]; the preservation of types theorem of [20]). We write $\Gamma^{\bullet}$ for antecedent (input) formulas of structure Γ; Γ° for succedent (output) formulas of structure Γ. In the case where the antecedent (succedent) consists purely of input (output) parts, the call-by-value translation for an λ**LG** term v of type B is an **LP** proof taking antecedent *values* to a

[6] In [7], we used the translation of [9] where a difference type $A \oslash B$ is seen as a *pair* of an A value and a B continuation, and hence $\lceil A\backslash B\rceil = \lceil A \oslash B\rceil^{\perp}$. Under that interpretation, our account of scope ambiguity could only be executed under the cbn regime. The translation adopted here accommodates the scope account both under cbv and cbn.

Table 3
Call-by-value translation: proof terms.

(terms)	$\lceil x \rceil$	$=$	$\lambda k.(k\,\widetilde{x})$
	$\lceil \lambda^l x.v \rceil = \lceil \lambda^r x.v \rceil$	$=$	$\lambda k.(k\,\lambda u \lambda \widetilde{x}.(\lceil v \rceil\; u))$
	$\lceil v \oslash e \rceil = \lceil e \obslash v \rceil$	$=$	$\lambda k.(k\,\lambda u.(\lceil v \rceil\; (u\, \lceil e \rceil)))$
	$\lceil \mu\alpha.c \rceil$	$=$	$\lambda \widetilde{\alpha}.\lceil c \rceil$
(contexts)	$\lceil \alpha \rceil$	$=$	$\widetilde{\alpha}$
	$\lceil v \backslash e \rceil = \lceil e \,/\, v \rceil$	$=$	$\lambda u.(\lceil v \rceil\; (u\, \lceil e \rceil))$
	$\lceil \widetilde{\lambda}^r \beta.e \rceil = \lceil \widetilde{\lambda}^l \beta.e \rceil$	$=$	$\lambda k.(k\,\lambda \widetilde{\beta}.\lceil e \rceil)$
	$\lceil \widetilde{\mu} x.c \rceil$	$=$	$\lambda \widetilde{x}.\lceil c \rceil$
(commands)	$\lceil \langle v \mid e \rangle \rceil$	$=$	$(\lceil v \rceil\; \lceil e \rceil)$

Table 4
Source logic **LG** to target logic **LP**.

LG	$\xrightarrow{\text{CPS translation}}$	**LP**
$\Gamma \xrightarrow{v} B \mid$		$\lceil \Gamma^\bullet \rceil,\ \ \lceil \Gamma^\circ \rceil^\perp \vdash \lceil v \rceil : \lceil B \rceil^{\perp\perp}$
		$\lfloor \Gamma^\bullet \rfloor^\perp, \lfloor \Gamma^\circ \rfloor\ \ \vdash \lfloor v \rfloor : \lfloor B \rfloor^\perp$
$\mid A \xrightarrow{e} \Delta$		$\lceil \Delta^\bullet \rceil,\ \ \lceil \Delta^\circ \rceil^\perp \vdash \lceil e \rceil : \lceil A \rceil^\perp$
		$\lfloor \Delta^\bullet \rfloor^\perp, \lfloor \Delta^\circ \rfloor\ \ \vdash \lfloor e \rfloor : \lfloor A \rfloor^{\perp\perp}$

B computation; the call-by-name translation yields an **LP** proof taking antecedent *computations* to a B computation. For an λ**LG** context e of type A, the call-by-name translation is an **LP** proof taking succedent *continuations* to an A continuation; the call-by-name translation yields an **LP** proof taking succedent continuations to a function from A computations into R.

An enlightening discussion of the dynamics of call-by-value and call-by-name can be found in [26]. From our parsing-as-deduction perspective, the cbv strategy can be seen as data-driven: it processes the arguments before processing the function; the cbn strategy is goal-driven: it processes the function until there is a need to process the arguments. Evaluation *contexts*, in this perspective, are expressions with a placeholder for a missing term: one can either apply a context to a value of the required type, or give it a name ($\widetilde{\mu}$) to refer to it later in the derivation. From the dual perspective, a *term* is an expression with a placeholder for a missing context which wraps itself around the term. A term-as-a-computation can either get input from its context, or one can give it a name (μ) for later reference.

To close this section, we illustrate the cbv and cbn regimes with the translations of the **LG** derivation we gave before. First cbv for $q : (s \oslash s) \obslash np \xrightarrow{v} s/(np\backslash s) \mid$. The λ**LG** term and its cbv translation are given in (25).

$$\begin{aligned} v &= \lambda^r y.(\mu\gamma.\langle\, q \mid \widetilde{\lambda}^l \beta.(\widetilde{\mu} z.\langle\, (\mu\alpha.\langle\, y \mid (z \backslash \alpha)\,\rangle \oslash \gamma) \mid \beta\,\rangle)\,\rangle) \\ \lceil v \rceil &= \lambda k.(k\,\lambda \widetilde{y}.(\lambda \widetilde{\gamma}.(\widetilde{q}\,\lambda \widetilde{\beta}.(\lambda \widetilde{z}.(\widetilde{\beta}\,\lambda u.((\widetilde{\gamma}\;(u\,\widetilde{y}))\;\widetilde{z})))))) \end{aligned} \tag{25}$$

The cbn translation is read off from the $\cdot^\infty$ image of the derivation. We explicitly label all steps this time

$$q : (s \oslash s) \obslash np \xrightarrow{v} s/(np\backslash s) \mid \qquad \xleftrightarrow{\infty} \qquad \mid (s \oslash np) \obslash s \xrightarrow{e} \xi : np/(s\backslash s)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{y_0}{y_0 : s \vdash s \mid} \quad \dfrac{\dfrac{\overset{y_1}{y_1 : s \vdash s \mid} \quad \overset{\gamma}{\mid np \vdash \gamma : np}}{(y_1 : s \oslash \gamma : np) \xrightarrow{y_1 \oslash \gamma} (s \oslash np) \mid}\,\oslash R}{\mid s \xrightarrow{\widetilde{\mu} y_1.\langle\, (y_1 \oslash \gamma) \mid \alpha\,\rangle} (\alpha : (s \oslash np) \oplus \gamma : np)}\,\rightleftharpoons}{\mid (s\backslash s) \xrightarrow{y_0 \backslash \widetilde{\mu} y_1.\langle\, (y_1 \oslash \gamma) \mid \alpha\,\rangle} (y_0 : s \backslash (\alpha : (s \oslash np) \oplus \gamma : np))}\,\backslash L}{(\alpha : (s \oslash np) \obslash (y_0 : s \otimes x : (s\backslash s))) \xrightarrow{\mu\gamma.\langle\, x \mid (y_0 \backslash \widetilde{\mu} y_1.\langle\, (y_1 \oslash \gamma) \mid \alpha\,\rangle)\,\rangle} np \mid}\,\leftrightharpoons}{(\alpha : (s \oslash np) \obslash y_0 : s) \xrightarrow{\lambda^r x.(\mu\gamma.\langle\, x \mid (y_0 \backslash \widetilde{\mu} y_1.\langle\, (y_1 \oslash \gamma) \mid \alpha\,\rangle)\,\rangle)} (np/(s\backslash s)) \mid}\,/R'}{\mid s \xrightarrow{\widetilde{\mu} y_0.\langle\, \lambda^r x.(\mu\gamma.\langle\, x \mid (y_0 \backslash \widetilde{\mu} y_1.\langle\, (y_1 \oslash \gamma) \mid \alpha\,\rangle)\,\rangle) \mid \xi\,\rangle} (\alpha : (s \oslash np) \oplus \xi : (np/(s\backslash s)))}\,\rightleftharpoons}{\mid ((s \oslash np) \obslash s) \xrightarrow{\widetilde{\lambda}^l \alpha.(\widetilde{\mu} y_0.\langle\, \lambda^r x.(\mu\gamma.\langle\, x \mid (y_0 \backslash \widetilde{\mu} y_1.\langle\, (y_1 \oslash \gamma) \mid \alpha\,\rangle)\,\rangle) \mid \xi\,\rangle)} \xi : (np/(s\backslash s))}\,\obslash L' \tag{26}$$

