J Philos Logic (2009) 38:681–710
DOI 10.1007/s10992-009-9118-6

Symmetric Categorial Grammar

Michael Moortgat

Received: 22 May 2009 / Accepted: 31 August 2009 / Published online: 16 October 2009

Abstract The Lambek-Grishin calculus is a symmetric version of categorial grammar obtained by augmenting the standard inventory of type-forming operations (product and residual left and right division) with a dual family: coproduct, left and right difference. Interaction between these two families is provided by distributivity laws. These distributivity laws have pleasant invariance properties: stability of interpretations for the Curry-Howard derivational semantics, and structure-preservation at the syntactic end. The move to symmetry thus offers novel ways of reconciling the demands of natural language form and meaning.

Keywords Categorial grammar · Lambek calculus · Lambda calculus · Curry-Howard correspondence · Substructural logic

> ...the stability of meaningfulness under an amount of permutation (whether grammaticality is lost or not), is a phenomenon which a linguistic semantics ought to *explain*, rather than ignore. ([48], p. 213)

In the early 1980s, Johan van Benthem's work on the Lambek calculus lured me into categorial temptations: I have never regretted this. As for the symmetric developments, it was a pleasure to collaborate with Raffaella Bernardi, Natasha Kurtonina, and Mati Pentus on some of the results reported on here; Richard Moot and Philippe de Groote provided valuable feedback at different stages of the project. For comments on an earlier draft I thank Arno Bastenhof, Sylvain Pogodalla, Sylvain Salvati, and two anonymous reviewers. Remaining errors are my own.

M. Moortgat (✉)
Utrecht University, OTS, Trans 10, 3512 JK Utrecht, Netherlands
e-mail: Michael.Moortgat@phil.uu.nl

1 Introduction

Semantic interpretation in categorial grammar takes the form of a compositional translation: a homomorphism sends types and derivations of a syntactic source calculus to types and proofs of a semantic target calculus; the latter, by the Curry-Howard proofs-as-programs interpretation, are isomorphic to terms of the λ calculus, which one reads as 'denotational recipes' for the parsed expressions [47]. Grammatical composition is resource-sensitive; this makes the Lambek-Van Benthem calculus **LP** (or Multiplicative Intuitionistic Linear Logic) the natural target logic for meaning assembly. But the marriage of natural language *form* and *meaning* is not without tensions. Categorial calculi that adequately deal with syntactic well-formedness lack semantic expressivity: desirable interpretations available in **LP** are unobtainable as the image of derivations that respect grammatical word order and/or phrase structure.

The limited semantic expressivity manifests itself most clearly in the area of discontinuous dependencies. The examples below can informally introduce the issues at stake.

a A solution is needed
b John is afraid a disaster is unavoidable
c an article that appeared yesterday
d an article that John hopes will appear before the end of the year (1)

In (a), one finds a combination of a subject quantifier phrase and a verb phrase. The sentence has two readings: one expressing the need for some solution or other, the second reading stating the existence of a particular solution which is needed. In the most basic categorial calculus, the system **NL** of [31], these readings correlate with two derivations for the sentence, given types $s/(np\backslash s)$ and $(s/(np\backslash s))\backslash s$ for subject and verb phrase. The (b) example, likewise, has two readings, interpreting 'a disaster' locally within the complement clause, or giving it wide scope ('there is a disaster x such that John is afraid x is unavoidable'). But this time, the wide scope reading cannot be obtained in **NL**, or in the more flexible **L** of [30], from the $s/(np\backslash s)$ type assignment to 'a disaster'. In (c) and (d), we see examples of what in generative grammar would be called movement. The relative pronoun 'that' in (c) binds the subject of the relative clause body 'x appeared yesterday'. A syntactic type assignment $(n\backslash n)/(np\backslash s)$ to 'that' correctly produces this interpretation. But it cannot establish such a dependency with respect to the embedded subject 'John hopes x will appear before the end of the year' in (d).

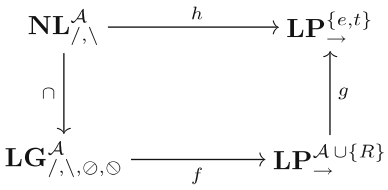
The conflicting demands of linguistic form and meaning give rise to the question: *What are the structural transformations under which interpretations are stable?* Multimodal typelogical grammars answer this question by adding structural modalities that provide facilities for reconfiguring the grammatical material in a controlled way; examples of such work are [36, 41]. In the Discontinuous Lambek Calculi of Morrill and co-workers the categorial ontology is enriched with expressions consisting of detached parts ('split' strings) that

wrap themselves around the phrases they combine with, rather than composing via concatenation; see for example [42].

In this paper, I report on a line of work that extends categorial grammar along a different axis. The point of departure is the generalization of the Ajdukiewicz-Lambek calculus proposed in [27], where in addition to the familiar product operation $\otimes$ and its residual implications one finds a dual coproduct $\oplus$ with residual co-implications. The extended vocabulary leads to derivability judgements of the form $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$, i.e. the intuitionistic ban on multiple conclusions on the right of the turnstile has been lifted. A Curry-Howard interpretation of derivations in the symmetric calculus naturally leads to a dual view on linguistic resources. In conformity with the original Lambek calculi, the patterning of the hypotheses A_i in the antecedent is used to model the syntactic composition of phrases out of their constituent parts. The configuration of the conclusions B_j in the succedent, on the other hand, expresses composition of *evaluation contexts* (continuations) for the semantic values associated with the composed phrases. A key ingredient of Grishin's work are the *distributivity laws* that govern the interaction between the product and coproduct operations. These laws allow for the interleaving of hypotheses A_i and conclusions B_j, and thus for the *synchronization* of the composition of phrases and contexts. Syntactically, Grishin's distributivities are *structure preserving*: they respect the (non-commutative, non-associative) character of the operations they combine. Semantically, interpretations are *invariant* under the distributivity laws (and, of course, under cut elimination).

Outline In Fig. 1 we contrast the standard categorial view on the syntax-semantics interface with the alternative we develop in this paper. For the standard view, let us take **NL** as a representative of the syntactic source calculus. For the typing of lexical items, **NL** has at its disposal a set of atomic types $\mathcal{A}$, and types built out of these atoms by means of the directional implications / and \. Types and derivations of the source calculus are mapped to the semantic target calculus **LP** by a homomorphism h which translates the directional implications into the linear implication, $h(A\backslash B) = h(B/A) = h(A) \to h(B)$, and interprets the syntactic atoms in terms of the semantic primitives e and t for entities and truth values. In Section 2, we depart from this standard view, and introduce the symmetric calculus **LG**. The symmetric calculus contains **NL** as a proper subsystem; the inventory of type-forming operations, in addition to the directional implications, now also includes the dual co-implications $\oslash$, $\otimes$. In

Fig. 1 Outline of the paper



order to obtain a decision procedure, we present **LG** in the format of a display sequent calculus in Section 3. We then turn to the syntax-semantics interface in Sections 4 and 5. The target calculus for **LG** is the same as for **NL**, but we reach the target now in two steps. The first step, discussed in Section 4, is to transform the multiple conclusion source calculus **LG** back into single conclusion **LP** by means of a continuation-passing-style (CPS) translation, the mapping f in Fig. 1. The CPS translation sends values of type A to their 'double negation' $(A \to R) \to R$, where R is a distinguished target language type, the type of responses. In Section 5 we then discuss a second mapping g that connects the image of the CPS translation to $\mathbf{LP}_{\to}^{\{e,t\}}$. At the type level, g agrees with h on the primitive types in $\mathcal{A}$, and maps the response type R to t. At the level of derivations, g 'lowers' the double-negation CPS proof terms to the level of the proof terms of $\mathbf{LP}_{\to}^{\{e,t\}}$. We can then show, first, that in moving from **NL** to **LG**, nothing has been lost: for the **NL** fragment of **LG**, the composition of the mappings f and g produces the same recipes for meaning assembly as the direct interpretation given by h. Secondly, for derivations that exploit the new type-forming operations $\oslash$, $\otimes$, we can establish the kind of dependencies illustrated in Eq. 1 (b), (d) that are beyond the reach of asymmetric **NL**. We conclude with a survey of some formal expressivity and complexity results for **LG** in Section 6.

Background, related work Grishin's 1983 paper, published in a Russian anthology, was initially ignored in the West. It came to the attention of a wider public during the Tübingen conference on substructural logics organized by Došen and Schröder-Heister in 1990. In the proceedings of that conference, [32] gives the first thorough presentation of this work, and expresses his scepticism as to the linguistic applicability. Grishin's interaction principles, which play a key role in what follows, have been intensively studied from a category theoretic perspective; see the work on 'weak' or linear distributive categories, for example [14, 21]. At the Rome workshops organized in the late 1990s by Abrusci and Casadio, Lambek introduced Grishin's work to a linguistic audience; an English translation by his student Čubric facilitated access to the source for researchers lacking fluency in Russian. The impetus for the actual linguistic exploration of symmetric calculi was given by the course on categorial grammar and display logic taught by Bernardi and Goré at ESSLLI 2004. Below I review results that have been obtained since then and add some new.

Two cautionary remarks to close this introduction. Grishin 1983 is a *modular* architecture for generalizing Lambek calculi. The system **LG**, the particular mix I describe in the following pages, is dictated by the linguistic agenda, and differs from choices made elsewhere with other motivations. As for terminology: I don't use the term 'classical' for **LG**, reserving that term for systems where $\otimes$ and $\oplus$ are related by de Morgan dualities, via an involutive negation [19], or cancelling pre- and post-negations, as in classical bilinear logic [1, 32]. Casadio [12] has argued for the use of such classical systems for linguistic analysis. Compact bilinear logic is obtained by identifying the $\otimes$ and

$\oplus$ operations; Lambek's pregroup grammars are based on this idea; see [33] for a recent exposition.

2 Symmetry, Distributivity

Recall the essentials of Lambek's typelogical programme. A categorial *grammar* consists of a universal and a language-specific part. The universal component is a *syntactic calculus* freely generated from a given set of atomic types (say, s for sentences, np for noun phrases, n for common nouns, ...). The set of types is defined inductively from these atoms by some family of type-forming operations; for a start, let these be product $\otimes$, left $\backslash$ and right $/$ division. The proofs of the syntactic calculus are freely generated from axioms by a set of logical and structural inference rules, to be specified below. A categorial *lexicon* is a relation associating each word with a finite number of types. Given a lexicon Lex, a categorial grammar assigns type B to a string of words $w_1 \cdots w_n$ iff for $1 \leq i \leq n$, $(w_i, A_i) \in Lex$, and $X \to B$ is provable in the syntactic calculus where X is a product formula with yield $A_1 \cdots A_n$. The grammar can be considered adequate if the strings $w_1 \cdots w_n$ are indeed judged to be wellformed expressions of type B.

