Proofs nets and the categorial flow of information

Michael Moortgat, Richard Moot

Proofs nets and the categorial flow of information

Michael Moortgat and Richard Moot

Utrecht Institute of Linguistics OTS, LABRI-CNRS Bordeaux
M.J.Moortgat@uu.nl,moot@labri.fr

Abstract

The Lambek-Grishin calculus (**LG**) is a multiple-conclusion extension of Lambek's categorial type logic with dual families of fusion ('merge') and fission operations, and linear distributivity principles relating these two. Thanks to the distributivity principles, **LG** captures dependency patterns beyond context-free, both in syntax and semantics. In this paper we represent the information flow in categorial derivations in terms of a proof net graphical calculus. We study the correspondence between the composition graphs for these nets and the terms associated with focused sequent derivations.

1 Background, motivation

In this paper, we study **LG**, a type logic based on the generalization of Lambek's Syntactic Calculus proposed in Grishin (1983). The formula language of this

logic is given in (1).

$$\begin{array}{lll} A, B ::= & p \mid & \text{atoms: } s, np, \ldots \\ & A \otimes B \mid B\backslash A \mid A/B \mid & \text{product, left vs right division} \\ & A \oplus B \mid A \oslash B \mid B \ominus A & \text{coproduct, right vs left difference} \end{array} \tag{1}$$

Algebraically, **LG** combines the residuated triple of (3) — fusion with its two residuals — with the dual residuated triple in (4): fission, left and right difference.

$$A \leq A \quad ; \quad \text{from } A \leq B \text{ and } B \leq C \text{ infer } A \leq C \tag{2}$$

$$A \leq C/B \quad \text{iff} \quad A \otimes B \leq C \quad \text{iff} \quad B \leq A\backslash C \tag{3}$$

$$B \ominus C \leq A \quad \text{iff} \quad C \leq B \oplus A \quad \text{iff} \quad C \oslash A \leq B \tag{4}$$

For the interaction between the fusion and fission families, we have the postulates of (5).[1] These postulates have come to be called *linear distributivity principles* (e.g. Cockett and Seely (1996)): linear, because they respect resources (no material gets copied).

$$\begin{array}{ll} (A \ominus B) \otimes C \leq A \ominus (B \otimes C) & C \otimes (B \oslash A) \leq (C \otimes B) \oslash A \\ C \otimes (A \ominus B) \leq A \ominus (C \otimes B) & (B \oslash A) \otimes C \leq (B \otimes C) \oslash A \end{array} \tag{5}$$

LG is attractive for syntactic and for semantic reasons. Syntactically, the interaction principles of (5) bring expressivity beyond context-free. Moot (2007) gives an **LG** encoding of the adjunction operation of Tree Adjoining Grammar, the most restricted formalism in the mildly context-sensitive hierarchy Kallmeyer (2010); Moortgat (2009) has an **LG** grammar for MIX, according to Salvati (p.c.) an instance of a non-wellnested 2-MCFG. The upper bound for the syntactic expressivity of **LG** grammars in their full generality is open; see Melissen (2010) for discussion.

At the semantic level, **LG** derivations can be given an interpretation in the continuation-passing style. The CPS interpretation leads to a considerable simplification of the syntax/semantics interface: semantic scope construal can

[1] There is a second set, with the inequalities reversed, which we'll not discuss here.

be obtained on the basis of simple first-order syntactic types, as shown in Bastenhof (2012) and discussed in §3.1.

Despite these attractions, working with the standard 'symbolic' presentations of **LG** involves rather formidable technical machinery. Our focus in this paper is on *proof nets* — an elegant graphical calculus that captures the essence of **LG** derivations without the bureaucracy of heavy symbol manipulation.

2 Display sequent calculus and proof nets

A sequent calculus for **LG**, in the 'display logic' style, can be found in Goré (1997). We present it in §2.1, using the notation of Moortgat (2009). In §2.2, we introduce the proof net graphical calculus, and show how it leads to a representation of **LG** derivations that is free of spurious ambiguities.