$$\begin{array}{lcl}
e &=& \widetilde{\lambda}^l\alpha.(\widetilde{\mu}y_0.\langle\, \lambda^r x.(\mu\gamma.\langle\, x \mid (y_0 \setminus \widetilde{\mu}y_1.\langle\, (y_1 \oslash \gamma) \mid \alpha\,\rangle)\,\rangle) \mid \xi\,\rangle) \;=\; v^\infty \\
\lceil e \rceil &=& \lambda k.(k\, \lambda\widetilde{\alpha}.(\lambda\widetilde{y_0}.(\widetilde{\xi}\, \lambda\widetilde{\gamma}.(\lambda\widetilde{x}.((\widetilde{x}\, \lambda\widetilde{y_1}.(\widetilde{\alpha}\, \lambda u.((u\, \widetilde{\gamma})\, \widetilde{y_1})))\, \widetilde{y_0}))))) \;=\; \lfloor e^\infty \rfloor
\end{array}$$

4. Application: scope construal

We turn now to our analysis of scope construal using the machinery developed in the previous sections. Our account has two components.

Derivational semantics. Here we concentrate on the λ**LG** *proof terms* and their CPS interpretation. We show how, at the level of these proof terms, a type assignment $(s \oslash s) \obslash np$ to generalized quantifier phrases solves the problems for a type assignment $s/(np\backslash s)$ in the asymmetric setting mentioned in Section 1. As for type uniformity: with a $(s \oslash s) \obslash np$ assignment, a QP can occupy all positions where a np argument is selected. As for type flexibility: in environments with multiple generalized quantifier phrases, **LG** generates scope construals parallel to the surface order of the QPs, but also permutations of these construals, both locally (within the context of the predicate selecting the np arguments bound by the QPs) and non-locally. Our account accommodates both the call-by-value and the call-by-name regimes, the first reflecting a data-driven, the second a goal-driven parsing strategy.

Lexical semantics. Our second aim is to relate the CPS interpretation to the model-theoretic interpretation for Lambek derivations discussed in Section 1. To realize this second aim, we identify the response type R with V_s as far as their denotations are concerned. We define translations $\llbracket\cdot\rrbracket$, $\lfloor\!\lfloor\cdot\rfloor\!\rfloor$ lifting the lexical constants from the type they have in the original Lambek semantics to the type required by the cbv or cbn level. For types staying within the Lambek vocabulary, the $\llbracket\cdot\rrbracket$, $\lfloor\!\lfloor\cdot\rfloor\!\rfloor$ translations produce the readings associated with **NL** derivations. For types using the dual vocabulary, specifically our QP type $(s \oslash s) \obslash np$, the translations result in scope construals that are beyond the reach of **NL**.

To illustrate the interplay between derivational and lexical semantics, we give a worked-out example of a simple Subject Verb Phrase combination, 'Molly left', juxtaposing the cbv and cbn interpretation strategies.

$$\begin{array}{ccc}
\textit{cbv} & & \textit{cbn} \\[2ex]
\dfrac{\dfrac{\dfrac{\overset{\beta}{\mid np \vdash np}}{np \vdash np \mid}\leftrightharpoons \qquad \overset{\alpha}{\mid s \vdash s}}{\mid (np\backslash s) \vdash (np \cdot \backslash \cdot s)}\,\backslash L}{(np \cdot \otimes \cdot (np\backslash s)) \vdash s \mid}\leftrightharpoons
& \xleftrightarrow{\;\infty\;} &
\dfrac{\dfrac{\overset{x}{s \vdash s \mid} \qquad \dfrac{\overset{y}{np \vdash np \mid}}{\mid np \vdash np}\rightleftharpoons}{(s \cdot \oslash \cdot np) \vdash (s \oslash np) \mid}\,\oslash R}{\mid s \vdash ((s \oslash np) \cdot \oplus \cdot np)}\rightleftharpoons
\end{array} \tag{27}$$

$$\begin{array}{lll}
\lambda\mathbf{LG}\text{ term:} & \mu\alpha.\langle\, \mathsf{left} \mid (\mu\beta.\langle\, \mathsf{molly} \mid \beta\,\rangle \setminus \alpha)\,\rangle & \widetilde{\mu}x.\langle\, (x \oslash \widetilde{\mu}y.\langle\, y \mid \mathsf{molly}\,\rangle) \mid \mathsf{left}\,\rangle \\
\text{CPS image:} & \lambda c.((\llbracket\mathsf{left}\rrbracket\; c)\; \llbracket\mathsf{molly}\rrbracket) & \lambda c.(\lfloor\!\lfloor\mathsf{left}\rfloor\!\rfloor\; \lambda h.((h\; \lfloor\!\lfloor\mathsf{molly}\rfloor\!\rfloor)\; c))
\end{array}$$

Consider the derivation on the left. The task is to parse a sentence (focused goal formula s), using lexical constants molly and left of type np and $np\backslash s$, respectively. Each lexical item is inserted by a command ⟨*word* | *environment*⟩. The CPS image of the derivation, under the call-by-value interpretation, is an **LP** term denoting a function mapping antecedent *values* to a *computation* for the focused succedent type:

$$V_{np}^{\mathsf{CBV}} \times V_{np\backslash s}^{\mathsf{CBV}} \longrightarrow C_s^{\mathsf{CBV}}$$

The typing of the target language term is given in (28) below. At this level, ⟦molly⟧ and ⟦left⟧ denote an np value and a $np\backslash s$ value, i.e. a function from s continuations to np continuations, respectively.

$$\begin{array}{lclll}
 & & V_{np}^{\mathsf{CBV}} & \llbracket\mathsf{molly}\rrbracket & \lceil np \rceil \\
V_{np\backslash s}^{\mathsf{CBV}} & = & K_s^{\mathsf{CBV}} \to K_{np}^{\mathsf{CBV}} & \llbracket\mathsf{left}\rrbracket & \lceil s \rceil^\perp \to \lceil np \rceil^\perp \\
 & & K_s^{\mathsf{CBV}} & c & \lceil s \rceil^\perp
\end{array} \tag{28}$$

The call-by-name interpretation for the same sentence is given on the right in (27). The target, for the call-by-name interpretation, is an **LP** term denoting a function mapping antecedent *computations* to a *computation* for the focused goal type:

$$C_{np}^{\mathsf{CBN}} \times C_{np\backslash s}^{\mathsf{CBN}} \longrightarrow C_s^{\mathsf{CBN}}$$

The call-by-name interpretation is obtained as the composition of the $(\cdot)^\infty$ duality and the call-by-value mapping; the typing of the resulting term in the target language is given below. Note that at the level of the atoms, we are now dealing with *continuations*, not values. For example, $\lfloor np \rfloor$ denotes an np continuation, and hence $\lfloor np \rfloor^\perp (= \lfloor np \rfloor \to R)$ a computation.