From this recipe, a hierarchy of syntactic calculi unfolds, depending on the logical and, maybe, structural properties one attributes to the type-forming operations. The system **NL**, the basis of this hierarchy, has only *logical* axioms, and transitivity is the only structural rule. The inference rules of Eq. 3 state that the type-forming operations form a residuated triple. In **NL** types are assigned to *phrases* (bracketed strings). The calculus **L** is obtained from **NL** by adding a non-logical associativity axiom for $\otimes$ (or an equivalent structural inference rule); **LP** adds both associativity and commutativity. These then are logics of *strings* and *multisets*, respectively.

$$A \to A; \qquad \text{from } A \to B \text{ and } B \to C \text{ infer } A \to C \qquad \text{preorder laws} \quad (2)$$

$$A \to C/B \quad \text{iff} \quad A \otimes B \to C \quad \text{iff} \quad B \to A\backslash C \qquad \text{residuation laws} \quad (3)$$

In [27], the above hierarchy is symmetrically extended by adding a family of type-forming operations obeying the *dual* residuation laws of Eq. 4: a coproduct $\oplus$ together with left $\oslash$ and right $\odot$ difference operations. As for the notation: the subtracted quantity is under the circled slash: $A \odot B$ is read "A minus B", $B \oslash A$ "B from A". Like **NL**, the minimal symmetric system, which we refer to as $\mathbf{LG}_{\emptyset}$, does not attribute any extra structural properties to $\oplus, \odot, \oslash$.

$$B \oslash C \to A \quad \text{iff} \quad C \to B \oplus A \quad \text{iff} \quad C \odot A \to B \qquad \text{dual residuation laws} \quad (4)$$

The residuation patterns of Eqs. 3 and 4 give rise to two kinds of symmetry: for the left-right symmetry relating the directional slashes and difference

operations we write $\cdot^{\bowtie}$; for the arrow-reversal symmetry that relates the residuated and the dual residuated families we write $\cdot^{\infty}$. For atomic types p, $p^{\bowtie} = p = p^{\infty}$. For complex types the definitions are given by the bidirectional translation tables in Eq. 5. The tables succinctly represent a tedious list of definitional equations $(C/D)^{\bowtie} = D^{\bowtie}\backslash C^{\bowtie}$, $(D\backslash C)^{\bowtie} = C^{\bowtie}/D^{\bowtie}$, ...

$$\bowtie \quad \begin{array}{cccc} C/D & A\otimes B & B\oplus A & D\oslash C \\ \hline\hline D\backslash C & B\otimes A & A\oplus B & C\oslash D \end{array} \qquad \infty \quad \begin{array}{ccc} C/B & A\otimes B & A\backslash C \\ \hline\hline B\oslash C & B\oplus A & C\oslash A \end{array} \tag{5}$$

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.

$$A^{\bowtie} \to B^{\bowtie} \quad \text{iff} \quad A \to B \quad \text{iff} \quad B^{\infty} \to A^{\infty} \tag{6}$$

The connections between residuation theory and substructural logics have been well studied; see [24] for a recent survey. The following properties are useful for an understanding of the rest of the paper.

Compositions Consider the operations $A\otimes -$ and $A\backslash -$, i.e. multiplication to the left and left division by some fixed type A. The fact that these operations form a residuated pair means that the composition $(A\otimes -)(A\backslash -)$ is contracting; the composition $(A\backslash -)(A\otimes -)$ is expanding. From the two symmetries, we obtain the patterns in Eq. 7; the columns are related by $\cdot^{\infty}$. the rows by $\cdot^{\bowtie}$.

$$\begin{array}{ll} A\otimes(A\backslash B)\to B\to A\backslash(A\otimes B) & (B/A)\otimes A\to B\to(B\otimes A)/A \\ (B\oplus A)\oslash A\to B\to(B\oslash A)\oplus A & A\oslash(A\oplus B)\to B\to A\oplus(A\oslash B) \end{array} \tag{7}$$

Monotonicity From the (dual) residuation laws, one obtains monotonicity rules, allowing one to infer from $A\to B$ and $C\to D$ any of the following: $A\otimes C\to B\otimes D$; $A/D\to B/C$; $D\backslash A\to C\backslash B$; $A\oplus C\to B\oplus D$; $A\oslash D\to B\oslash C$; $D\oslash A\to C\oslash B$. The tonicity properties of the type-forming operations are summarized in the schema $(\uparrow\otimes\uparrow)$, $(\uparrow/\downarrow)$, $(\downarrow\backslash\uparrow)$, $(\uparrow\oplus\uparrow)$, $(\uparrow\oslash\downarrow)$, $(\downarrow\oslash\uparrow)$.

Relational semantics The type language can be given an interpretation along the lines of the Kripke semantics for modal logics and Routley-Meyer semantics for relevance logic. In this semantics, the n-ary type-forming operations are interpreted in terms of $(n+1)$-ary relations. Models for the symmetric calculus are based on frames $F = (W, R_{\otimes}, R_{\oplus})$, where W is the set of 'grammatical resources', and $R_{\otimes}$ and $R_{\oplus}$ the ternary relations interpreting the connectives of

the $\otimes$ and $\oplus$ families. A valuation V assigns subsets of W to the type formulas, respecting the truth conditions of Eqs. 8 and 9 for complex formulas.

$$\begin{aligned} x \Vdash A \otimes B &\text{ iff } \exists yz.R_{\otimes}xyz \text{ and } y \Vdash A \text{ and } z \Vdash B \\ y \Vdash C/B &\text{ iff } \forall xz.(R_{\otimes}xyz \text{ and } z \Vdash B) \text{ implies } x \Vdash C \\ z \Vdash A\backslash C &\text{ iff } \forall xy.(R_{\otimes}xyz \text{ and } y \Vdash A) \text{ implies } x \Vdash C \end{aligned} \tag{8}$$

$$\begin{aligned} x \Vdash A \oplus B &\text{ iff } \forall yz.R_{\oplus}xyz \text{ implies } (y \Vdash A \text{ or } z \Vdash B) \\ y \Vdash C \oslash B &\text{ iff } \exists xz.R_{\oplus}xyz \text{ and } z \nVdash B \text{ and } x \Vdash C \\ z \Vdash A \osl C &\text{ iff } \exists xy.R_{\oplus}xyz \text{ and } y \nVdash A \text{ and } x \Vdash C \end{aligned} \tag{9}$$

In the relevance logic community, the ternary relations $R_{\otimes}$ and $R_{\oplus}$ interpret the binary operations known as *fusion* and *fission*. Using terminology more familiar from generative grammar, we will refer to $R_{\otimes}$ as the Merge relation. The multiplicative conjunction $\otimes$ is interpreted as an existential modality with respect to Merge; the residual / and \ operations are the corresponding universal modalities. For the coproduct $\oplus$ and its residuals, a dual situation obtains: the multiplicative disjunction $\oplus$ here is the universal modality with respect to the co-Merge relation $R_{\oplus}$; the co-implications are the corresponding existential modalities.

Soundness and completeness of $\mathbf{LG}_{\emptyset}$ is established in [29]: $A \to B$ is provable in $\mathbf{LG}_{\emptyset}$ iff for all frames F and valuations V, $V(A) \subseteq V(B)$. In the canonical model, worlds are construed as *weak filters*, i.e. sets of formulas closed under derivability. The key point to note here is that the semantics for the minimal system $\mathbf{LG}_{\emptyset}$ does not impose any restrictions on the interpretation of Merge and co-Merge: as long as one does not consider interaction principles, $R_{\otimes}$ and $R_{\oplus}$ are distinct relations. $\mathbf{LG}_{\emptyset}$ in this respect differs from **CNL**, the 'classical' version of **NL** studied in [19]. The latter system has tensor and par operations, and an involutive negation that relates them in terms of de Morgan duality.

There are many variations on the semantics discussed here. Reading $R_{\otimes}xyz$ as $x = y \cdot z$ where $\cdot$ is a multiplicative operation, not necessarily associative, one obtains the powerset residuated groupoid interpretation of **NL**, see [11]. A second groupoid operation, say $+$, can similarly interpret the operations of the $\oplus$ family, with $R_{\oplus}xyz$ now seen as $x = y + z$. Next to the 'intensional' multiplicative operations considered in this paper, one can add an 'extensional' vocabulary for the lattice operations. For the relational interpretation, these require more structured interpretants, as can be found in the Generalized Kripke Frames of [25] or in the Generalized Galois Logics of [8]. The symmetric situation is discussed in [9].

Interaction: distributivity principles The symmetric vocabulary creates the opportunity for interactions between the product and the coproduct families. We are interested in interactions that respect the non-associative,

non-commutative character of the multiplicative operations they combine. Let us call such interaction *structure-preserving*.

In [27], the interaction principles take the form of distributivity *postulates* (i.e. non-logical axioms) added to the pure residuation logic $\mathbf{LG}_\emptyset$. In Eq. 10 below, we give an equivalent presentation in the form of *inference rules*. Grishin's postulates, in their many interderivable forms, are easily derived from Eq. 10. More importantly, in Section 3, we will be able to recast the rules of Eq. 10 in a sequent format that allows for decidable proof search.

Grishin's interaction principles come in two groups: the rules of Eq. 10, and their inverses, where premise and conclusion of the rules change place. Each of these choices satisfies the structure-preservation requirement, but their combination does not, as we will see. Consider first the rules of Eq. 10. They are obtained from the following blueprint: from $A \otimes B \to C \oplus D$, select a product and a coproduct term, and *simultaneously* introduce the residual operations for the remaining two terms. The four ways of realizing this recipe are related by the $\cdot^{\bowtie}$ symmetry: selecting any one of them as a starting point, the remaining three are obtained by taking the $\cdot^{\bowtie}$ image of either the $\otimes$ operation and its residual, or the $\oplus$ operation and its residual, or both.

$$\frac{A \otimes B \;\to\; C \oplus D}{C \oslash A \;\to\; D \,/\, B}\;(\oslash, /) \qquad \frac{A \otimes B \;\to\; C \oplus D}{B \oslash D \;\to\; A \setminus C}\;(\oslash, \setminus)$$

$$\frac{A \otimes B \;\to\; C \oplus D}{A \oslash D \;\to\; C \,/\, B}\;(\oslash, /) \qquad \frac{A \otimes B \;\to\; C \oplus D}{C \oslash B \;\to\; A \setminus D}\;(\oslash, \setminus) \tag{10}$$

The rules in Eq. 10 are formulated in such a way that they fully factor out the (dual) residuation principles. Applying these principles, one derives the original form of Grishin's postulates. We illustrate in Eq. 11 with some consequences of the $(\oslash, /)$ rule.

$$(C \oplus D)/B \;\to\; (C/B) \oplus D \qquad (A \oslash D) \otimes B \;\to\; (A \otimes B) \oslash D$$
$$A\backslash(C \oplus D) \;\to\; (A \oslash D)\backslash C \qquad (C/B) \oslash A \;\to\; C \oslash (A \otimes B) \tag{11}$$

As indicated above, Grishin's paper also introduces the *inverses* of the rules in Eq. 10 as an orthogonal choice for the interaction between the product and the coproduct families. Characteristic principles now are Eq. 12. These principles appear in [22] under the name of *hemi-distributivity* laws; their semantic content within the setting of generalized Galois logics is investigated in [9].

$$(C \oplus B) \otimes A \;\to\; C \oplus (B \otimes A) \qquad A \otimes (B \oplus C) \;\to\; (A \otimes B) \oplus C$$
$$A \otimes (C \oplus B) \;\to\; C \oplus (A \otimes B) \qquad (B \oplus C) \otimes A \;\to\; (B \otimes A) \oplus C \tag{12}$$

The addition to $\mathbf{LG}_\emptyset$ of both Eq. 10 and the inverses no longer satisfies the structure-preservation requirement. It is shown in [6] that this combination introduces associative and commutative perturbations in the individual $\otimes$ and

$\oplus$ families. In the remainder of this paper, we will use the extension of the minimal symmetric system $\mathbf{LG}_{\emptyset}$ with the irreversible distributivity rules in Eq. 10, and we refer to this calculus as **LG**. For the linguistic application, this is not really a limitation: we will see that **LG** is expressive enough to handle both the semantic and the syntactic type of discontinuous dependency illustrated in Eq. 1.

In the relational semantics of [29] and [9] one accommodates the distributivity laws by imposing appropriate frame constraints connecting the interpretation of the $R_{\otimes}$ and $R_{\oplus}$ relations. A more satisfactory semantics would endow the interpretants of **LG** formulas with a mathematical structure rich enough for the distributivity principles to arise naturally. It would be interesting in this respect to see whether the categorical semantics of Cockett and Seely for linear distributive categories (see for example [14] and subsequent work) can be tailored for **LG**. In the semantics for linear distributive categories the two tensor operations $\otimes, \oplus$ are taken as associative (possibly also commutative) operations, and the distributivity laws are collateral to these assumptions. Commutativity/associativity of $\otimes, \oplus$ is incompatible with the linguistic agenda pursued here.