2.1 sLG: display sequent calculus

The characteristic feature of Display Logic is that for every logical connective, there is a corresponding structural connective, not just for conjunction and disjunction as in standard sequent calculus. We use the same symbols for the logical operations and their structural counterparts; structural operations are marked off by centerdots. Below the grammar for input (sequent left hand side), and output structures (sequent rhs). Atomic structures are formulas $\mathcal{F}$.

$$\begin{array}{lcl} \mathcal{I} & ::= & \mathcal{F} \mid \mathcal{I} \cdot \otimes \cdot \mathcal{I} \mid \mathcal{I} \cdot \oslash \cdot \mathcal{O} \mid \mathcal{O} \cdot \oslash \cdot \mathcal{I} \\ \mathcal{O} & ::= & \mathcal{F} \mid \mathcal{O} \cdot \oplus \cdot \mathcal{O} \mid \mathcal{I} \cdot \backslash \cdot \mathcal{O} \mid \mathcal{O} \cdot / \cdot \mathcal{I} \end{array} \tag{6}$$

Figures 1 and 2 then give the structural and logical rules of **sLG**. The (dual) residuation principles take the form of 'display postulates', so called because they allow any formula component of a structure to be displayed as the sole occupant of the sequent lhs or rhs. The logical rules apply to formulas thus displayed. The one-premise rules simply replace a logical connective by its structural counterpart; these rules are invertible. The non-invertible two-premise rules give expression to the monotonicity properties of the type-forming operations. The distributivity principles (5) appear in rule form here as $G1 - G4$ in the structural group.

$$\frac{}{A \Rightarrow A}\ \text{Ax} \qquad \frac{X \Rightarrow A \quad A \Rightarrow Y}{X \Rightarrow Y}\ \text{Cut}$$

$$\frac{X \Rightarrow Z \cdot / \cdot Y}{X \cdot \otimes \cdot Y \Rightarrow Z}\ rp \qquad \frac{Y \cdot \oslash \cdot Z \Rightarrow X}{Z \Rightarrow Y \cdot \oplus \cdot X}\ drp$$

$$\frac{X \cdot \otimes \cdot Y \Rightarrow Z}{Y \Rightarrow X \cdot \backslash \cdot Z}\ rp \qquad \frac{Y \cdot \oslash \cdot Z \Rightarrow X}{Z \cdot \oslash \cdot X \Rightarrow Y}\ drp$$

$$\frac{X \cdot \otimes \cdot Y \Rightarrow Z \cdot \oplus \cdot W}{Z \cdot \oslash \cdot X \Rightarrow W \cdot / \cdot Y}\ G1 \qquad \frac{X \cdot \otimes \cdot Y \Rightarrow Z \cdot \oplus \cdot W}{Y \cdot \oslash \cdot W \Rightarrow X \cdot \backslash \cdot Z}\ G3$$

$$\frac{X \cdot \otimes \cdot Y \Rightarrow Z \cdot \oplus \cdot W}{Z \cdot \oslash \cdot Y \Rightarrow X \cdot \backslash \cdot W}\ G2 \qquad \frac{X \cdot \otimes \cdot Y \Rightarrow Z \cdot \oplus \cdot W}{X \cdot \oslash \cdot W \Rightarrow Z \cdot / \cdot Y}\ G4$$

Figure 1: **sLG**. Structural rules.

$$\frac{A \cdot \$ \cdot B \Rightarrow Y}{A \,\$\, B \Rightarrow Y}\ \$L \quad \$ \in \{\otimes, \oslash, \oslash\} \qquad \frac{X \Rightarrow A \cdot \# \cdot B}{X \Rightarrow A \,\#\, B}\ \#R \quad \# \in \{\oplus, /, \backslash\}$$

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

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

$$\frac{X \Rightarrow A \quad B \Rightarrow Y}{B / A \Rightarrow Y \cdot / \cdot X}\ /L \qquad \frac{X \Rightarrow A \quad B \Rightarrow Y}{Y \cdot \oslash \cdot X \Rightarrow B \oslash A}\ \oslash R$$

Figure 2: **sLG**. Logical rules.

It is shown in Moortgat (2009) that the display sequent format **sLG** enjoys cut elimination and thus allows for decidable proof search. Still, there is room for improvement:

- spurious ambiguity: as with sequent calculi in general, one and the same matching of occurrences of atomic subformulae in a proof's axiom leaves may be obtained in different ways as a result of irrelevant rule permutations;

- no parsing: backward chaining sequent proof search requires the *structure* of the formulas making up the end sequent to be given in advance; for genuine **LG** *parsing*, one would like this structure to be computed as the outcome of the deduction process;
- display equivalences represent alternative views on one and the same structure: one would like to have a proof format where there is no need for the explicit structural manipulations of the display postulates.

The proof net approach to be discussed below removes these problematic aspects.