$$V^{\mathsf{CBN}}_{np\backslash s} \;=\; C^{\mathsf{CBN}}_{np} \to \begin{array}{lcl} C^{\mathsf{CBN}}_{np} & \lfloor\!\lfloor \mathsf{molly} \rfloor\!\rfloor & \lfloor np \rfloor^{\perp} \\ C^{\mathsf{CBN}}_{s} & h & \lfloor np \rfloor^{\perp} \to \lfloor s \rfloor^{\perp} \\ C^{\mathsf{CBN}}_{np\backslash s} & \lfloor\!\lfloor \mathsf{left} \rfloor\!\rfloor & (\lfloor np \rfloor^{\perp} \to \lfloor s \rfloor^{\perp})^{\perp\perp} \\ K^{\mathsf{CBN}}_{s} & c & \lfloor s \rfloor \end{array} \tag{29}$$

4.1. Lifting the lexicon

Moving from the derivational to the lexical level, the task is to lift the constants of the direct interpretation of type A' to the type of their CPS images: values V^{CBV}_{A} for the cbv interpretation, computations C^{CBN}_{A} for cbn. We give sample lexical entries in Figs. 9 and 10. For cbv, $\lceil\!\lceil \mathsf{molly} \rceil\!\rceil$ is the identity transformation on a constant molly denoting an individual. For $\lceil\!\lceil \mathsf{left} \rceil\!\rceil$, the recipe lifts a constant left denoting a function from individuals to truth values, $np' \to s'$, to a function from s continuations to np continuations. For the cbn interpretation, the same constants are lifted to computations. Substituting the lexical specifications in the CPS terms, and simplifying, produces the same result for the cbv and cbn interpretations: $\lambda c.(c\ (\mathsf{left\ molly}))$, a term denoting an s computation. Providing this computation with the trivial continuation id, we obtain (left molly), a term that will be evaluated as true if the individual denoted by molly is a member of the set of individuals denoted by left.

$$\begin{array}{lll} \textit{cbv} & & \lambda c.((\lceil\!\lceil \mathsf{left} \rceil\!\rceil\ c)\ \lceil\!\lceil \mathsf{molly} \rceil\!\rceil) \\ & = & \lambda c.((\lambda c'.\lambda x.(c'\ (\mathsf{left}\ x))\ c)\ \mathsf{molly}) \\ & =_{\beta} & \lambda c.(c\ (\mathsf{left\ molly})) \\ \textit{cbn} & & \lambda c.(\lfloor\!\lfloor \mathsf{left} \rfloor\!\rfloor\ \lambda h.((h\ \lfloor\!\lfloor \mathsf{molly} \rfloor\!\rfloor)\ c)) \\ & = & \lambda c.(\lambda Q.(Q\ \lambda q.\lambda c'.(q\ \lambda x.(c'\ (\mathsf{left}\ x))))\ \lambda h.((h\ \lambda k.(k\ \mathsf{molly}))\ c)) \\ & =_{\beta} & \lambda c.(c\ (\mathsf{left\ molly})) \end{array} \tag{30}$$

The sample cbv and cbn lexica of Figs. 9 and 10 contain some more entries that will be used in the discussion of scope construal below. We briefly comment on the way the mappings $\lceil\!\lceil \cdot \rceil\!\rceil$ and $\lfloor\!\lfloor \cdot \rfloor\!\rfloor$ are worked out. For simple n-place predicates like intransitive ‘left’ or transitive ‘teases’ we have constants of type $np' \to s'$ and $np' \to np' \to s'$ denoting (characteristic functions of) sets of individuals and relations between individuals, respectively. A simple first-order transitive verb like ‘teases’ has to be distinguished from higher-order predicates like ‘seeks’, ‘needs’ which cannot be interpreted as relations between individuals. By assigning a type $(np\backslash s)/(s/(np\backslash s))$, with a ‘lifted’ direct object category, we can give a lexical specification in terms of a constant needs denoting a relation between an individual (the subject) and a set of sets of individuals, $((np' \to s') \to s') \to np' \to s'$. The mediator between that type and the cbv value $\lceil (np\backslash s)/(s/(np\backslash s)) \rceil$ is the *shift* combinator; cf [30]; for the cbn interpretation we use the variant *shiftn*. The motivation for a type assignment with a lifted argument carries over to verbs with a sentential complement, like ‘think’, where again a treatment as a relation between an individual and a truth value, corresponding to a type assignment $(np\backslash s)/s$, would be inappropriate. Finally, the sample lexica contain entries for subject QPs, such as ‘somebody’ with type $s/(np\backslash s)$. The lexical specification is given in

$$\begin{array}{rrcl} np & \lceil\!\lceil \mathsf{molly} \rceil\!\rceil & = & \mathsf{molly} \\ np\backslash s & \lceil\!\lceil \mathsf{left} \rceil\!\rceil & = & \lambda c.\lambda x.(c\ (\mathsf{left}\ x)) \\ (np\backslash s)/np & \lceil\!\lceil \mathsf{teases} \rceil\!\rceil & = & \lambda v.\lambda y.(v\ \lambda c.\lambda x.(c\ ((\mathsf{teases}\ y)\ x))) \\ (np\backslash s)/(s/(np\backslash s)) & \lceil\!\lceil \mathsf{needs} \rceil\!\rceil & = & \lambda v.\lambda q.(v\ \lambda c.\lambda x.(c\ ((\mathsf{needs}\ (\textit{shift}\ q))\ x))) \\ & \textit{shift} & = & \lambda h.\lambda c'.((h\ id)\ \lambda c''.\lambda x'.(c''\ (c'\ x'))) \\ (np\backslash s)/(s/(s\backslash s)) & \lceil\!\lceil \mathsf{thinks} \rceil\!\rceil & = & \lambda v.\lambda q.(v\ \lambda c.\lambda x.(c\ ((\mathsf{thinks}\ (\textit{shift}\ q))\ x))) \\ s/(np\backslash s) & \lceil\!\lceil \mathsf{somebody} \rceil\!\rceil & = & \lambda c.\lambda v.(c\ (\exists\ \lambda x.((v\ id)\ x))) \end{array}$$

Fig. 9. Sample lexical entries: cbv interpretation (values).

$$\begin{array}{rrcl} np & \lfloor\!\lfloor \mathsf{molly} \rfloor\!\rfloor & = & \lambda k.(k\ \mathsf{molly}) \\ np\backslash s & \lfloor\!\lfloor \mathsf{left} \rfloor\!\rfloor & = & \lambda Q.(Q\ \lambda q.\lambda c.(q\ \lambda x.(c\ (\mathsf{left}\ x)))) \\ (np\backslash s)/np & \lfloor\!\lfloor \mathsf{teases} \rfloor\!\rfloor & = & \lambda Q.(Q\ \lambda q.\lambda Q'.(Q'\ \lambda q'.\lambda c.(q'\ \lambda x.(q\ \lambda y.(c\ ((\mathsf{teases}\ y)\ x)))))) \\ (np\backslash s)/(s/(np\backslash s)) & \lfloor\!\lfloor \mathsf{needs} \rfloor\!\rfloor & = & \lambda Q.(Q\ \lambda C.\lambda Q'.(Q'\ \lambda q.\lambda c.(q\ \lambda x.(c\ ((\mathsf{needs}\ (\textit{shiftn}\ C))\ x))))) \\ & \textit{shiftn} & = & \lambda h.\lambda c.((h\ \lambda u.(u\ \lambda u'.(u'\ \lambda q'.\lambda c'.(q'\ \lambda x.(c'\ (c\ x))))))\ id) \\ (np\backslash s)/(s/(s\backslash s)) & \lfloor\!\lfloor \mathsf{thinks} \rfloor\!\rfloor & = & \lambda Q.(Q\ \lambda C.\lambda Q'.(Q'\ \lambda q.\lambda c.(q\ \lambda x.(c\ ((\mathsf{thinks}\ (\textit{shiftn}\ C))\ x))))) \\ s/(np\backslash s) & \lfloor\!\lfloor \mathsf{somebody} \rfloor\!\rfloor & = & \lambda Q.(Q\ \lambda u.(u\ \lambda v.\lambda c.(c\ (\exists\ \lambda x.((v\ \lambda c'.(c'\ x))\ id))))) \end{array}$$