3 LG as a Display Logic

Given types A and B, the *decision problem* for a categorial calculus is the question whether $A \to B$ is a theorem, in other words, whether there is a proof for the type transformation turning A into B. For the original calculi **(N)L**, Lambek solved this problem affirmatively by casting these systems in the format of Gentzen-style sequent calculi, and showing that the cut rule—the sequent version of transitivity—can be eliminated. The remaining logical inference rules each introduce a connective in the conclusion. The desired decision procedure is then obtained by finitely searching for proofs in backward-chaining fashion.

Sequents for **(N)L** are statements of the form $X \vdash A$, where X is a structure built out of formulas by means of a structure-building operation matching the multiplicative conjunction $\otimes$. In Gentzen's classical system, sequents have the form $X \vdash Y$, and we have two structural operations: a conjunctive one building the antecedent X, and a disjunctive one for the succedent structure Y. It is not clear how we could adapt the Gentzen format to the situation of **LG**: the distributivity rules, as we saw in Eq. 10, simultaneously introduce one of the implications $/, \backslash$ and one of the coimplications $\oslash, \obslash$. This is incompatible with the format of Gentzen's *logical* rules, which involve a single logical connective, but also with his *structural* rules, which are formulated purely in terms of the structural operations. The way out of this conundrum is to cast the sequent calculus for **LG** in the format of a Display Logic: in Display Logic, for each logical connective there is a matching structural operation. This turns the distributivity laws into well-behaved structural rules that do not interfere with the cut elimination process.

A comprehensive presentation of the landscape of substructural logics as display calculi can be found in [26]. Goré discusses symmetric Lambek calculus and (a selection of) Grishin's distributivity laws. Below we amend Goré's presentation in two respects. First, in anticipation of the Curry-Howard interpretation of **LG** to be discussed in Section 4, we make the distinction between *active* and *passive* occurrences of formulas explicit by adding inference rules for activating a passive formula. Second, we replace the partial account of the Grishin distributivity laws by the (display version) of the full set of rules in Eq. 10.

We comment on Figs. 2 and 3. *Formulas* $A, B, \ldots$ are built out of atomic formulas by means of the logical connectives $\otimes, /, \backslash, \oplus, \oslash, \oblong$; *structures* $X, Y, \ldots$ are built out of atomic parts by means of structural connectives. Atomic parts of structures are formulas. As indicated above, each logical connective has a matching structural connective. We use the same symbols for logical and structural connectives. We borrow a typographic practice from Latin inscriptions and use the midpoint to demarcate the components making up a structure. Compare $c/(a\backslash b)$, which is a formula, built from atomic formulas a, b, c with the logical connectives $/, \backslash$ and $\cdot c \cdot / \cdot a \backslash b \cdot$ which is a structure built out of two atomic parts (formulas) $\cdot c \cdot$ and $\cdot a \backslash b \cdot$ with the structural slash

Axioms, cut

$$\frac{}{\cdot A \cdot \vdash A}\ axiom_r \qquad \frac{X \vdash A \quad A \vdash Y}{X \vdash Y}\ cut \qquad \frac{}{A \vdash \cdot A \cdot}\ axiom_l$$

Focusing

$$\frac{X \vdash \cdot A \cdot}{X \vdash A}\ focus_r \qquad \frac{\cdot A \cdot \vdash Y}{A \vdash Y}\ focus_l$$

The reason the unfocussing rules are not present in this axiomatisation is because they can be derived by means of an axiom and a cut.

(Dual) residuation laws

$$\frac{\dfrac{Y \vdash X \cdot \backslash \cdot Z}{X \cdot \otimes \cdot Y \vdash Z}\ r}{X \vdash Z \cdot / \cdot Y}\ r \qquad \frac{\dfrac{Z \cdot \oslash \cdot X \vdash Y}{Z \vdash Y \cdot \oplus \cdot X}\ dr}{Y \cdot \oblong \cdot Z \vdash X}\ dr$$

Distributivity laws

$$\frac{X \cdot \otimes \cdot Y \vdash Z \cdot \oplus \cdot W}{X \cdot \oslash \cdot W \vdash Z \cdot / \cdot Y}\ d\oslash/ \qquad \frac{X \cdot \otimes \cdot Y \vdash Z \cdot \oplus \cdot W}{Y \cdot \oslash \cdot W \vdash X \cdot \backslash \cdot Z}\ d\oslash\backslash$$

$$\frac{X \cdot \otimes \cdot Y \vdash Z \cdot \oplus \cdot W}{Z \cdot \oblong \cdot X \vdash W \cdot / \cdot Y}\ d\oblong/ \qquad \frac{X \cdot \otimes \cdot Y \vdash Z \cdot \oplus \cdot W}{Z \cdot \oblong \cdot Y \vdash X \cdot \backslash \cdot W}\ d\oblong\backslash$$

Fig. 2 **LG** as a display calculus. Axioms, cut, focusing; structural rules

$$\frac{X \vdash A \quad B \vdash Y}{A\backslash B \vdash X \cdot \backslash \cdot Y}\ \backslash L \qquad \frac{X \vdash A \quad B \vdash Y}{X \cdot \oslash \cdot Y \vdash A \oslash B}\ \oslash R$$

$$\frac{X \vdash A \quad B \vdash Y}{B/A \vdash Y \cdot / \cdot X}\ /L \qquad \frac{X \vdash A \quad B \vdash Y}{Y \cdot \otimes \cdot X \vdash B \otimes A}\ \otimes R$$

$$\frac{X \vdash A \cdot \backslash \cdot B}{X \vdash A\backslash B}\ \backslash R \qquad \frac{B \cdot \oslash \cdot A \vdash X}{B \oslash A \vdash X}\ \oslash L$$

$$\frac{X \vdash B \cdot / \cdot A}{X \vdash B/A}\ /R \qquad \frac{A \cdot \otimes \cdot B \vdash X}{A \otimes B \vdash X}\ \otimes L$$

$$\frac{X \vdash A \quad Y \vdash B}{X \cdot \otimes \cdot Y \vdash A \otimes B}\ \otimes R \qquad \frac{B \vdash Y \quad A \vdash X}{B \oplus A \vdash Y \cdot \oplus \cdot X}\ \oplus L$$

$$\frac{A \cdot \otimes \cdot B \vdash Y}{A \otimes B \vdash Y}\ \otimes L \qquad \frac{X \vdash B \cdot \oplus \cdot A}{X \vdash B \oplus A}\ \oplus R$$

Fig. 3 **LG** as a display calculus. Introduction rules for the connectives

connective. We omit outermost midpoints wherever this can be done without introducing ambiguity.

We distinguish input (antecedent) and output (succedent) structures; they are built out of atomic constituents (formulas) by the grammar rules below.

$$\begin{aligned} \text{(input)}\ \mathcal{S}^\bullet &::= \cdot\mathcal{F}\cdot \mid \mathcal{S}^\bullet \cdot \otimes \cdot \mathcal{S}^\bullet \mid \mathcal{S}^\circ \cdot \otimes \cdot \mathcal{S}^\bullet \mid \mathcal{S}^\bullet \cdot \oslash \cdot \mathcal{S}^\circ \\ \text{(output)}\ \mathcal{S}^\circ &::= \cdot\mathcal{F}\cdot \mid \mathcal{S}^\circ \cdot \oplus \cdot \mathcal{S}^\circ \mid \mathcal{S}^\bullet \cdot \backslash \cdot \mathcal{S}^\circ \mid \mathcal{S}^\circ \cdot / \cdot \mathcal{S}^\bullet \end{aligned} \tag{13}$$

Active/passive formulas The formulas out of which structures are built are *passive*. Sequents can have at most one *active* formula. As a result, we have three types of sequents: $X \vdash A$ (active succedent formula); $A \vdash X$ (active antecedent formula); $X \vdash Y$ (no active formula). We call the latter structural sequents. Axiomatic sequents come in two forms, depending on whether the active formula is in the antecedent or in the succedent. Cut cancels matching succedent and antecedent active formulas and results in a structural sequent. Structural sequents are the domain of application of structural rules: the (dual) residuation laws (aka display postulates) and the distributivity laws. We comment on the latter below. With the aid of the structural rules, one can display any atomic structural component (i.e. passive formula) as the single antecedent or succedent part, depending on its polarity. The focusing rules then activate the displayed formula.

Introduction rules There are two kinds of introduction rules for the logical connectives. The one-premise rules put together an active formula out of two passive sister formulas by replacing the structural connective that takes them together in the premise by the matching logical one in the conclusion.

These rules are *invertible*. The two-premise rules have an active formula in each of their premises; the conclusion builds an active formula out of these, introducing matching logical and structural connectives in the antecedent and in the succedent. We recognize the monotonicity principles of the arrow presentation.

Decidability Using the cut rule, it is easy to show that the sequent presentation and the two-formula arrow presentation of Section 2 are equivalent: for every arrow $A \to B$ there is a sequent proof of $\cdot A \cdot \vdash \cdot B \cdot$ (or the focused forms $A \vdash \cdot B \cdot, \cdot A \cdot \vdash B$) and for every sequent proof $X \vdash Y$ ($X \vdash B$, $A \vdash Y$), there is an arrow $\overline{X} \to \overline{Y}$ ($\overline{X} \to B$, $A \to \overline{Y}$), where overlining indicates replacement of the structural connectives by their logical counterpart. For the display calculus one can then show that cuts can be eliminated, except for the 'innocent' cuts where one of the premises is axiomatic. Decidability of **LG** follows as a corollary.

The cut-elimination procedure for **LG** targets 'topmost' cuts: cut inferences which themselves are derived without cut steps. One distinguishes two cases: (i) the cut formula is introduced in the two premises of the cut inference by means a *logical* rule; (ii) the cut formula in one of the premises of the cut inference is introduced by a *focusing* rule. In the case of (i), the cut formula is of the form $A\#B$ for some connective #. The one-premise logical rule decomposes that formula into passive subformulas A and B. We can transform this type of cut into cuts on the subformulas A and B, by displaying the passive A and B and then activating them by means of a focusing rule. The output of the transformation is of lower complexity (the # connective has gone). In Eq. 14, we illustrate with $\oplus$. The $\otimes$ case is obtained by arrow reversal. The (co)implication cases are given in Section 4 below, together with their redex-contractum image in the term language for **LG**.

$$\dfrac{\dfrac{X \vdash A \cdot \oplus \cdot B}{X \vdash A \oplus B}\,\oplus R \quad \dfrac{A \vdash Y \quad B \vdash Z}{A \oplus B \vdash Y \cdot \oplus \cdot Z}\,\oplus L}{X \vdash Y \cdot \oplus \cdot Z}\,cut$$

$$\rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{X \vdash A \cdot \oplus \cdot B}{X \cdot \oslash \cdot B \vdash \cdot A \cdot}}{X \cdot \oslash \cdot B \vdash A} \quad A \vdash Y}{X \cdot \oslash \cdot B \vdash Y}}{X \vdash Y \cdot \oplus \cdot B}}{Y \cdot \oslash \cdot X \vdash \cdot B \cdot}}{Y \cdot \oslash \cdot X \vdash B} \qquad B \vdash Z}{Y \cdot \oslash \cdot X \vdash Z}}{X \vdash Y \cdot \oplus \cdot Z} \tag{14}$$

Case (ii), the situations where the cut formula in one of the premises is introduced by a focusing rule, is dealt with as follows. In the proof of the premise

that ends with the focusing step, one replaces the *axiom* that introduces the passive A formula by the other cut premise. One thus obtains the conclusion of the original cut inference without the focusing and cut steps. In Eq. 15, we illustrate with the left focusing rule. By arrow reversal, one obtains the right focusing case.

$$\dfrac{X \vdash A \qquad \dfrac{\begin{array}{c}\cdot A \cdot \vdash A \\ \vdots \\ \cdot A \cdot \vdash Y\end{array}}{A \vdash Y}\, focus_l}{X \vdash Y}\, cut \quad \rightsquigarrow \quad \dfrac{\begin{array}{c}X \vdash A \\ \vdots \\ X \vdash Y\end{array}} \tag{15}$$