2.2 Proof nets

Proof nets are a graphical way of representing proofs, introduced first for linear logic Girard (1987). The proof nets for **LG** we present in this section are a simple extension of the proof nets for the multimodal Lambek calculus of Moot and Puite (2002). A proof structure is a (hyper)graph where the vertices are labeled by formulas and the edges connect these formulas.[2] The hyperedges correspond to the logical rules, linking the active formulas and the main formula of the rule and keeping track of whether one is dealing with a non-invertible two-premise rule or with an invertible one-premise rule. We'll call these *tensor* and *cotensor* links respectively.

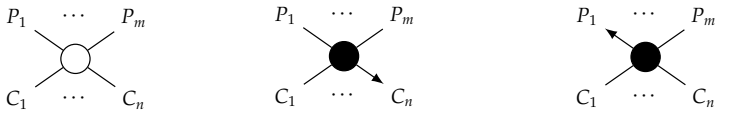
Definition 2.1. A link is a tuple $\langle t, p, c, m\rangle$ where

- t is the type of the link — tensor or cotensor
- p is the list of premises of the link,
- c is the list of conclusions of the link,
- m, the main vertex/formula of the link, is either a member of p, a member of c or the constant "nil".

In case m is a member of p we speak of a *left* link (corresponding to the left rules of the sequent calculus, where the main formula of the link occurs in the antecedent) and in case m is a member of c we speak of a *right* link.

[2] In what follows we will often speak of formula occurrences (or simply *formulas* if there is no possibility of confusion) instead of vertices labeled by formulas.

Graphically, links are displayed as shown below. A central node links together the premises and conclusions of the link; when we need to refer to the connections between the central node and the vertices, we will call them its *tentacles*. The interior of this central node is white for a tensor link and black for a cotensor link. The premises are drawn, in left-to-right order, above the central node and the conclusions, also in left-to-right order, are drawn below it. The main formula of cotensor links is drawn with an arrow towards it; the main formula of a tensor link can only be determined by inspection of the formulas.



tensor rule cotensor rule (right rule) cotensor rule (left rule)

Figure 3 shows the links for **LG**. The links for the fission connectives are up-down symmetric versions of the links for the fusion connectives.

Definition 2.2. A *proof structure* $\langle S, \mathcal{L} \rangle$ is a finite set of formula occurrences S and a set of links $\mathcal{L}$ from those shown in Figure 3 such that

- each formula is at most once the premise of a link,
- each formula is at most once the conclusion of a link.

Formulas which are not the conclusion of any link are called the *hypotheses* of the proof structure. Formulas which are not the premise of any link are called the *conclusions* of the proof structure.

We will say that a proof structure with hypotheses $H_1, \ldots, H_m$ and conclusions $C_1, \ldots, C_n$ is a proof structure of $H_1, \ldots, H_m \Rightarrow C_1, \ldots C_n$.

Example 1. Figure 4 shows the hypothesis unfolding of $(s \oslash s) \oslash np$ and the conclusion unfolding of $s \,/\, (np \setminus s)$. Both are obtained by simple application of the rules of Figure 3 until we reach the atomic subformulas.

Though the figure satisfies the condition on proof structures (connectedness is not a requirement), it is a proof structure of $(s \oslash s) \oslash np, s, s, np \Rightarrow s / (np \setminus s), s, s, np$. We obtain a proof structure of $(s \oslash s) \oslash np \Rightarrow s \,/\, (np \setminus s)$ by identifying atomic formulas.[3] In this case, we choose to identify the top s of the left subgraph

[3] This node identification corresponds to the "axiom links" of linear logic proof nets.