Fig. 10. Sample lexical entries: cbn interpretation (computations).

terms of a logical constant $\exists$ of type $(np' \rightarrow s') \rightarrow s'$ denoting a set of sets of individuals. To obtain the set of individuals corresponding to the verb phrase value $np\backslash s$, its CPS image under cbv (the variable v) is given the identity s continuation. Similarly for cbn, where the np hypothesis is treated as a computation, not a value.

In (31) we present the λ**LG** proof terms for some derivations with these lexical items, leaving it to the reader to verify that the CPS images of these proof terms, after substitution of the lexical specifications, produce the given readings, under cbv as well as under cbn

$$\begin{array}{ll}
a & \text{Somebody left.} \\
 & \mu\alpha.\langle\, \mathsf{somebody} \mid (\alpha \;/\; \lambda^{l}y.(\mu\gamma.\langle\, \mathsf{left} \mid (y \setminus \gamma)\,\rangle))\,\rangle \qquad (= v) \\
 & \lambda c.(c\ (\exists\, \lambda x.(\mathsf{left}\, x))) \\
b & \text{Somebody teases Molly.} \\
 & \mu\alpha.\langle\, \mathsf{somebody} \mid (\alpha \;/\; \lambda^{l}y.(\mu\gamma.\langle\, \mathsf{teases} \mid ((y \setminus \gamma) \;/\; \mu\beta.\langle\, \mathsf{molly} \mid \beta\,\rangle)\,\rangle))\,\rangle \\
 & \lambda c.(c\ (\exists\, \lambda x.((\mathsf{teases\ molly})\ x))) \\
c & \text{Molly needs somebody.} \\
 & \mu\alpha.\langle \mathsf{needs} \mid ((\mu\gamma.\langle \mathsf{molly} \mid \gamma\rangle \setminus \alpha) \;/\; \lambda^{r}y.(\mu\beta.\langle \mathsf{somebody} \mid (\beta \;/\; \lambda^{l}z.(\mu\alpha_1.\langle y \mid (z \setminus \alpha_1)\rangle))\rangle))\rangle \\
 & \lambda c.(c\ ((\mathsf{needs}\ \lambda P.(\exists\, \lambda x.(P\, x)))\ \mathsf{molly})) \\
d & \text{Molly thinks somebody left} \\
 & \mu\alpha.\langle\, \mathsf{thinks} \mid ((\mu\delta.\langle\, \mathsf{molly} \mid \delta\,\rangle \setminus \alpha) \;/\; \lambda^{r}z.(\mu\beta.\langle\, z \mid (v \setminus \beta)\,\rangle))\,\rangle \\
 & \lambda c.(c\ ((\mathsf{thinks}\ \lambda c'.(c'\ (\exists\, \lambda x.(\mathsf{left}\, x))))\ \mathsf{molly}))
\end{array} \tag{31}$$

4.1.1. Scope construal

The readings computed above coincide with the direct interpretation of the **NL** derivations, and suffer from the limitations of **NL**. A QP with type $s/(np\backslash s)$ can occur in subject position; with this type, there is no derivation for 'Molly teases somebody' next to (b). In the (c) example, we do find 'somebody' in the object position because the higher-order predicate 'needs' has the lifted $s/(np\backslash s)$ type for its object. But of the two interpretations we would like to associate with this sentence, the QP type assignment $s/(np\backslash s)$ produces only the narrow scope reading, with 'needs' taking scope over $\exists$; the wide scope reading where $\exists$ outscopes 'needs' is lacking. Similarly, for (d), we find the local interpretation where 'somebody' takes scope in the embedded clause; the non-local reading with $\exists$ taking scope at the main clause level cannot be obtained from the $s/(np\backslash s)$ type assignment. Let us turn then to our QP type assignment $(s \oslash s) \obslash np$, and see how it overcomes these expressive limitations.

Type uniformity for simple QP sentences Consider first the variant of (a) where we use 'someone' with the $(s \oslash s) \obslash np$ type. The derivation and the associated λ**LG** term together with the CPS images under cbv and cbn were given above in (25) and (26).

$$\begin{array}{rl}
 & \mu\alpha.\langle\, \mathsf{someone} \mid \widetilde{\lambda}^{l}\beta.(\widetilde{\mu}z.\langle\, (\mu\alpha_1.\langle\, \mathsf{left} \mid (z \setminus \alpha_1)\,\rangle \oslash \alpha) \mid \beta\,\rangle)\,\rangle \qquad (= v) \\
\lceil v \rceil = & \lambda c.(\lceil\!\lceil \mathsf{someone} \rceil\!\rceil\ \lambda y.(\lambda x.(y\ \lambda c'.((\lceil\!\lceil \mathsf{left} \rceil\!\rceil\ (c'\ c))\ x)))) \\
\lfloor v \rfloor = & \lambda c.(\lfloor\!\lfloor \mathsf{someone} \rfloor\!\rfloor\ \lambda q.(\lambda y'.((y'\ \lambda c''.(\lfloor\!\lfloor \mathsf{left} \rfloor\!\rfloor\ \lambda u.((u\ q)\ c'')))\ c)))
\end{array} \tag{32}$$

The CPS images determine the typing constraints that we need in order to solve the equations for the lexical specification of $\lceil\!\lceil \mathsf{someone} \rceil\!\rceil$ and $\lfloor\!\lfloor \mathsf{someone} \rfloor\!\rfloor$. For expository purposes, we give an interpretation in terms of the logical constant $\exists$ in (33). This interpretation is provisional: it is adequate for the derivations of the sample sentences we present below, but it will have to be refined when we discuss the full set of derivational choice points at the end of this section. For cbv, the parameters y and x are of type $K^{\mathsf{CBV}}_{s\oslash s}$ and V^{CBV}_{np}, respectively. The lexical specification for $\lceil\!\lceil \mathsf{someone} \rceil\!\rceil$ in (33) provides the lifted identity combinator $\lambda u.(u\ id)$ for y. For cbn, the parameters q and y' are of type C^{CBN}_{np} and $C^{\mathsf{CBN}}_{s} \rightarrow C^{\mathsf{CBN}}_{s}$, respectively. The lexical specification for $\lfloor\!\lfloor \mathsf{someone} \rfloor\!\rfloor$ provides the computation for the individual variable x for q and the identity on s computations for y'. After lexical substitution, the cbv and cbn readings reduce to $\lambda c.(\exists\, \lambda x.(c\ (\mathsf{left}\, x)))$, as desired.