Notice that, where cases (i) and (ii) do not apply, we also have the situation where one of the premises of a cut inference is a (left or right) *axiom*. These cuts are not eliminated: in fact, we need them to *shift* the focus from one active formula to another. The cuts with an axiom premise do not form a threat for decidability: they are consistent with finite backward-chaining proof search in terms of subformulae of the end sequent. The use of axiomatic cuts for shifting focus can be encapsulated in *derived* inference rules. In Eq. 16 we derive the rule for shifting from an antecedent to a succedent active formula as an example. The open premise has active formula A. By means of a cut against an axiom, we deactivate this formula, obtaining a structural sequent. The vertical dots represent a sequence of structural steps (using the (dual) residuation rules and distributivity rules) transforming $\cdot A \cdot \vdash Y$ into $X \vdash \cdot B \cdot$. The final step then activates the B formula. In all, we have four focus shifting rules of this type (the shift from lhs to rhs of Eq. 16, and shifts rhs/lhs, rhs/rhs, lhs/rhs). In a similar way, we can close the one-premise connective introduction rules $(/R)$, $(\backslash R)$, $(\oslash L)$, $(\otimes L)$, $(\oplus R)$ under structural inferences, shifting (backward-chaining) directly from the active formula introduced by the rule to some active formula in the premise. Using these derived inference rules, a proof consists of logical rules applied to active formulas, and steps that shift the focus to select a new active formula.

$$\dfrac{\dfrac{\dfrac{}{\cdot A \cdot \vdash A}\, ax_r \qquad A \vdash Y}{\cdot A \cdot \vdash Y}\, cut}{\begin{array}{c}\vdots \\ X \vdash \cdot B \cdot\end{array}}\, focus_r \quad \rightsquigarrow \quad \dfrac{A \vdash Y}{X \vdash B} \leftrightharpoons \tag{16}$$

4 From LG to LP: CPS Translation

Let us turn now to the Curry-Howard interpretation of **LG** derivations and to the embedding of **LG** into single-conclusion **LP**. In [7], the term language for **LG** derivations is a customized version of the $\bar{\lambda}\mu\tilde{\mu}$ calculus of [28] and [16].

The $\bar{\lambda}\mu\tilde{\mu}$ term calculus is isomorphic to LK, Gentzen's sequent calculus for classical logic. It is embedded into the λ calculus (isomorphic to natural deduction proofs in Intuitionistic Logic) via a continuation-passing-style translation. In the case of **LG**, the source logic is resource sensitive: its derivations are coded by *linear* terms. The CPS translation itself respects resource sensitivity; as a result, the target logic of the translation is **LP** i.e. MILL, Multiplicative Intuitionistic Linear Logic. We refer to the term calculus for **LG** derivations as λ**LG**. Apart from the linearity restriction, we also need a refinement of $\bar{\lambda}\mu\tilde{\mu}$ to reflect the fact that in **LG** we are dealing with *directional* (co)implications; the λ**LG** term language has to discriminate between the constructs for / and \, and between those for $\oslash$ and $\oslash$. In the rules below, we adjust the presentation of [7] to the display format of Section 3.

The language of λ**LG** makes a threefold distinction, corresponding to the three types of sequents. Sequents with an active succedent formula correspond to *terms*. For terms, we write $x, v, \ldots$. Sequents with an active antecedent formula correspond to evaluation *contexts* (or: environments, coterms). For contexts we write $\alpha, e, \ldots$. For structural sequents, we have *commands* (c). Passive input formulas are labeled with variables $x, y, \ldots$; passive output formulas are labeled with covariables $\alpha, \beta, \ldots$. Each (co)variable has a unique occurrence. In the rules below, we write the proof term as a superscript of the turnstile.

We comment on the rules. In the resource-sensitive setting of **LG**, we can picture a term as a *producer* of a resource (a 'value'), and a context as a *consumer* of this resource. The command $\langle v \mid e \rangle$, associated with the cut rule, is the *transaction* between consumer and producer. In a command, we have reached an equilibrium between supply and demand: there is a balance between the available resources (the input assumptions) and their consumers (the output co-assumptions). We disturb the equilibrium by activating an input assumption, creating a surplus on the supply side, or an output co-assumption, creating a surplus on the demand side. These states of disequilibrium are what makes the computation proceed.

$$x : A \overset{x}{\vdash} A \qquad \frac{X \overset{v}{\vdash} A \quad A \overset{e}{\vdash} Y}{X \overset{\langle v|e \rangle}{\vdash} Y}\ cut \qquad A \overset{\alpha}{\vdash} \alpha : A$$

$$\frac{X \overset{c}{\vdash} \alpha : A}{X \overset{\mu\alpha.c}{\vdash} A}\ \mu \qquad \frac{x : A \overset{c}{\vdash} Y}{A \overset{\tilde{\mu}x.c}{\vdash} Y}\ \tilde{\mu}$$

The rules above apply to formulas in general; they do not mention any connectives. Let us turn then to the introduction rules for the logical connectives. We restrict attention to the (co)implicational fragment here. Consider first the two-premise rules for /, \ and $\oslash$, $\oslash$. To bring out the symmetries, we have written the rules in such a way that they start from identical premises: they combine exactly the same pieces of information—a term v of type A and a context e of type B. Bringing these together on the left of the turnstile, we

create an environment for the consumption of the value associated with a slash formula, $A\backslash B$ or B/A, depending on whether this consist of our term of type A followed by the context of type B, or the context followed by the term. We overload the notation, and write $v\backslash e$ and e/v, so that it is clear what the associated types are. We can also put together v and e on the right of the turnstile. This results in terms $v \oslash e$ and $e \osl v$, i.e. producers of $A \oslash B$ and $B \osl A$ values.

$$\frac{X \overset{v}{\vdash} A \quad B \overset{e}{\vdash} Y}{A\backslash B \overset{v\backslash e}{\vdash} X \cdot \backslash \cdot Y}\ \backslash L \qquad \frac{X \overset{v}{\vdash} A \quad B \overset{e}{\vdash} Y}{X \cdot \oslash \cdot Y \overset{v\oslash e}{\vdash} A \oslash B}\ \oslash R$$

$$\frac{B \overset{e}{\vdash} Y \quad X \overset{v}{\vdash} A}{B/A \overset{e/v}{\vdash} Y \cdot / \cdot X}\ / L \qquad \frac{B \overset{e}{\vdash} Y \quad X \overset{v}{\vdash} A}{Y \cdot \osl \cdot X \overset{e\osl v}{\vdash} B \osl A}\ \osl R$$

The matching one-premise rules give us the means to restore the balance between supply and demand for (co)implication formulas. The premise of these rules is a command, where an A assumption and a B co-assumption occur in construction with each other. In the conclusion for the implication rules, we create terms for active succedent formulas $A\backslash B$ or B/A by simultaneously binding the A variable and the B covariable. In a symmetric fashion, we can create contexts for active antecedent formulas $A \oslash B$ and $B \osl A$ by simultaneously binding variable and covariable with the dual $\widetilde{\lambda}$ binder.

$$\frac{X \overset{c}{\vdash} x : A \cdot \backslash \cdot \beta : B}{X \overset{\lambda(x,\beta).c}{\vdash} A\backslash B}\ \backslash R \qquad \frac{x : A \cdot \oslash \cdot \beta : B \overset{c}{\vdash} X}{A \oslash B \overset{\widetilde{\lambda}(x,\beta).c}{\vdash} X}\ \oslash L$$

$$\frac{X \overset{c}{\vdash} \beta : B \cdot / \cdot x : A}{X \overset{\lambda(\beta,x).c}{\vdash} B/A}\ / R \qquad \frac{\beta : B \cdot \osl \cdot x : A \overset{c}{\vdash} X}{B \osl A \overset{\widetilde{\lambda}(\beta,x).c}{\vdash} X}\ \osl L$$

The proof terms for $(\backslash R)$, $(/R)$, $(\oslash L)$, $(\osl L)$ are given here in a format that is in Curry-Howard correspondence with the display rules, which replace a structural connective by the matching logical one. A more standard construction would have $\lambda x^A.v^B$, $\widetilde{\lambda}\beta^B.e^A$, where the premises have active B and A respectively. To encode the difference between $/, \backslash$ and $\osl, \oslash$ in these constructs, one would have to distinguish between left and right abstractions λ^l and λ^r, similarly for co-abstraction.

Computation rules Curien and Herbelin [16] show that interpretation in $\overline{\lambda}\mu\widetilde{\mu}$ is invariant under reduction/cut-elimination. For the (co)implications of directional λ**LG**, the correspondence between cut-elimination steps and redex/contractum rewriting is as in Eq. 17. As an example, the image of

replacement of a cut on $A\backslash B$ by cuts on the subformulas A and B is shown in Eq. 18, abbreviating the structural steps a little. Further removal of the μ and $\widetilde{\mu}$ redexes would give $c[x \leftarrow v, \beta \leftarrow e]$. The other cases are obtained by left-right symmetry and arrow reversal.

$$\begin{array}{lrcl}
(\backslash) & \langle \lambda(x,\beta).c \mid v \backslash e \rangle & \rightsquigarrow & \langle v \mid \widetilde{\mu}x.\langle \mu\beta.c \mid e \rangle \rangle \\
(/) & \langle \lambda(\beta,x).c \mid v / e \rangle & \rightsquigarrow & \langle v \mid \widetilde{\mu}x.\langle \mu\beta.c \mid e \rangle \rangle \\
(\oslash) & \langle v \oslash e \mid \widetilde{\lambda}(x,\beta).c \rangle & \rightsquigarrow & \langle \mu\beta.\langle v \mid \widetilde{\mu}x.c \rangle \mid e \rangle \\
(\otimes) & \langle e \otimes v \mid \widetilde{\lambda}(\beta,x).c \rangle & \rightsquigarrow & \langle \mu\beta.\langle v \mid \widetilde{\mu}x.c \rangle \mid e \rangle \\
(\mu) & \langle \mu\alpha.c \mid e \rangle & \rightsquigarrow & c[\alpha \leftarrow e] \\
(\widetilde{\mu}) & \langle v \mid \widetilde{\mu}x.c \rangle & \rightsquigarrow & c[x \leftarrow v]
\end{array} \tag{17}$$

$$\dfrac{\dfrac{X \overset{c}{\vdash} x : A \cdot \backslash \cdot \beta : B}{X \overset{\lambda(x,\beta).c}{\vdash} A\backslash B} \quad \dfrac{Y \overset{v}{\vdash} AB \overset{e}{\vdash} Z}{A\backslash B \overset{v\backslash e}{\vdash} Y \cdot \backslash \cdot Z}}{X \vdash Y \cdot \backslash \cdot Z} \quad \rightsquigarrow \quad \dfrac{Y \overset{v}{\vdash} A \quad \dfrac{\dfrac{X \overset{c}{\vdash} x : A \cdot \backslash \cdot \beta : B}{x : A \cdot \otimes \cdot X \overset{\mu\beta.c}{\vdash} B} \mu \quad B \overset{e}{\vdash} Z}{x : A \cdot \otimes \cdot X \overset{\langle \mu\beta.c \mid e \rangle}{\vdash} Z} \widetilde{\mu}}{A \overset{\widetilde{\mu}x.\langle \mu\beta.c \mid e \rangle}{\vdash} Z \cdot / \cdot X}}{Y \overset{\langle v \mid \widetilde{\mu}x.\langle \mu\beta.c \mid e \rangle \rangle}{\vdash} Z \cdot / \cdot X}}{X \vdash Y \cdot \backslash \cdot Z} \tag{18}$$

Continuation-passing-style translation For the translation from our source logic **LG** into the target logic **LP** we have a choice between a call-by-value $\lceil \cdot \rceil$ and a call-by-name $\lfloor \cdot \rfloor$ strategy. We focus on the call-by-value strategy in what follows. The call-by-name version, obtained via the arrow-reversal duality $\lfloor A \rfloor = \lceil A^{\infty} \rceil$, is worked out in [7].