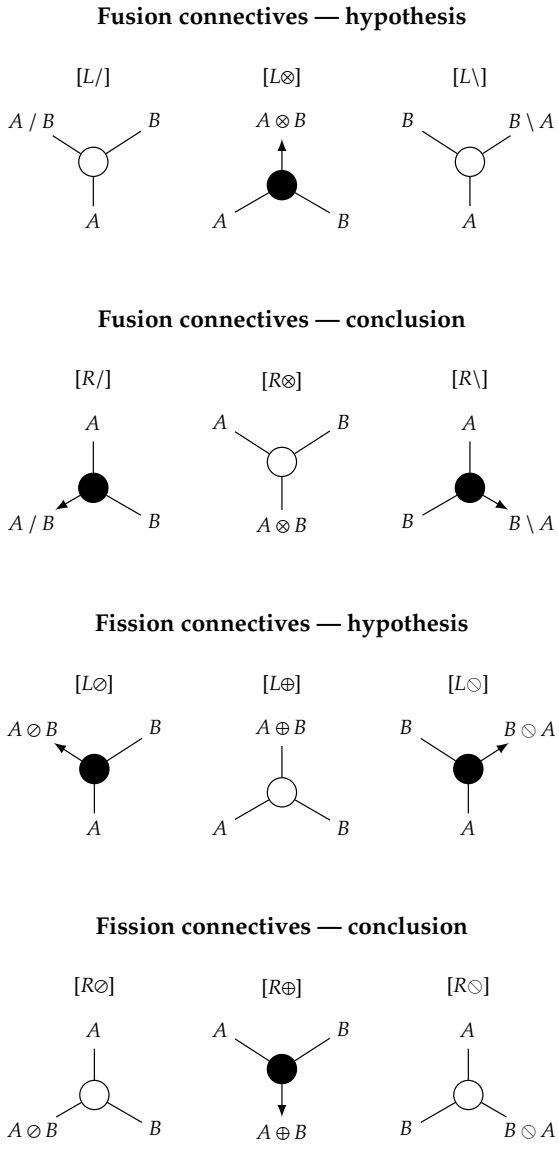


Figure 3: Links for proof structures of the Lambek-Grishin calculus

with the bottom s of the right subgraph and perform the unique choice for the remaining atomic formulas. The result is the proof structure shown in Figure 5

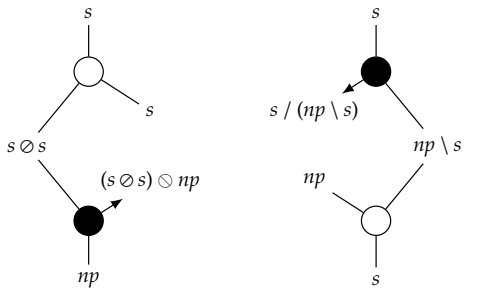


Figure 4: Lexical unfolding

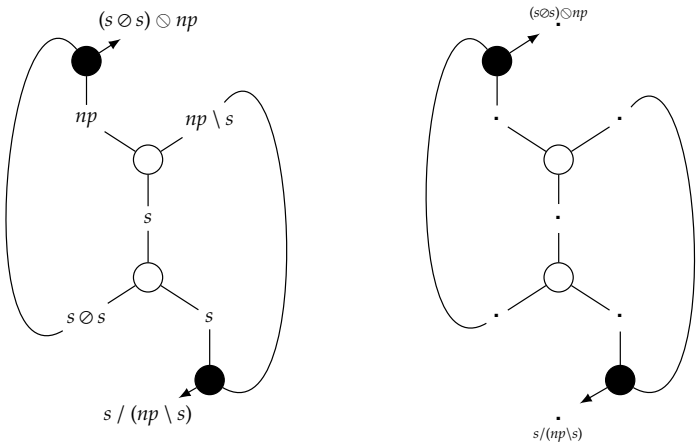


Figure 5: Proof structure of $(s \oslash s) \nslash np \Rightarrow s \,/\, (np \setminus s)$ corresponding to the lexical unfolding of Figure 4 and its corresponding abstract proof structure

on the left.

Due to the graphical constraints of writing these proof nets on the plane — we want to draw the $np \setminus s$ node *below* the cotensor link at the bottom of the figure, since it is a conclusion of this link, but would have to draw the figure on a cylinder to make this work — we need to use curved tentacles connect the minor premise of (co-)implication links to the rest of the proof structure.

Definition 2.3. An *abstract proof structure* $\langle V, \mathcal{L}, h, c \rangle$ is a set of vertices V, a set of (unlabeled) links $\mathcal{L}$ and two functions h and c, such that

- each formula is at most once the premise of a link,
- each formula is at most once the conclusion of a link,
- h and c are functions from the hypotheses resp. conclusions of the abstract proof structure to formulas

Note that the abstract proof structure corresponding to a two formula sequent $A \Rightarrow B$ has only a single vertex v, with $h(v) = A$ and $c(v) = B$.

The transformation from proof structure to abstract proof structure is a forgetful mapping: we transform a proof structure into an abstract proof structure by erasing all formula information on the internal vertices. Visually, we remove the formula labels from the graph and replace them by simple vertices (**.**). We indicate the results of the functions h and c above (resp. below) the vertices (for the hypotheses and conclusions respectively). As a result, we have to following four types of vertices in an abstract proof structure.

internal hypothesis conclusion both

Example 2. Figure 5 shows (on the right) the transformation of the proof structure on its left into an abstract proof structure.