$$\begin{array}{lrcl}
(s \oslash s) \obslash np & \lceil\!\lceil \mathsf{someone} \rceil\!\rceil & = & \lambda Q.(\exists\, \lambda x.((Q\ \lambda u.(u\ \mathsf{id}))\ x)) \\
 & \lfloor\!\lfloor \mathsf{someone} \rfloor\!\rfloor & = & \lambda Q.(\exists\, \lambda x.((Q\ \lambda k.(k\, x))\ \lambda c'.\lambda c.(c'\ c)))
\end{array} \tag{33}$$

The **LG** type assignment is compatible with any syntactic position where an np argument is selected. Compare (32) with the derivation for a transitive sentence with $(s \oslash s) \obslash np$ in object position, for example 'Molly teases someone'.

$$(\underbrace{np}_{\text{Molly}} \cdot \otimes \cdot (\underbrace{((np\backslash s)/np)}_{\text{teases}} \cdot \otimes \cdot \underbrace{((s \oslash s) \obslash np)}_{\text{someone}})) \overset{v}{\vdash} s \mid$$

$$\begin{array}{rl}
v = & \mu\alpha.\langle\, \mathsf{someone} \mid \widetilde{\lambda}^{l}\beta.(\widetilde{\mu}z.\langle\, (\mu\alpha_1.\langle\, \mathsf{teases} \mid ((\mu\beta_1.\langle \mathsf{molly} \mid \beta_1\rangle \setminus \alpha_1) \;/\; z)\,\rangle \oslash \alpha) \mid \beta\,\rangle)\,\rangle \\
\lceil v \rceil = & \lambda c.(\lceil\!\lceil \mathsf{someone} \rceil\!\rceil\ \lambda y.(\lambda x.(y\ \lambda c'.((\lceil\!\lceil \mathsf{teases} \rceil\!\rceil\ \lambda w.((w\ (c'\ c))\ \lceil\!\lceil \mathsf{molly} \rceil\!\rceil))\ x)))) \\
= & \lambda c.(\exists\, \lambda x.(c\ ((\mathsf{teases}\ x)\ \mathsf{molly})))
\end{array} \tag{34}$$

Surface scope versus inverted scope We turn to sentences with a transitive verb and a $(s \oslash s) \obslash np$ QP in subject and in object position. In shifting the focus from the goal formula to an antecedent formula, there is a choice now: do we activate the subject or the object first? This *derivational* ambiguity leads to the two scope construals for the sentence. The derivations have a common core: $\mu\gamma.\langle\, \mathsf{tv} \mid ((x \setminus \gamma) / y)\,\rangle$ (v' in (35)), the combination of the transitive verb tv with *np* parameters x and y for the subject and object. If one reaches v' by first activating the subject QP, removing its connectives with ($\obslash L$) and ($\oslash R$), one obtains the $\forall/\exists$ construal which coincides with the surface order of the QPs. The alternative option of first activating the object QP leads to the inverted $\exists/\forall$ construal.

$$(\underbrace{((s \oslash s) \obslash np)}_{\text{everyone}} \cdot \otimes \cdot (\underbrace{((np\backslash s)/np)}_{\text{teases}} \cdot \otimes \cdot \underbrace{((s \oslash s) \obslash np)}_{\text{someone}})) \vdash s \mid$$

$$\begin{array}{rl} & \mu\alpha.\langle\, \mathsf{everyone} \mid \widetilde{\lambda}^l\beta_0.(\widetilde{\mu}x.\langle\, (\mu\alpha_1.\langle\, \mathsf{someone} \mid \widetilde{\lambda}^l\beta_1.(\widetilde{\mu}y.\langle\, (v' \oslash \alpha_1) \mid \beta_1 \,\rangle)\,\rangle \oslash \alpha) \mid \beta_0 \,\rangle)\,\rangle \\ = & \lambda c.(\forall\, \lambda x.(\exists\, \lambda y.(c\ ((\mathsf{teases}\ y)\ x)))) \\ & \mu\alpha.\langle\, \mathsf{someone} \mid \widetilde{\lambda}^l\beta_1.(\widetilde{\mu}y.\langle\, (\mu\alpha_1.\langle\, \mathsf{everyone} \mid \widetilde{\lambda}^l\beta_0.(\widetilde{\mu}x.\langle\, (v' \oslash \alpha_1) \mid \beta_0 \,\rangle)\,\rangle \oslash \alpha) \mid \beta_1 \,\rangle)\,\rangle \\ = & \lambda c.(\exists\, \lambda y.(\forall\, \lambda x.(c\ ((\mathsf{teases}\ y)\ x)))) \end{array} \tag{35}$$

Extensional versus higher-order predicates The non-determinism in the choice of the active formula also underlies the scope ambiguity for ‘Molly needs someone’ with a higher-order transitive verb. We saw above that the *de dicto* reading $\lambda c.(c\ ((\mathsf{needs}\ \lambda P.(\exists\, \lambda x.(P\ x)))\ \mathsf{molly}))$, where ‘needs’ outscopes $\exists$, is within the reach of **NL**. To obtain that reading in **LG**, we can activate the type of the verb first. This leads to a derivation where the goal formula s matches the input s of the type for ‘needs’. We have given the derivation for the abbreviated branch in (20).

$$\dfrac{\dfrac{\dfrac{\dfrac{\overset{\gamma}{\mid np \vdash np}}{np \vdash np \mid}\leftrightharpoons \qquad \mid \boxed{s} \overset{\alpha}{\vdash} \boxed{s}}{\mid (np\backslash s) \vdash (np \cdot \backslash \cdot s)}\backslash L \qquad \begin{array}{c}\vdots \\ (20) \\ ((s \oslash s) \obslash np) \ \vdash\ (s/(np\backslash s)) \mid\end{array}}{\mid ((np\backslash s)/(s/(np\backslash s))) \vdash ((np \cdot \backslash \cdot s) \cdot / \cdot ((s \oslash s) \obslash np))}/L}{(np \cdot \otimes \cdot (((np\backslash \boxed{s})/(s/(np\backslash s))) \cdot \otimes \cdot ((s \oslash s) \obslash np))) \vdash \boxed{s} \mid}\leftrightharpoons$$

$$\mu\alpha.\langle\, \mathsf{needs} \mid ((\mu\gamma.\langle\, \mathsf{molly} \mid \gamma \,\rangle \setminus \alpha) \,/\, (20))\,\rangle \tag{36}$$

LG also produces the *de re* reading, with $\exists$ outscoping ‘needs’. In the derivation below, the QP is activated first, and the goal formula matches the input s of the type for ‘someone’. In the abbreviated part of the derivation with proof term v', the input *np* of the QP is lifted to the $s/(np\backslash s)$ level required by the predicate ‘needs’.