Depending on the targeted fragment of Multiplicative Intuitionistic Linear Logic, the CPS translation can be defined in a number of ways. The version in Eqs. 19, 20 below is based on [34], extended with clauses for the co-implications. This translation stays within the *implication* fragment of **LP**/MILL. The target language for the translation in [16] has conjunction (pairs) in addition to negation (implication with range R).

Consider first the action of $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ on **LG** *types*. The target logic has the same atomic types as the source logic, plus a distinguished type R, the response type. As was the case for the direct interpretation of Lambek derivations, the CPS translation identifies left and right (co)implications at the level of the target logic. Under the call-by-value interpretation, for every type A in the source logic in the target logic we distinguish *values* of type A, $\lceil A \rceil$; *continuations*, i.e. functions from values to the response type, $\lceil A \rceil \rightarrow R$; and

computations, i.e. functions from continuations to the response type, $(\ulcorner A \urcorner \to R) \to R$. As an abbreviation, we write $A^\perp$ for $A \to R$. For atoms p, $\ulcorner p \urcorner = p$. The dual call-by-name translation is given on the right in Eq. 19. Here $\llcorner A \lrcorner$ represents a *continuation* of type A; $\llcorner A \lrcorner \to R$ then is a *computation*.

$$\begin{aligned}
\ulcorner B/A \urcorner = \ulcorner A \backslash B \urcorner &= \ulcorner B \urcorner^\perp \to \ulcorner A \urcorner^\perp \\
\ulcorner B \oslash A \urcorner = \ulcorner A \oslash B \urcorner &= \ulcorner A \backslash B \urcorner^\perp \\
&= (\ulcorner B \urcorner^\perp \to \ulcorner A \urcorner^\perp)^\perp \\
\llcorner A \oslash B \lrcorner = \llcorner B \oslash A \lrcorner &= \llcorner B \lrcorner^\perp \to \llcorner A \lrcorner^\perp \\
\llcorner A/B \lrcorner = \llcorner B \backslash A \lrcorner &= \llcorner B \oslash A \lrcorner^\perp \\
&= (\llcorner B \lrcorner^\perp \to \llcorner A \lrcorner^\perp)^\perp
\end{aligned} \tag{19}$$

The effect of the CPS translation on λ**LG** proof terms is given in Eq. 20, for the $\ulcorner \cdot \urcorner$ regime. Terms are mapped to computations; contexts to continuations. The leading abstraction, then, for the translation of terms, is over a *continuation* variable; in the case of contexts, the abstraction is over a variable for a *value* of the relevant type. We write $\widetilde{x}$ ($\widetilde{\alpha}$) for the target language (co)variable corresponding to source language x (α). The dual $\llcorner \cdot \lrcorner$ translation is left as an exercise.

$$\begin{array}{lrl}
\text{(terms)} & \ulcorner x \urcorner & = \lambda k.(k\ \widetilde{x}) \\
& \ulcorner \lambda(x,\beta).c \urcorner = \ulcorner \lambda(\beta,x).c \urcorner & = \lambda k.(k\ \lambda\widetilde{\beta}\lambda\widetilde{x}.\ulcorner c \urcorner) \\
& \ulcorner v \oslash e \urcorner = \ulcorner e \oslash v \urcorner & = \lambda k.(k\ \lambda u.(\ulcorner v \urcorner\ (u\ \ulcorner e \urcorner))) \\
& \ulcorner \mu\alpha.c \urcorner & = \lambda\widetilde{\alpha}.\ulcorner c \urcorner \\
\text{(contexts)} & \ulcorner \alpha \urcorner & = \widetilde{\alpha} \quad (= \lambda x.(\widetilde{\alpha}\ x)) \\
& \ulcorner v \backslash e \urcorner = \ulcorner e/v \urcorner & = \lambda u.(\ulcorner v \urcorner\ (u\ \ulcorner e \urcorner)) \\
& \ulcorner \widetilde{\lambda}(x,\beta).c \urcorner = \ulcorner \widetilde{\lambda}(\beta,x).c \urcorner & = \lambda u.(u\ \lambda\widetilde{\beta}\lambda\widetilde{x}.\ulcorner c \urcorner) \\
& \ulcorner \widetilde{\mu}x.c \urcorner & = \lambda\widetilde{x}.\ulcorner c \urcorner \\
\text{(commands)} & \ulcorner \langle v \mid e \rangle \urcorner & = (\ulcorner v \urcorner\ \ulcorner e \urcorner)
\end{array} \tag{20}$$

The CPS translation is a Montagovian *compositional* mapping in the sense of Eq. 21, Lengrand's Preservation of Types theorem adjusted to the directional situation of **LG**. We write $X^\bullet$, X° for input resp. output literals of structure X. Recall that for grammars based on **LG** we consider sequents $X \vdash s$ where s is the start symbol, and X is a pure product structure with yield $A_1, \ldots, A_n$,

the A_i being the types of the lexical constants making up the sentence under consideration. A call-by-value interpretation, for such a sequent, maps the *values* associated with the lexical items to a *s* computation; the call-by-name interpretation maps *computations* for the lexical assumptions to a *s* computation.

$$\begin{array}{lrll}
\text{source: } \mathbf{LG}^{\mathcal{A}}_{/,\backslash,\oslash,\obslash} \xrightarrow{\text{CPS translation}} & & & \text{target: } \mathbf{LP}^{\mathcal{A}\cup\{R\}}_{\rightarrow} \\
X \overset{v}{\vdash} B & \lceil X^{\bullet}\rceil, & \lceil X^{\circ}\rceil^{\perp} & \vdash \lceil v\rceil : \lceil B\rceil^{\perp\perp} \\
 & \lfloor X^{\bullet}\rfloor^{\perp}, & \lfloor X^{\circ}\rfloor & \vdash \lfloor v\rfloor : \lfloor B\rfloor^{\perp} \\
A \overset{e}{\vdash} Y & \lceil Y^{\bullet}\rceil, & \lceil Y^{\circ}\rceil^{\perp} & \vdash \lceil e\rceil : \lceil A\rceil^{\perp} \\
 & \lfloor Y^{\bullet}\rfloor^{\perp}, & \lfloor Y^{\circ}\rfloor & \vdash \lfloor e\rfloor : \lfloor A\rfloor^{\perp\perp} \\
X \overset{c}{\vdash} Y & \lceil X^{\bullet}\rceil, \lceil Y^{\bullet}\rceil, & \lceil X^{\circ}\rceil^{\perp}, \lceil Y^{\circ}\rceil^{\perp} & \vdash \lceil c\rceil : R \\
 & \lfloor X^{\bullet}\rfloor^{\perp}, \lfloor Y^{\bullet}\rfloor^{\perp}, & \lfloor X^{\circ}\rfloor, \lfloor Y^{\circ}\rfloor & \vdash \lfloor c\rfloor : R
\end{array} \tag{21}$$

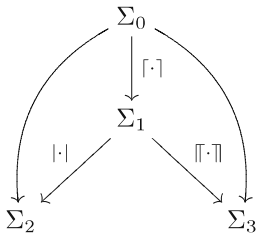
5 From CPS Translations to $\mathbf{LP}^{\{e,t\}}_{\rightarrow}$ Terms

We are ready now to complete the picture of Fig. 1 by connecting the image of the CPS translation to the terms of $\mathbf{LP}^{\{e,t\}}_{\rightarrow}$ coding meaning composition under the direct interpretation of categorial derivations. Our aim in this section is twofold. First, for the **NL** fragment of **LG**, we show that the proof terms of the direct interpretation can be recovered from the CPS terms by means of a mapping $[\![\cdot]\!]$ that lowers the double negation CPS proof terms. Second, for derivations that rely on the new connectives $\oslash, \obslash$, we show that we can establish dependencies beyond the reach of **NL**.

At the outset of this discussion, it will be useful to draw the line between *derivational* and *lexical* semantics. Derivational semantics is the focus of this article: its objects are the proof terms for the various calculi studied here. These proof terms are linear, i.e. resource-sensitive; 'constants' at the level of the derivational semantics are simply variables that remain unbound. In the target logic $\mathrm{LP}^{\{e,t\}}_{\rightarrow}$, one will ultimately substitute lambda terms expressing the lexical semantics of the 'logical' part of the vocabulary. These lexical recipes are no longer required to respect linearity constraints: in the recipes for determiners 'all'/'some', for example, the logical constants $\exists$, $\forall$ would doubly bind an *e*-type variable in the nominal restriction and the verb phrase interpretation. Such non-linear aspects of lexical semantics are beyond the scope of this paper.

Let us turn then to the $[\![\cdot]\!]$ mapping, which defines the derivational constants of the CPS translation in terms of the corresponding constants of the direct interpretation as given by the mapping h in Fig. 1. We will present $[\![\cdot]\!]$ here from the perspective of Abstract Categorial Grammar (ACG) [17, 20]—after all, the target of the CPS translation is the language where ACG higher order linear signatures are defined. For comparison with [44] we also give an ACG-style interpretation $|\cdot|$ extracting the generated *string* from the abstract CPS

proof term, obtaining the mappings in (24). In the case of λ**LG**, Σ_0, as the starting point for these mappings, is a (non-commutative, non-associative) *bilinear* signature, with linear implications and co-implications. The abstract syntax of the ACG analysis of **NL** in [44] is given in terms of a 'regular' linear higher order signature, capturing the derivation trees of **NL** proofs.



(22)

For concreteness, assume a tiny **LG** source lexicon with entries 'Mary' : np, 'left' : $np\backslash s$, 'teases': $(np\backslash s)/np$, 'everybody' : $s/(np\backslash s)$, 'someone' : $(s \oslash s) \oslash np$. The latter two entries will allow us to contrast a Lambek-style type assignment for subject quantifier phrases with the **LG** assignment of [7] that allows non-local construal.

Consider first the higher order linear signature Σ_1. The set of atoms $\mathcal{A}_1$ is $\{np, s, R\}$, with R the distinguished response type of the CPS target language. For lexical constants 'c' of type A in the **LG** source language, at the CPS level Σ_1, we have constants c of type $\lceil A\rceil$. The terms, similarly, are restricted to the $\lceil\cdot\rceil$ images of **LG** derivations: a *proper sublanguage* of the **LP** terms over the vocabulary, reflecting the non-associative, non-commutative regime of the source logic.

$$
\begin{aligned}
\Sigma_1 \;= (\{np, s, R\}, \{&\mathsf{mary}, \mathsf{left}, \mathsf{teases}, \mathsf{everybody}, \mathsf{someone}\},\\
&\{\mathsf{mary} \mapsto np,\\
&\quad \mathsf{left} \mapsto s^\perp \to np^\perp,\\
&\quad \mathsf{teases} \mapsto (s^\perp \to np^\perp)^\perp \to np^\perp,\\
&\quad \mathsf{everybody} \mapsto s^\perp \to (s^\perp \to np^\perp)^\perp,\\
&\quad \mathsf{someone} \mapsto ((s^\perp \to s^\perp)^{\perp\perp} \to np^\perp)^\perp \quad \} \quad)
\end{aligned}
$$

From the terms $\lceil v\rceil$ of the goal type $s^{\perp\perp}$ (i.e. s computations) associated with the **LG** analysis of a sentence, we want to obtain $\mathbf{LP}^{\{e,t\}}_{\to}$ terms and (after substitution of lexical recipes for the derivational constants) readings for a Montague-style modeltheoretic interpretation. Consider then the signature Σ_3, with $\mathcal{A}_3 = \{e, t\}$, denoting individuals and truth values, respectively. The non-logical constants from C_3 are given below; for constants 'c' of type A at the source level Σ_0, we now have c (smallcaps) of type $h(A)$. Assume we also have the usual logical constants at our disposal, specifically $\exists, \forall$. These are in fact

lexical recipes, as we saw, but we sidestep the non-linearity issues by restricting our small sample grammar to full quantifier phrases and pretending the domain consists entirely of persons.