Definition 2.4. A *tree* is an acyclic, connected abstract proof structure which does not contain any cotensor links.

The trees of Definition 2.4 correspond to sequents in a rather direct way. In fact, they have the pleasant property of "compiling away" the display rules of the sequent calculus. Or, in other words, trees represent a class of sequents which is equivalent up to the display postulates.

Definition 2.5. Given an abstract proof structure A, we say that A *contracts in one step* to A', written $A \to A'$ iff A' is obtained from A by replacing one of the subgraphs of the form shown in Figures 6 and 7 by a single vertex.

H represents the result of the function h for the indicated node (relevant only in case this node is a hypothesis of the abstract proof structure). Similarly, C represents the formula assigned by the function c to the indicated node.

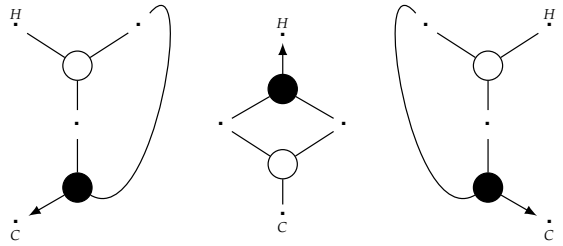


Figure 6: Contractions — Lambek connectives

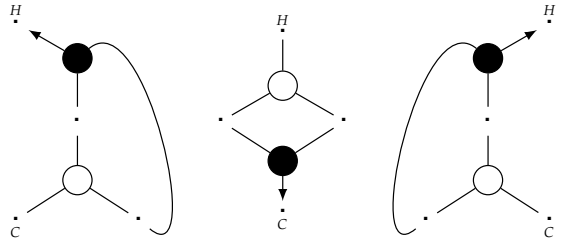


Figure 7: Contractions — Grishin connectives

Given an abstract proof structure A we say that *A contracts to* an abstract proof structure A' if there is a sequence of zero or more one step contractions from A to A'.

When we say that a *proof structure P* contracts to an abstract proof structure A' we will mean that the underlying abstract proof structure A of P contracts to A'.

To obtain expressivity beyond context-free, we are interested in **LG** with added interaction principles. Figures 8 and 9 give the additional rewrite rules on abstract proof structures that correspond to the rule form of Grishin's distributivity laws.

Definition 2.6. A proof structure P is a *proof net* iff its underlying abstract proof structure A converts to a tree using the contractions of Figures 6 and 7 and the

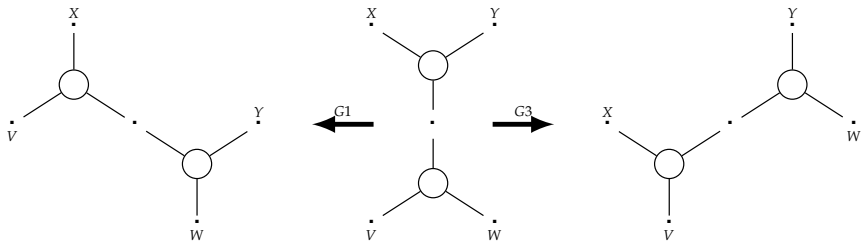


Figure 8: Grishin interactions I — "mixed associativity"

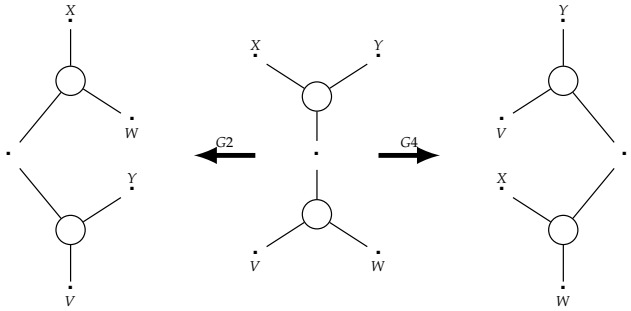


Figure 9: Grishin interactions II — "mixed commutativity"

structural rules of Figures 8 and 9.

Theorem 1. *A proof structure P is a proof net — that is, P converts to a tree T — iff there is a sequent proof of T.*

The proof is an easy adaptation of the proof of Moot and Puite (2002). A detailed proof can be found in Moot (2007).

Example 3. We show that the proof structure of Figure 5 is a proof net by contracting it to a tree. Starting with rule ($G1$), the two cotensor links can be contracted in any order. Figure 10 shows a complete sequence.