$$\dfrac{\dfrac{\dfrac{\dfrac{(np \cdot \otimes \cdot (((np\backslash s)/(s/(np\backslash s))) \cdot \otimes \cdot np)) \overset{v'}{\vdash} s \mid \qquad \mid \boxed{s} \overset{\alpha}{\vdash} \boxed{s}}{((np \cdot \otimes \cdot (((np\backslash s)/(s/(np\backslash s))) \cdot \otimes \cdot np)) \cdot \oslash \cdot s) \vdash (s \oslash s) \mid}\oslash R}{\mid np \vdash (((np\backslash s)/(s/(np\backslash s))) \cdot \backslash \cdot (np \cdot \backslash \cdot ((s \oslash s) \cdot \oplus \cdot s)))}\rightleftharpoons}{\mid ((s \oslash s) \obslash np) \vdash (((np\backslash s)/(s/(np\backslash s))) \cdot \backslash \cdot (np \cdot \backslash \cdot s))}\obslash L'}{(np \cdot \otimes \cdot (((np\backslash s)/(s/(np\backslash s))) \cdot \otimes \cdot ((s \oslash \boxed{s}) \obslash np))) \vdash \boxed{s} \mid}\leftrightharpoons$$

$$\begin{array}{rl} & \mu\alpha.\langle\, \mathsf{someone} \mid \widetilde{\lambda}^l\beta.(\widetilde{\mu}y.\langle\, (v' \oslash \alpha) \mid \beta \,\rangle)\,\rangle \\ v' = & \mu\alpha_0.\langle\, \mathsf{needs} \mid ((\mu\gamma.\langle\, \mathsf{molly} \mid \gamma \,\rangle \setminus \alpha_0) \,/\, \lambda^r z.(\mu\beta_0.\langle\, z \mid (y \setminus \beta_0)\,\rangle))\,\rangle \\ \rightsquigarrow & \lambda c.(\exists\, \lambda y.(c\ ((\mathsf{needs}\ \lambda P.(P\ y))\ \mathsf{molly}))) \end{array} \tag{37}$$

Complement clauses and non-local construal The examples of scope ambiguity we have seen so far are realized within a single clausal domain. In the situation where a quantifier phrase occurs in an embedded clause, **LG** allows for the possibility of non-local scope construal. Examples would be sentences like ‘Alice described how every prisoner escaped’ (which could involve a description for every prisoner, or a description of a collective escape) or ‘Alice thinks someone is cheating’ (which could express Alice’s idea about a particular cheater, or her general suspicion that cheating is going on). Compare the local and non-local readings for ‘Molly thinks someone left’ in (38).

$$\begin{array}{rl} \text{local scope:} & \lambda c.(c\ ((\mathsf{thinks}\ \lambda c'.(c'\ (\exists\, \lambda x.(\mathsf{left}\ x))))\ \mathsf{molly})) \\ \text{non-local scope:} & \lambda c.(\exists\, \lambda x.(c\ ((\mathsf{thinks}\ \lambda c'.(c'\ (\mathsf{left}\ x)))\ \mathsf{molly}))) \end{array} \tag{38}$$

The local reading, as we saw, is the only one available in **NL** under the $s/(np\backslash s)$ type assignment for a QP. In **LG** it is obtained by providing the derivation for ‘someone left’, which we gave above as (32), as the complement for ‘think’. The non-local reading is computed below. The derivation activates $(s \oslash s) \obslash np$ first, and establishes an axiom link between the goal formula and the input *s* of the QP. The abbreviated part of the derivation then has a simple *np* subject for the complement of ‘think’: the input *np* subformula of the QP, which is bound by $\widetilde{\mu}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{(np\cdot\otimes\cdot(((np\backslash\boxed{s})/(s/(s\backslash s)))\cdot\otimes\cdot(np\cdot\otimes\cdot(np\backslash s))))\overset{v'}{\vdash}\boxed{s}\,|\qquad|\,\boxed{s}\overset{\alpha}{\vdash}\boxed{s}}{((np\cdot\otimes\cdot(((np\backslash s)/(s/(s\backslash s)))\cdot\otimes\cdot(np\cdot\otimes\cdot(np\backslash s))))\cdot\oslash\cdot s)\vdash(s\oslash s)\,|}\,\oslash R}{|\,np\vdash((((np\backslash s)/(s/(s\backslash s)))\cdot\backslash\cdot(np\cdot\backslash\cdot((s\oslash s)\cdot\oplus\cdot s)))\cdot/\cdot(np\backslash s))}\,\rightleftharpoons}{|\,((s\oslash s)\obslash np)\vdash((((np\backslash s)/(s/(s\backslash s)))\cdot\backslash\cdot(np\cdot\backslash\cdot s))\cdot/\cdot(np\backslash s))}\,\obslash L'}{(np\cdot\otimes\cdot(((np\backslash s)/(s/(s\backslash s)))\cdot\otimes\cdot(((s\oslash\boxed{s})\obslash np)\cdot\otimes\cdot(np\backslash s))))\vdash\boxed{s}\,|}\,\leftrightharpoons$$

$$\begin{aligned}&\mu\alpha.\langle\,\text{someone}\mid\widetilde{\lambda}^{l}\beta.(\widetilde{\mu}x.\langle\,(v'\oslash\alpha)\mid\beta\,\rangle)\,\rangle\\ v'=\;&\mu\alpha_1.\langle\text{thinks}\mid((\mu\gamma.\langle\text{molly}\mid\gamma\rangle\backslash\alpha_1)\,/\,\lambda^{r}x_2.(\mu\beta_1.\langle\text{left}\mid(x\backslash\widetilde{\mu}z_1.\langle x_2\mid(z_1\backslash\beta_1)\rangle)\rangle))\rangle\\ \rightsquigarrow\;&\lambda c.(\exists\,\lambda x.(c\;((\text{thinks}\;\lambda c'.(c'\;(\text{left}\;x)))\;\text{molly})))\end{aligned}\tag{39}$$

Depending on the choice of lexical predicate, the possibility for a non-local construal will not always be available for a quantifier phrase in an embedded clause. Some predicates selecting a clausal complement turn this complement into a scope *island* for embedded quantifier phrases. An example would be a sentence ‘Alice observed everyone slept’ which arguably has no reading involving individual observations for each sleeper, this in contrast with a monoclausal structure ‘Alice watched everyone sleep’. Our QP type assignment $(s\oslash s)\obslash np$ allows non-local scope for quantifiers in embedded clauses, as we have just seen, so a scope island will have to be explicitly imposed by the predicate selecting a complement clause. A general method for the imposition of island constraints, using unary modalities, is discussed in [24]. The technique can be readily imported in **LG**.

Choice points for focusing In the derivations we have presented so far, the choice of focusing on a QP formula $(s\oslash s)\obslash np$ and eliminating its main connective by means of $(\obslash L')$ was always immediately followed by the activation of the $s\oslash s$ subformula and application of the $(\oslash R)$ rule. The $(\obslash L')$ step binds the covariable for $s\oslash s$; the $(\obslash R)$ co-application step determines the scope.