$$\begin{aligned}\Sigma_3 \;= (&\{e, t\}, \{\textsc{mary}, \textsc{left}, \textsc{teases}\},\\ &\{\textsc{mary} \mapsto e,\\ &\quad \textsc{left} \mapsto e \to t,\\ &\quad \textsc{teases} \mapsto e \to (e \to t) \quad \} \quad)\end{aligned}$$

An ACG interpretation $\mathcal{L}_{13} : \Sigma_1 \to \Sigma_3$ is a pair of functions (η_{13}, θ_{13}) compatible with the typing relation: η_{13} interprets the atoms of Σ_1 as linear implicative types built over $\mathcal{A}_3$; θ_{13} interprets the source constants of Σ_1 as linear λ terms built upon Σ_3. For the model-theoretic interpretation we are after, η_{13} maps the response type R to the truth value type t. In the schema above, we have written θ_{13} as $[\![\cdot]\!]$. We spell out these functions for the sample lexical entities below.

$$\begin{aligned}\mathcal{L}_{13} \;= (&\{np \mapsto e, s \mapsto t, R \mapsto t\},\\ &\{\mathsf{mary} \mapsto \textsc{mary},\\ &\quad \mathsf{left} \mapsto \lambda c.\lambda x.(c\ (\textsc{left}\ x)),\\ &\quad \mathsf{teases} \mapsto \lambda v.\lambda y.(v\ \lambda c.\lambda x.(c\ ((\textsc{teases}\ y)\ x))),\\ &\quad \mathsf{everybody} \mapsto \lambda c.\lambda v.(c\ (\forall\ \lambda x.((v\ id)\ x))),\\ &\quad \mathsf{someone} \mapsto \lambda h.(\exists\ \lambda x.((h\ \lambda u.(u\ id))\ x)) \quad \} \quad)\end{aligned}$$

To extract a *string* from a CPS term, consider the signature Σ_2. There is one atomic type, σ; strings are typed as linear functions $\sigma \to \sigma$. All the lexical constants of Σ_2 are of type string.

$$\begin{aligned}\Sigma_2 = (&\{\sigma\}, \{\mathit{mary}, \mathit{left}, \mathit{teases}, \mathit{everybody}, \mathit{someone}\},\\ &\{\mathit{mary} \mapsto \sigma \to \sigma,\\ &\quad \mathit{left} \mapsto \sigma \to \sigma,\\ &\quad \mathit{teases} \mapsto \sigma \to \sigma,\\ &\quad \mathit{everybody} \mapsto \sigma \to \sigma,\\ &\quad \mathit{someone} \mapsto \sigma \to \sigma \quad \} \quad)\end{aligned}$$

The mapping $\mathcal{L}_{12}$ sends the atoms of Σ_1 to the string type $\sigma \to \sigma.\ \theta_{12}$, the interpretation of the constants, is written as $|\cdot|$ in the schema above. In the terms, $\cdot$ stands for composition (concatenation of strings).

$$\begin{aligned}\mathcal{L}_{12} = (\{&np \mapsto \sigma \to \sigma, s \mapsto \sigma \to \sigma, R \mapsto \sigma \to \sigma\},\\ &\{\mathsf{mary} \mapsto \mathit{mary},\\ &\quad \mathsf{left} \mapsto \lambda c.\lambda x.(c\ (x \cdot \mathit{left})),\\ &\quad \mathsf{teases} \mapsto \lambda v.\lambda y.(v\ \lambda c.\lambda x.(c\ (x \cdot \mathit{teases} \cdot y))),\\ &\quad \mathsf{everybody} \mapsto \lambda c.\lambda v.(c\ ((v\ \mathit{id})\ \mathit{everybody}))),\\ &\quad \mathsf{someone} \mapsto \lambda h.((h\ \lambda u.(u\ \mathit{id}))\ \mathit{someone})\quad \}\quad)\end{aligned}$$

To illustrate the combination of the mappings discussed, consider the sentence below, containing a regular Lambek type assignment $s/(np\backslash s)$ for the subject and the **LG** assignment proposed in [7] for the direct object. The Lambek type is restricted to the construction of local scope readings for its surface occurrence; the **LG** type is syntactically compatible with any *np* position, and allows for the construction of local and non-local scope readings.

$$\underbrace{s/(np\backslash s)}_{\mathsf{everybody}} \cdot \otimes \cdot \underbrace{((np\backslash s)/np}_{\mathsf{teases}} \cdot \otimes \cdot \underbrace{(s \oslash s) \oslash np)}_{\mathsf{someone}} \overset{v}{\vdash} s$$

In Eqs. 23 and 24, one finds two proof terms v of type s associated with derivations at the level of the source logic **LG**. We take their CPS image under cbv, $\ulcorner v \urcorner$ of type $s^{\perp\perp}$. We compose with the $\mathcal{L}$ mappings, and evaluate (ε), i.e. we supply the indentity function ($t \to t$ or *string* $\to$ *string*) to obtain the final result. The two **LG** proofs produce distinct semantic readings; the extracted string in the two cases is the same.

$$\begin{aligned}v &= \mu\alpha_0.\langle\ \mathsf{everybody} \mid (\alpha_0\ /\ \lambda(y_0, \gamma_0).\langle\ \mathsf{someone} \mid\\ &\qquad \widetilde{\lambda}(\alpha_1, y_1).\langle\ (\mu\gamma_1.\langle\ \mathsf{teases} \mid ((y_0 \setminus \gamma_1)\ /\ y_1)\ \rangle \oslash \gamma_0) \mid \alpha_1\ \rangle\ )\ \rangle\\ \ulcorner v \urcorner &= \lambda\widetilde{\alpha}_0.((\mathsf{everybody}\ \widetilde{\alpha}_0)\ \lambda\widetilde{\gamma}_0.(\lambda\widetilde{y}_0.(\mathsf{someone}\\ &\qquad \lambda\widetilde{\alpha}_1.(\lambda\widetilde{y}_1.(\widetilde{\alpha}_1\ \lambda h.((\mathsf{teases}\ \lambda u.((u\ (h\ \widetilde{\gamma}_0))\ \widetilde{y}_0))\ \widetilde{y}_1))))))\\ \varepsilon\left(\widehat{\theta}_{13}\ulcorner v \urcorner\right) &= (\forall\ \lambda x.(\exists\ \lambda y.((\textsc{teases}\ y)\ x)))\\ \varepsilon\left(\widehat{\theta}_{12}\ulcorner v \urcorner\right) &= \lambda u_9.(\mathit{everybody}\ (\mathit{teases}\ (\mathit{someone}\ u_9)))\end{aligned} \tag{23}$$

$$
\begin{aligned}
v &= \mu\alpha_0.\langle\ \textsf{someone} \mid \widetilde{\lambda}(\beta_0, z_0).\langle\ (\mu\alpha_1.\langle\ \textsf{everybody} \mid \\
&\qquad (\alpha_1 \,/\, \lambda(y_1, \gamma_1).\langle\ \textsf{teases} \mid ((y_1 \setminus \gamma_1) \,/\, z_0)\ )\ \rangle \oslash \alpha_0)\ \mid \beta_0\ \rangle\ \rangle \\
\ulcorner v \urcorner &= \lambda\widetilde{\alpha}_0.(\textsf{someone}\ \lambda\widetilde{\beta}_0.(\lambda\widetilde{z}_0.(\widetilde{\beta}_0\ \lambda h.((\textsf{everybody} \\
&\qquad (h\ \widetilde{\alpha}_0))\ \lambda\widetilde{\gamma}_1.(\lambda\widetilde{y}_1.((\textsf{teases}\ \lambda u.((u\ \widetilde{\gamma}_1)\ \widetilde{y}_1))\ \widetilde{z}_0)))))) \\
\varepsilon(\widehat{\theta}_{13}\ulcorner v \urcorner) &= (\exists\ \lambda y.(\forall\ \lambda x.((\textsc{teases}\ y)\ x))) \\
\varepsilon(\widehat{\theta}_{12}\ulcorner v \urcorner) &= \lambda u_9.(\textit{everybody}\ (\textit{teases}\ (\textit{someone}\ u_9))) \qquad (24)
\end{aligned}
$$

The use of continuations in natural language *semantics* was advocated originally in [18] and [2]. One could say that the symmetric **LG** calculus pulls continuations back into the *syntactic* source calculus; their semantic effects now arise through the CPS translation. Barker and Shan, in a series of papers, [3–5, 45] among others, have worked out continuation-based analyses for a variety of semantic phenomena. An explicit comparison with **LG** is worked out in [6], where it is shown that the Barker-Shan analyses can be *simulated* in the symmetric calculus, provided one assumes an associative regime for *context* composition, while phrasal composition maintains the **NL** non-associative regime. Of course, simulation here is not an end in itself, but a tool to bring out commonalities and differences between the approaches.

Parsing as deduction Apart from the λ**LG** terms of Eqs. 23 and 24, which were chosen because they correspond to different semantic readings, there are many more derivations in the source logic: sequent calculus, as is well known, identifies far fewer proofs than natural deduction. This many-to-one relation between sequent versus natural deduction derivations is often taken to be a flaw of the sequent calculus ('spurious ambiguity', 'bureaucracy of syntax'). In [15], the flexibility of the $\overline{\lambda}\mu\widetilde{\mu}$ calculus and its underlying sequent proof format in choosing the order of activation of formulas is exploited as a feature, rather than being perceived as a bug. Coen presents a 'rendering semantics' for $\overline{\lambda}\mu\widetilde{\mu}$ calculus, which compositionally translates a $\overline{\lambda}\mu\widetilde{\mu}$ term into a natural language *text*, a step-by-step explanation of the mathematical proof encoded by the term. The duality between call-by-value and call-by-name, from this perspective, reflects bottom-up versus top-down proof styles.

Recasting this approach in terms of *parsing* strategies, one sees that call-by-value steps are bottom-up in processing the arguments before processing the function; top-down call-by-name steps process the function up to the point where the need arises to process the arguments. We illustrate with two derivations for the sentence 'Molly teases the girl', the first one is a pure bottom-up derivation, the second one proceeds top-down. At the level of the

CPS translation, these two derivations are identified: the term obtained by $\lceil\cdot\rceil$ is isomorphic to a *natural deduction* derivation in **LP** (MILL).

$$np \cdot \otimes \cdot ((np\backslash s)/np \cdot \otimes \cdot (np/n \cdot \otimes \cdot n)) \vdash s$$

$$\begin{aligned} v_{bottom\text{-}up} &= \mu\alpha_0.\langle\ \mathsf{girl} \mid \widetilde{\mu} y_0.\langle\ \mathsf{the} \\ &\qquad \mid (\widetilde{\mu} z_0.\langle\ \mathsf{teases} \mid ((\mu\alpha_1.\langle\ \mathsf{molly} \mid \alpha_1\ \rangle \setminus \alpha_0) \,/\, z_0)\ \rangle \,/\, y_0)\ \rangle\ \rangle \\ v_{top\text{-}down} &= \mu\alpha_0.\langle\ \mathsf{teases} \mid ((\mu\beta_0.\langle\ \mathsf{molly} \mid \beta_0\ \rangle \setminus \alpha_0) \\ &\qquad / \mu\gamma_0.\langle\ \mathsf{the} \mid (\gamma_0 \,/\, \mu\alpha_1.\langle\ \mathsf{girl} \mid \alpha_1\ \rangle)\ \rangle)\ \rangle \end{aligned}$$

$$\lceil v_{bottom\text{-}up}\rceil = \lceil v_{top\text{-}down}\rceil = \lambda\widetilde{\alpha}_0.((\llbracket\mathsf{the}\rrbracket\ (\llbracket\mathsf{teases}\rrbracket\ \lambda k.((k\ \widetilde{\alpha}_0)\ \llbracket\mathsf{molly}\rrbracket)))\ \llbracket\mathsf{girl}\rrbracket)$$

The procedural view on derivations taken here allows for a logical reconstruction of parsing algorithms for **LG** to be compared with the deductive parsing approach for context-free rewriting systems and their extensions in [46]. Proof nets provide a declarative alternative. Moot [38] shows that the nets for **NL** of [40] can be straightforwardly generalized to the symmetric calculus **LG**; he proves soundness and completeness of these nets with respect to a Display Logic presentation of **LG** *without* the focusing rules. These nets, then, identify **LG** derivations producing the same axiom linking but maybe differing in the order of application of the inference steps that lead to this axiom linking.