Taking into account the full set of derivational choice points, we see that there is nothing that commits us to this particular sequence of steps: there can be rules intervening between the $(\obslash L')$ and $(\oslash R)$ steps that decompose the QP type in its atomic parts. Contrast the derivation in (40) below with (39). Both derivations start with the activation of the QP formula. But in (40) the $(\obslash L')$ step is followed by shifting the focus to the type for ‘needs’, which is decomposed in its main connective by $(/L)$. The effect on the axiom linking is that the goal formula is matched with the input *s* of ‘needs’, and that the input *s* of the QP is matched in the computation for the lifted direct object. The derivation in (40), in other words, corresponds to the *de dicto* reading, with ‘needs’ outscoping the QP. But substitution of the lexical specifications for $[\![\text{someone}]\!]$ or $\lfloor\!\lfloor\text{someone}\rfloor\!\rfloor$ in (33) in the cbv or cbn CPS image of (40) would wrongly produce the wide scope *de re* reading, because it determines the scope at the point where $(\obslash L')$ is applied, ignoring the contribution of the $(\oslash R)$ step.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{|\,np\overset{\gamma}{\vdash}np}{np\vdash np\,|}\,\leftrightharpoons\qquad|\,\boxed{s}\overset{\alpha}{\vdash}\boxed{s}}{|\,(np\backslash s)\vdash(np\cdot\backslash\cdot s)}\,\backslash L\qquad\begin{matrix}\vdots\\((s\oslash\boxed{s})\cdot\obslash\cdot np)\overset{v''}{\vdash}(\boxed{s}/(np\backslash s))\,|\end{matrix}}{|\,((np\backslash s)/(s/(np\backslash s)))\vdash((np\cdot\backslash\cdot s)\cdot/\cdot((s\oslash s)\cdot\obslash\cdot np))}\,/L}{|\,np\vdash((s\oslash s)\cdot\oplus\cdot(((np\backslash s)/(s/(np\backslash s)))\cdot\backslash\cdot(np\cdot\backslash\cdot s)))}\,\Leftarrow}{|\,((s\oslash s)\obslash np)\vdash(((np\backslash s)/(s/(np\backslash s)))\cdot\backslash\cdot(np\cdot\backslash\cdot s))}\,\obslash L'}{(np\cdot\otimes\cdot(((np\backslash\boxed{s})/(s/(np\backslash s)))\cdot\otimes\cdot((s\oslash s)\obslash np)))\vdash\boxed{s}\,|}\,\leftrightharpoons$$

$$\begin{aligned}&\mu\alpha.\langle\,\text{someone}\mid\widetilde{\lambda}^{l}\beta_0.(\widetilde{\mu}y.\langle\,\text{needs}\mid((\mu\gamma.\langle\,\text{molly}\mid\gamma\,\rangle\backslash\alpha)\,/\,v'')\,\rangle)\,\rangle\\ v''=\;&\lambda^{r}z_1.(\mu\alpha_1.\langle\,(\mu\beta_1.\langle\,z_1\mid(y\backslash\beta_1)\,\rangle\oslash\alpha_1)\mid\beta_0\,\rangle)\end{aligned}\tag{40}$$

There are two ways of dealing with this situation. The first is to leave the inference rules of Fig. 8 unchanged, and adjust the lexical specifications of $[\![\text{someone}]\!]$ and $\lfloor\!\lfloor\text{someone}\rfloor\!\rfloor$ in such a way that they properly reflect the derivational independence of the $(\obslash L')$ and $(\oslash R)$ steps. The second option would be to add a *primitive* 3-place connective to λ**LG**, with a pair of inference rules that in *one step* decompose the QP type in its three subtypes. Such a binding connective $q(A,B,C)$ has in fact been proposed in [21], but a complete logic for it has not been given. We leave the second option as a topic for further research and work out the first here.

The adjusted lexical specifications for cbv and cbn QPs in (41) are given in terms of a logical constant $\mathcal{E}$, rather than ∃. The ∃ constant denotes a set of sets of individuals, as we saw. The $\mathcal{E}$ constant instead denotes a set of sets of *pairs* of individuals (the *x* parameter) and functions from *s* values to *s* values (the c' parameter). Intuitively, the *x* parameter identifies the variable that will be bound; the c' parameter identifies the scopal domain where the quantificational effect is actually computed. We can use argument lowering *alow* and a *shift*$'$ combinator to write the cbv recipe in a more concise form.

$$\begin{array}{lrcl} (s \oslash s) \obslash np & \lceil\!\lceil \mathsf{someone} \rceil\!\rceil & = & \lambda Q.(\mathcal{E}\, \mathit{shift}'(\mathit{alow}\, Q)) \\ & \mathit{shift}' & = & \lambda h \lambda c'.(h\, \lambda c \lambda v.(c\ (c'\ v))) \\ & \mathit{alow} & = & \lambda f \lambda v.(f\ \lambda k.(k\, v)) \\ & & = & \lambda Q.(\mathcal{E}\ \lambda x.\lambda c'.((Q\ \lambda u.(u\ \lambda c.\lambda v.(c\ (c'\ v))))\ x)) \\ & \lfloor\!\lfloor \mathsf{someone} \rfloor\!\rfloor & = & \lambda Q.(\mathcal{E}\ \lambda x.\lambda c'.((Q\ \lambda k.(k\, x))\ \lambda c''.\lambda c.(c''\ \lambda v.(c\ (c'\ v))))) \end{array} \tag{41}$$

With the revised lexical specifications, the cbv/cbn interpretation of (40) correctly produces the *de dicto* reading, with c' in the scope of needs. (With $\mathcal{E}^{y}_{c'}\ \phi$ as syntactic sugaring for $\mathcal{E}\ \lambda y.\lambda c'.\phi$.) The *de re* interpretation of (39) has c' outscoping needs. The CPS image of the derivation (36) reflects the fact that the QP is activated after the verb ‘needs’. Also for this alternative way of computing the *de dicto* reading we find c' in the scope of needs.

$$\begin{array}{ll} \lceil(36)\rceil = \lfloor(36)\rfloor = & \lambda c.(c\ ((\mathsf{needs}\ \lambda P.(\mathcal{E}^{y}_{c'}\ (c'\ (P\, y))))\ \mathsf{molly})) \\ \lceil(40)\rceil = \lfloor(40)\rfloor = & \lambda c.(\mathcal{E}^{y}_{c'}\ (c\ ((\mathsf{needs}\ \lambda P.(c'\ (P\, y)))\ \mathsf{molly}))) \\ \lceil(39)\rceil = \lfloor(39)\rfloor = & \lambda c.(\mathcal{E}^{y}_{c'}\ (c\ (c'\ ((\mathsf{needs}\ \lambda P.(P\, y))\ \mathsf{molly})))) \end{array} \tag{42}$$

A model-theoretic interpretation of $\mathcal{E}$ requires that we change the intended denotation domains for s values. Instead of seeing them as denoting truth values, we can model them as *state transitions*, i.e. functions $\mathit{STATE} \rightarrow \mathit{STATE} \rightarrow \mathit{BOOL}$, the denotations for *dynamic* propositions. Spelling out $\mathcal{E}$ at the level of the lambda recipe, we can provide the program $\lambda\phi\lambda i\lambda j.(\exists k.(i[x]k \wedge (\phi\ k\, j)))$ for the c' parameter, effecting dynamic existential binding at the point where c' is applied. The syntactic position of $\mathcal{E}$ in the terms of (42) corresponds with the moment of activation in the proof; for the computation of the semantic value $[\cdot]$, this procedural bit of information is passed over.

$$i\,[(\mathcal{E}^{x}_{c}\ \phi[(c\ \psi[x])])]\,j \quad = \quad i\,[\,\phi[\ (\exists x.\psi[x])\ ]\,]\,j \tag{43}$$

5. Conclusions, further directions

The move from asymmetric Lambek calculus to the symmetric Lambek–Grishin system opens up a number of complementary perspectives on linguistic resources: terms versus contexts; implications versus co-implications; call-by-value versus call-by-name computations. In **LG** we have realized these dualities in a resource-sensitive and structure-sensitive way. The type we have assigned to quantifier phrases — $(s \oslash s) \obslash np$ — puts together the same pieces of information as the **NL** type $s/(np\backslash s)$, but it ‘packages’ this information in a novel way. As a result, we have been able to make a distinction between the local behavior of a QP, contributing an *np* resource in building the antecedent structure, and its scope-taking effects, when the $(s \oslash s)$ component is activated.