6 Generative Capacity, Complexity

The previous section shows that the distributivity laws of **LG** extend the semantic expressivity of the logic without sacrificing its sensitivity for grammatical word order and phrase structure. In the present section we turn to expressivity considerations of a more syntactic nature, showing also here that **LP** expressivity is compatible with the non-associative, non-commutative resource management regime of **LG**.

6.1 Type Similarity

The similarity relation $\sim$, introduced in [30], is the reflexive, transitive, symmetric closure of the derivability relation: $A \sim B$ iff there exists a sequence $C_1 \ldots C_n$ $(1 \leq n)$ such that $C_1 = A$, $C_n = B$ and $C_i \vdash C_{i+1}$ or $C_{i+1} \vdash C_i$ $(1 \leq i < n)$. Lambek proves that $A \sim B$ if and only if one of the following equivalent statements hold (the so-called diamond property): (i) $\exists C$ such that $A \vdash C$ and $B \vdash C$; (ii) $\exists D$ such that $D \vdash A$ and $D \vdash B$. In other words, given a join type C for A and B, one can compute a meet type D, and vice versa. The similarity relation $\sim$ captures the fact that types are derivational relatives in terms of the *multiplicative* composition operations. In contrast, *additive* meet

or join operations do not imply a derivational relation; they would allow, for example, combinations $p \sqcap q$, $p \sqcup q$ of distinct atomic types.

The similarity relation for various typelogical systems has been characterized in terms of an algebraic interpretation of the types $[\![\cdot]\!]$, in the sense that $A \sim B$ iff $[\![A]\!] =_{\mathcal{S}} [\![B]\!]$ in the relevant algebraic structures $\mathcal{S}$. Whether $[\![A]\!] =_{\mathcal{S}} [\![B]\!]$ holds can in general be easily checked, making $\sim$ a powerful probe to detect the existence of a derivational relationship between types without actually performing a derivation.

Table 1 gives an overview of the results. For the pure residuation logic **NL**, $\mathcal{S}$ is the free quasigroup generated by the atomic types. A quasigroup is a set equipped with operations $/, \cdot, \backslash$ satisfying the equations $(x/y) \cdot y = x$, $y \cdot (y\backslash x) = x$, $(x \cdot y)/y = x$, $y\backslash(y \cdot x) = x$. The interpretation $[\![\cdot]\!]$ is defined in the obvious way: $[\![p]\!] = p$, $[\![A/B]\!] = [\![A]\!]/[\![B]\!]$, $[\![B\backslash A]\!] = [\![B]\!]\backslash[\![A]\!]$, $[\![A \otimes B]\!] = [\![A]\!] \cdot [\![B]\!]$.

In associative **L**, type similarity coincides with equality in the free group generated by the atomic types (free Abelian group for associative/commutative **LP**). The group interpretation has $[\![p]\!] = p$, $[\![A \otimes B]\!] = [\![A]\!] \cdot [\![B]\!]$, $[\![A/B]\!] = [\![A]\!] \cdot [\![B]\!]^{-1}$, $[\![B\backslash A]\!] = [\![B]\!]^{-1} \cdot [\![A]\!]$. We see then that for **NL**, **L**, **LP** expressivity for $\sim$ is inversely proportional to structural discrimination. In the case of symmetric **LG**, one achieves the same level of expressivity with respect to $\sim$ as the associative/commutative calculus **LP** *without* inducing loss of structural discrimination. The group interpretation for **LG** is a variant of that for **L(P)**: in addition to the atomic formulae, one adds an extra generator to keep track of the *connective count*. The connective count discriminates between, for example, $np\backslash s$ and $s \oslash np$—formulas agreeing on the atom count, but not in the similarity relation, because the connective count is not balanced.

The **LG** solutions for the diamond property in Eq. 25 are related by arrow-reversal duality: $(A/-) \otimes (- \oslash (B \osl -)) \overset{\infty}{\longleftrightarrow} ((-/B)\backslash -) \oplus (- \osl A)$. They have the same complexity as the solutions for **L**, but rely on the distributivity laws, rather than on the associativity of $\otimes$. The same applies for the constructive recipes given in [37] for neutral elements (modifiers) $A\backslash A \sim B/B$ with join $(A\backslash((A \otimes B) \oslash B)) \oplus (B/B)$; associative relatives $(A\backslash B)/C \sim A\backslash(B/C)$ with join $(A\backslash B) \oplus ((B \osl B)/C)$, and commutative relatives $A/B \sim B\backslash A$, with join/meet types combining the solutions for neutral and associative relatives.

$$\begin{array}{rll} & \text{meet } D \text{ given join } C: & \text{join } C \text{ given meet } D: \\ \mathbf{L}: & D = (A/C) \otimes (C \otimes (C\backslash B)) & C = (D/A)\backslash(D/(B\backslash D)) \\ \mathbf{LG}: & D = (A/C) \otimes (C \oslash (B \osl C)) & C = ((D/B)\backslash D) \oplus (D \osl A) \end{array} \tag{25}$$

Table 1 Models for type equivalence

Calculus	Interpretation
NL	Free quasigroup [23]
L	Free group [43]
LP	Free Abelian group [43]
LG	Free Abelian group [37]

Chameleon types: extraction The fact that types in the similarity relation have join and meet types for $\sim$ opens up the possibility for new grammatical analyses. Assigning such a meet type for $A \sim B$ to a word in the lexicon allows it to *derivationally* adapt to its environment, and behave either as an A or as a B as the need arises. Bastenhof [6] uses this strategy in an analysis of *extraction* discontinuities such as illustrated in Eq. 1 (b), (c) at the beginnning of this article. Predicates that allow the extraction of an argument are systematically associated with a secondary type assignment for their occurrence in extraction environments. The 'regular' type, for environments where the arguments are physically realized, and the secondary type for 'trace' environments can be shown to have a meet type. The type for extraction environments is an instance of the schema $(B \oslash C) \odot A$ that was used above to obtain the scope ambiguities of Eqs. 23 and 24: semantic and syntactic discontinuities are given a uniform treatment then.

6.2 Recognizing Capacity, Complexity

The discussion of type similarity shows that **LG** achieves the expressivity of **LP** in a structure-preserving way. With respect to recognizing capacity a similar situation obtains.

First, it is shown in [38] that the mildly context-sensitive languages recognized by lexicalized tree adjoining grammars are also recognized by **LG**. The result is established on the basis of a streamlined version of LTAG, for which a compositional translation into **LG** is given. The key idea of the translation is that adjunction points of the LTAG grammar are mapped to **LG** formulas of the schematic form $(T \oslash A) \odot \widehat{T}$. As we saw in Section 2, the connectives $\oslash, \odot$ form a Galois connected pair, which means that $(T \oslash A) \odot \widehat{T}$ *contracts* to A provided $\widehat{T} \vdash T$ holds. Such formulas then can behave as ordinary A formulas in case no adjunction takes place. Moot works out the construction for the canonical examples of mildly context-sensitive patterns: $\{ww \mid w \in \{\mathsf{a}, \mathsf{b}\}^+\}$ (the copy language), $\{\mathsf{a}^n\mathsf{b}^n\mathsf{c}^n \mid n > 0\}$ (counting dependencies), and $\{\mathsf{a}^n\mathsf{b}^m\mathsf{c}^n\mathsf{d}^m \mid n > 0\}$ (crossed dependencies). We reproduce the case of counting dependencies in Fig 4. In the LTAG source grammar, the vocabulary items a and c are introduced by nonterminals A and C respectively. For b, there is an initial tree rooted in S and an auxiliary tree with adjunction node (T) and foot node T. The translation maps the foot node to an **LG** formula $\widehat{t}$, built from t and an atomic formula i not occurring elsewhere in the grammar, such that $\widehat{t} \vdash t$. Possible solutions for $\widehat{t}$ are $(t/i) \otimes i$ or $i \oslash (t \odot i)$. This translation uses the fact that $t \nvdash \widehat{t}$ to enforce a null adjunction constraint for foot nodes. The adjunction node (T) is mapped to $(t \oslash A) \odot \widehat{t}$.

The recognizing capacity of **LP** goes beyond LTAG; [49] shows that all permutation closures of context-free languages are recognizable in **LP**. Among these is Bach's MIX language $\{w \mid w \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}^+$ and $|a|_w = |b|_w = |c|_w \geq 1\}$, i.e. the language consisting of strings over the alphabet $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ with an equal number of a's, b's and c's occurring in any order. MIX is the permutation closure of context-free languages such as $\mathsf{a}^n(\mathsf{bc})^n$, $(\mathsf{ab})^n\mathsf{c}^n$, or (regular) $(\mathsf{abc})^n$,

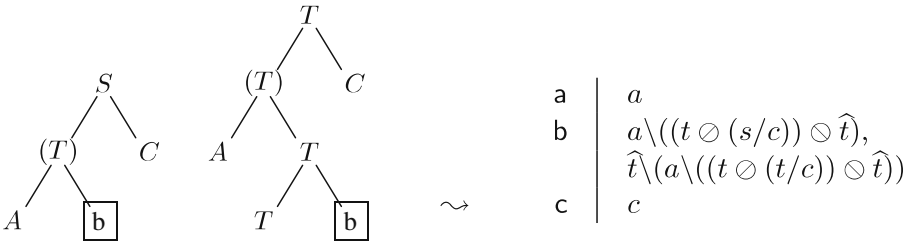


Fig. 4 Counting dependencies $\{\mathsf{a}^n\mathsf{b}^n\mathsf{c}^n\},\ n > 0$. LTAG (*left*), **LG** translation (*right*)

hence **LP** recognizable. Using the count invariant, it is straightforward to show that the **LP** lexicon below (for start symbol s) will do the job.

$$\begin{array}{lll} \textbf{LP} \text{ lexicon:} & \mathsf{a} & a \\ & \mathsf{b} & a \to b \\ & \mathsf{c} & b \to s,\ b \to s \to s \end{array} \tag{26}$$

To recognize MIX with this lexicon, **LP** relies on the associativity and commutativity of its resource management regime: thanks to these structural options, the types can be regrouped into a configuration corresponding to the regular pattern $(\mathsf{abc})^n$. In **LG** a similar effect can be achieved by means of the Grishin distributivity principles. The lexicon in Eq. 27 realizes an **LG** grammar for MIX, again for start symbol s. The construction uses type assignments with co-implication as the main connective: the input parts of these types are composed on the antecedent side of the turnstile; the output parts move to the succedent side by means of the distributivity and dual residuation laws, and compose there. The input part of the type assignments is either s/s or s. The latter is for the occurrence of the alphabet symbol as the last element of the string generated, the former for the non-final occurrences. The output part is such that s expands to a coproduct structure with as its yield the context-free pattern $a^n\ s\ (b'\ c)^n$, where we write b' for the output part of the types assigned to b: $s \oslash (a \oslash (s \oslash c))$.

$$\begin{array}{lll} \textbf{LG} \text{ lexicon:} & \mathsf{a} & a \oslash s,\ a \oslash (s/s) \\ & \mathsf{b} & s \oslash (s \oslash (a \oslash (s \oslash c))),\ (s/s) \oslash (s \oslash (a \oslash (s \oslash c))) \\ & \mathsf{c} & s \oslash c,\ (s/s) \oslash c \end{array} \tag{27}$$

Recognition of a string of length $3n$ $(1 \leq n)$ is modeled by a derivation for a sequent $X \vdash s$ with X a right-branching product structure with a yield of length $3n$. A derivation can proceed in two stages. First, repeat the following steps n times: (i) shift the focus from the goal formula s to an input b-type and apply the $(\oslash L)$ rule to it; (ii) shift the focus to an input a-type and apply the $(\oslash L)$ to it; (iii) shift the focus to an input c-type and apply the $(\oslash L)$ to it; (iv) shift the focus to the output $s \oslash (a \oslash (s \oslash c))$ formula and apply the $(\oslash R)$ (twice) and $(\oslash R)$ rules to it. For each iteration, b' consumes an a and a c producing a new goal s. After n repetitions, if the alphabet elements in the input are indeed

of equal multiplicity, the open premise is of the form $X' \vdash s$, where X' is a right-branching product structure with yield

$$\underbrace{(s/s) \cdots (s/s)}_{3n-1}\ s$$

This structure contracts to s by activating an antecedent s/s formula $3n-1$ times and applying the $(/L)$ rule to it. The key point behind this construction is that, *irrespective* of the surface order of the types for the alphabet symbols making up the antecedent structure, the Grishin distributivities make it possible to liberate the a and c formulas in the order required for consumption by $s \oslash (a \oslash (s \oslash c))$.

MIX is a simple example of a pattern not recognized by LTAG. Melissen [35] generalizes the type of construction exemplified in Eq. 27 and obtains the result below.

Theorem [35] *All languages which are the intersection of a context-free language and the permutation closure of a context-free language are recognizable in* **LG**.

In this class, we find generalized forms of MIX, with equal multiplicity of k alphabet symbols in any order, and counting dependencies $a_1^n \ldots a_k^n$ for any number k of alphabet symbols. Patterns of this type are recognized by Range Concatenation Grammars [10] and Global Index Grammars [13]; a comparison with these formalisms then might be useful to fix the upper bound of the recognizing capacity of **LG**, which is as yet unknown.

Also with respect to computational complexity, there are some intriguing open questions. For the class of **LG** grammars in correspondence with LTAG, [39] obtains a polynomial parsability result via a translation into Hyperedge Replacement Grammars of rank 2. As we saw above, this result assumes a restriction on the formulas of a **LG** grammar, where the Galois connected $\oslash, \oslash$ operations occur in matching pairs. The lexicon of the MIX construction (27), and the type assignment used for quantifier phrases in the analysis of scope construal in [7], do not respect this restriction, which means the complexity of these **LG** grammars is open. Note that both Range Concatenation Grammars and GIG have polynomial complexity.

7 Conclusion

A turning point in the reappraisal of Lambek's 1958 and 1961 papers in the early 1980s was the introduction of a Curry-Howard derivational semantics in [47]. At the same time, this work made the conflicts of interests between syntactic structure-sensitivity and semantic flexibility acutely felt. It is ironic that Grishin's 1983 paper also had to wait twenty years before the potential of his ideas for resolving these conflicts started to be exploited.

References

1. Abrusci, V. (1991). Phase semantics and sequent calculus for pure noncommutative classical linear propositional logic. *Journal of Symbolic Logic, 56*, 1403–1451.
2. Barker, C. (2002). Continuations and the nature of quantification. *Natural Language Semantics, 10*, 211–242.
3. Barker, C. (2004). Continuations in natural language. In H. Thielecke (Ed.), *CW'04: Proceedings of the 4th ACM SIGPLAN continuations workshop* (pp. 1–11). Tech. Rep. CSR-04-1, School of Computer Science, University of Birmingham.
4. Barker, C., & Shan, C. (2006). Types as graphs: Continuations in type logical grammar. *Journal of Logic, Language and Information, 15*(4), 331–370.
5. Barker, C., & Shan, C. (2008). Donkey anaphora is in-scope binding. *Semantics and Pragmatics, 1*(1), 1–46.
6. Bastenhof, A. (2009). *Continuations in natural language syntax and semantics*. Utrecht University: MPhil Linguistics. Available at http://www.igitur.nl/studenttheses/index.php.
7. Bernardi, R., & Moortgat, M. (2009). Continuation semantics for the Lambek-Grishin calculus. *Information and Computation* (to appear).
8. Bimbó, K., & Dunn, J. M. (2008). *Generalized Galois logics. Relational semantics of nonclassical logical calculi*. Stanford: CSLI.
9. Bimbó, K., & Dunn, J. M. (2009). Symmetric generalized Galois logics. *Logica Universalis, 3*(1), 125–152.
10. Boullier, P. (1999). Chinese numbers, MIX, scrambling, and range concatenation grammars. In *Proceedings of the 9th EACL conference* (pp. 53–60). Morristown: Association for Computational Linguistics.
11. Buszkowski, W. (1997). Mathematical linguistics and proof theory. In J. van Benthem & A. ter Meulen (Eds.), *Handbook of logic and language* (Chapter 12, pp. 683–736). Amsterdam: Elsevier/MIT.
12. Casadio, C. (2001). Non-commutative linear logic in linguistics. *Grammars, 4*(3), 167–185.
13. Castaño, J. (2004). Global index grammars and descriptive power. *Journal of Logic, Language and Information, 13*(4), 403–419.
14. Cockett, C., & Seely, R. A. G. (1996). Proof theory for full intuitionistic linear logic, bilinear logic and mix categories. *Theory and Applications of Categories, 3*, 85–131.
15. Coen, C. S. (2006). Explanation in natural language of lambda-mu-comu terms. In *Mathematical knowledge management, 4th international conference, MKM 2005, Bremen, Germany, 15–17 July 2005, revised selected papers. Lecture notes in computer science* (Vol. 3863, pp. 234–249). New York: Springer.
16. Curien, P., & Herbelin, H. (2000). Duality of computation. In *International conference on functional programming (ICFP'00)* (pp. 233–243) (2005: corrected version).
17. de Groote, P. (2001). Towards abstract categorial grammars. In *Proceedings of 39th annual meeting of the association for computational linguistics*, Toulouse, France (pp. 252–259). Morristown: Association for Computational Linguistics.
18. de Groote, P. (2001). Type raising, continuations, and classical logic. In M. S. R. van Rooy (Ed.), *Proceedings of the thirteenth Amsterdam colloquium* (pp. 97–101). Universiteit van Amsterdam: ILLC.
19. de Groote, P., & Lamarche, F. (2002). Classical non-associative Lambek calculus. *Studia Logica, 71*(3), 355–388.
20. de Groote, P., & Pogodalla, S. (2004). On the expressive power of abstract categorial grammars: Representing context-free formalisms. *Journal of Logic, Language and Information, 13*(4), 421–438.
21. de Paiva, V., & Ritter, E. (2006). A Parigot-style linear λ-calculus for full intuitionistic linear logic. *Theory and Applications of Categories, 17*(3), 30–48.
22. Dunn, J., & Hardegree, G. (2001). *Algebraic methods in philosophical logic*. Oxford: Oxford University Press.
23. Foret, A. (2003). On the computation of joins for non-associative Lambek categorial grammars. In J. Levy, M. Kohlhase, J. Niehren, & M. Villaret (Eds.), *Proceedings of the 17th international workshop on unification, UNIF'03 Valencia* (pp. 25–38).

24. Galatos, N., Jipsen, P., Kowalski, T., & Ono, H. (2007). *Residuated lattices: An algebraic glimpse at substructural logics. Studies in logic and the foundations of mathematics* (Vol. 151). Amsterdam: Elsevier.
25. Gehrke, M. (2006). Generalized Kripke frames. *Studia Logica, 84*(2), 241–275.
26. Goré, R. (1997). Substructural logics on display. *Logic Journal of IGPL, 6*(3), 451–504.
27. Grishin, V. (1983). On a generalization of the Ajdukiewicz-Lambek system. In A. Mikhailov (Ed.), *Studies in nonclassical logics and formal systems, Moscow* (pp. 315–334). Nauka (English translation in Abrusci & Casadio (Eds.), New perspectives in logic and formal linguistics. Bulzoni, Rome, 2002).
28. Herbelin, H. (2005). *C'est maintenant qu'on calcule: Au cœur de la dualité*. Université Paris XI: Habilitation à diriger les recherches.
29. Kurtonina, N., & Moortgat, M. (2007). Relational semantics for the Lambek-Grishin calculus. In *Proceedings MOL10, 10th conference on mathematics of language, Los Angeles*. Available at http://molweb.org/mol10/.
30. Lambek, J. (1958). The mathematics of sentence structure. *American Mathematical Monthly, 65*, 154–170.
31. Lambek, J. (1961). On the calculus of syntactic types. In R. Jakobson (Ed.), *Structure of language and its mathematical aspects* (pp. 166–178). Providence: American Mathematical Society.
32. Lambek, J. (1993). From categorial to bilinear logic. In K. Došen & P. Schröder-Heister (Eds.), *Substructural logics* (pp. 207–237). Oxford: Oxford University Press.
33. Lambek, J. (2008). *From word to sentence. A computational algebraic approach to grammar*. Milano: Polimetrica.
34. Lengrand, S. (2003). Call-by-value, call-by-name, and strong normalization for the classical sequent calculus. In B. Gramlich & S. Lucas (Eds.), *Post-proceedings of the 3rd international workshop on reduction strategies in rewriting and programming (WRS'03). Electronic notes in theoretical computer science* (Vol. 86). Amsterdam: Elsevier.
35. Melissen, M. (2009). The generative capacity of the Lambek-Grishin calculus: A new lower bound. In P. de Groote (Ed.), *Proceedings 14th conference on formal grammar. Lecture notes in computer science* (Vol. 5591). New York: Springer.
36. Moortgat, M. (1996). Multimodal linguistic inference. *Journal of Logic, Language and Information, 5*(3), 349–385.
37. Moortgat, M., & Pentus, M. (2007). Type similarity for the Lambek-Grishin calculus. In *Proceedings 12th formal grammar conference*. Dublin.
38. Moot, R. (2007). Proof nets for display logic. *CoRR, abs/0711.2444*.
39. Moot, R. (2008). Lambek grammars, tree adjoining grammars and hyperedge replacement grammars. In *Proceedings of TAG+9, the 9th international workshop on tree adjoining grammars and related formalisms, Tübingen* (pp. 65–72).
40. Moot, R., & Puite, Q. (2002). Proof nets for the multimodal Lambek calculus. *Studia Logica, 71*(3), 415–442.
41. Morrill, G., Leslie, N., Hepple, M., & Barry, G. (1990). Categorial deductions and structural operations. *Studies in Categorial Grammar, Edinburgh Working Papers in Cognitive Science, 5*, 1–21.
42. Morrill, G., Valentin, O., & Fadda, M. (2009). Dutch grammar and processing: A case study in TLG. In P. Bosch, D. Gabelaia, & J. Lang (Eds.), *Logic, language, and computation: 7th international Tbilisi symposium on logic, language and computation, Tbilisi, Georgia, 1–5 October 2007. Revised selected papers, LNAI 5422* (pp. 272–286). New York: Springer.
43. Pentus, M. (1993). The conjoinability relation in Lambek calculus and linear logic. ILLC Prepublication Series ML–93–03, Institute for Logic, Language and Computation, University of Amsterdam.
44. Retoré, C., & Salvati, S. (2007). Non-associative categorial grammars and abstract categorial grammars. In R. Muskens (Ed.), *New directions in type theoretic grammars, Dublin* (pp. 51–58).
45. Shan, C. (2005). Linguistic side effects. Ph.D. thesis, Harvard University.
46. Shieber, S., Schabes, Y., & Pereira, F. (1995). Principles and implementation of deductive parsing. *The Journal of Logic Programming, 24*(1–2), 3–36.
47. van Benthem, J. (1983). The semantics of variety in categorial grammar. Technical Report 83-29, Simon Fraser University, Burnaby (B.C.) (Revised version in W. Buszkowski,

W. Marciszewski, & J. van Benthem (Eds.), Categorial grammar, Benjamin, Amsterdam, 1988).
48. van Benthem, J. (1986). *Essays in logical semantics. Studies in linguistics and philosophy* (Vol. 29). Dordrecht: Reidel.
49. van Benthem, J. (1995). *Language in action: Categories, lambdas and dynamic logic*. Cambridge: MIT.