Our approach differs in a number of respects from the related work cited in Section 1. Abstracting away from the directionality issue, de Groote’s original application of $\lambda\mu$ calculus to scope construal in [10] syntactically types a QP as *np* with meaning representation $\mu\alpha(\exists\, \alpha)$, with α a *np* continuation. As a result, a sentence with multiple QPs is associated with a *unique* parse/proof term; the multiple readings for that term are obtained as a result of the non-confluence of the $\lambda\mu$ calculus, which is considered as a feature, not a bug. Our approach does not exploit the non-confluence: in parsing a sentence we fix either the cbv or the cbn strategy. Scope ambiguities are the effect of the different choices one can make as to when the connectives in a QP type are activated. The semantic analyses based on continuations in the work of Barker and Shan are formulated in the setting of ‘intuitionistic’ type-logical grammars, with a single succedent formula. To obtain scope flexibility, these authors in [4,28] rely on multimodal structural postulates. In future work, we hope to investigate to what extent their analyses can be recast in terms of **LG** and whether the symmetries of that system give rise to streamlined analyses.

References

[1] G. Allwein, J.M. Dunn, Kripke models for Linear Logic, Journal of Symbolic Logic, 58 (2) (1993) 514–545.

[2] C. Barker, Continuations and the nature of quantification, Natural language semantics 10 (2002) 211–242.

[3] C. Barker, Continuations in natural language, in: H. Thielecke (Ed.), CW’04: Proceedings of the Fourth ACM SIGPLAN Continuations Workshop, Technical Report CSR-04-1, School of Computer Science, University of Birmingham, pp. 1–11.

[4] C. Barker, C. Shan, Types as graphs: Continuations in type logical grammar, Journal of Logic, Language and Information 15 (4) (2006) 331–370.

[5] C. Barker, C. Shan, Donkey anaphora is in-scope binding, Semantics and Pragmatics 1 (1) (2008) 1–46.

[6] A. Bastenhof, Continuations in Natural Language Syntax and Semantics, Master Thesis M.Phil. Linguistics, Utrecht University, 2009.

[7] R. Bernardi, M. Moortgat, Continuation semantics for symmetric categorial grammar, in: D. Leivant, R. de Queiros (Eds.), Proceedings of the 14th Workshop on Logic, Language, Information and Computation (WoLLIC’07), LNCS, vol. 4576, Springer, 2007, pp. 53–71.

[8] J.R.B. Cockett, R.A.G. Seely, Weakly distributive categories, in: Journal of Pure and Applied Algebra, University Press, 1993, pp. 45–65.

[9] P. Curien, H. Herbelin, Duality of computation, in: International Conference on Functional Programming (ICFP’00), 2000, pp. 233–243 [2005: corrected version].

[10] P. de Groote, Type raising, continuations, and classical logic, in: M.S.R. van Rooy (Ed.), Proceedings of the Thirteenth Amsterdam Colloquium, ILLC, Universiteit van Amsterdam, 2001, pp. 97–101.

[11] K. Došen, A brief survey of frames for the Lambek calculus, Zeitschrift für mathematischen Logik und Grundlagen der Mathematik 38 (1992) 179–187.

[12] N. Galatos, P. Jipsen, T. Kowalski, H. Ono, Residuated Lattices: An Algebraic Glimpse at Substructural Logics, vol. 151 (Studies in Logic and the Foundations of Mathematics), Elsevier, Amsterdam, 2007.

[13] R. Goré, Substructural logics on display, Logic Journal of IGPL 6 (3) (1997) 451–504.
[14] V. Grishin, On a generalization of the Ajdukiewicz-Lambek system, in: Mikhailov (Ed.), Studies in Nonclassical Logics and Formal Systems, Moscow, pp. 315–334, Nauka [English translation in Abrusci and Casadio (Eds.) Proceedings 5th Roma Workshop, Bulzoni Editore, Roma, 2002] (1983).
[15] H. Hendriks, Studied Flexibility. Categories and Types in Syntax and Semantics. Ph.D. thesis, ILLC, Amsterdam University, 1993.
[16] N. Kurtonina, M. Moortgat, Relational semantics for the Lambek–Grishin calculus, in: C. Ebert, G. Jaeger, J. Michaelis (Eds.), The mathematics of language. Proceedings of the 10th and 11th Biennial Conference. Springer. LNAI, in press.
[17] J. Lambek, The mathematics of sentence structure, American Mathematical Monthly 65 (1958) 154–170.
[18] J. Lambek, On the calculus of syntactic types, in: R. Jakobson (Ed.), Structure of Language and its Mathematical Aspects, American Mathematical Society, 1961, pp. 166–178.
[19] J. Lambek, From categorial to bilinear logic, in: K. Došen, P. Schröder-Heister (Eds.), Substructural Logics, Oxford University Press, 1993, pp. 207–237.
[20] S. Lengrand, Call-by-value, call-by-name, and strong normalization for the classical sequent calculus, in: B. Gramlich, S. Lucas (Eds.), Post-proceedings of the Third International Workshop on Reduction Strategies in Rewriting and Programming (WRS'03), Electronic Notes in Theoretical Computer Science, vol. 86, Elsevier, 2003.
[21] M. Moortgat, Generalized quantifiers and discontinuous type constructors, in: H. Bunt, A. van Horck (Eds.), Discontinuous Constituency, De Gruyter, 1996, pp. 181–207.
[22] M. Moortgat, Symmetries in natural language syntax and semantics: the Lambek–Grishin calculus, in: D. Leivant, R. de Queiros (Eds.), Proceedings 14th Workshop on Logic, Language, Information and Computation (WoLLIC'07), LNCS, vol. 4576, Springer, 2007.
[23] M. Moortgat, Symmetric categorial grammar, Journal of Philosophical Logic 38 (6) (2009) 681–710.
[24] G. Morrill, Type Logical Grammar, Kluwer, 1994
[25] G. Morrill, M. Fadda, Valentin, Nondeterministic discontinuous Lambek calculus, in: Proceedings of the Seventh International Workshop on Computational Semantics (IWCS7), Tilburg, 2007.
[26] C. Sacerdoti-Coen. Explanation in natural language of lambda-mu-comu terms, in: Mathematical Knowledge Management, 4th International Conference, MKM 2005, Bremen, Germany, July 15–17, 2005, Revised Selected Papers, Lecture Notes in Computer Science, vol. 3863, Springer, 2006, pp. 234–249.
[27] P. Selinger, Control categories and duality: on the categorical semantics of the lambda-mu calculus, Mathematical Structures in Computer Science 11 (2001) 207–260.
[28] C. Shan, Linguistic Side Effects, Ph.D. thesis, Harvard University, 2005.
[29] J. van Benthem, The Semantics of Variety in Categorial Grammar, Technical Report 83-29, Simon Fraser University, Burnaby (BC), Revised version in W. Buszkowski, W. Marciszewski, J. van Benthem (Eds.), Categorial Grammar, Benjamin, Amsterdam, 1988.
[30] P. Wadler, Monads and composable continuations, Higher-Order and Symbolic Computation 7 (1) (1994) 39–55.
[31] P. Wadler, Call-by-value is dual to call-by-name, in: C. Runciman, O. Shivers (Eds.), Proceedings of the Eighth ACM SIGPLAN International Conference on Functional Programming, ICFP 2003, Uppsala, Sweden, August 25–29, 2003, pp. 189–201.