Example 4. For a second example (to be taken up again when we discuss focused proof search in §3) we turn to Figure 11 which shows the lexical proof structures for a generalized quantifier noun phrase, a transitive verb, a determiner and a lexical noun.

Consider the sentence 'everyone likes the teacher'. In the unfocused sequent calculus **sLG**, the sequent $(np \,/\, n) \otimes n, (np \setminus s) \,/\, np, np \,/\, n, n \Rightarrow s$ has at least seven

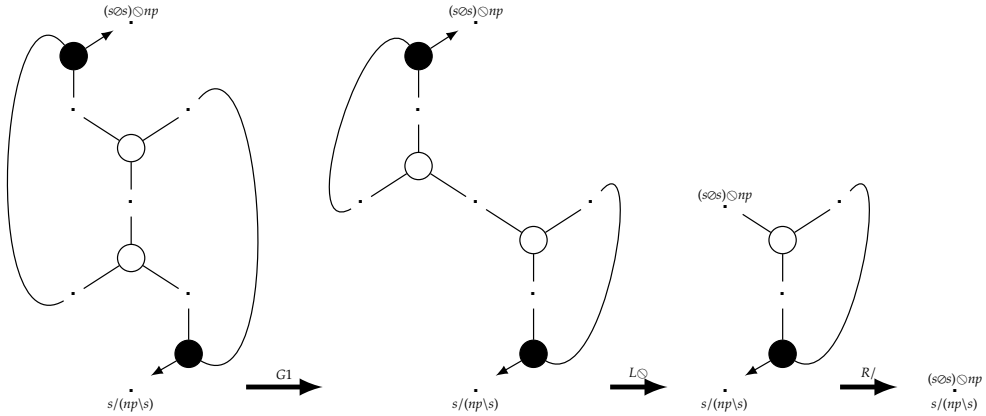


Figure 10: Reducing the abstract proof structure of Figure 5 to a tree.

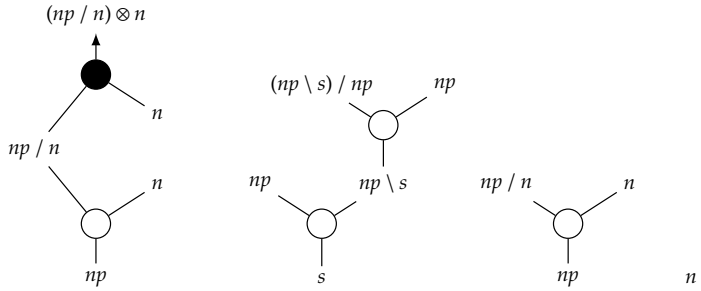


Figure 11: Lexical proof structures for a generalized quantifier noun phrase, a transitive verb, a determiner and a noun.

proofs, depending on the order of application of the introduction rules for the five occurrences of the logical connectives involved: ⊗ (once), / (three times), \ (once). Figure 12 gives, on the left, the *single* possible identification of *n* and *np* formulas that gives rise to a proof net with the lexical entries in the desired order. The corresponding abstract proof structure is given in the middle. This abstract proof structure allows us to apply a contraction directly, as shown on the right.

The table below summarizes the correspondence between proof nets and sequent proofs.

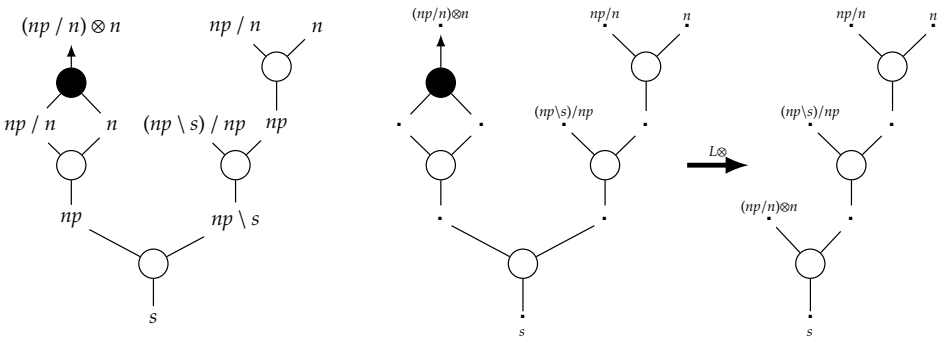


Figure 12: Judgement $(np \,/\, n) \otimes n, (np \setminus s) \,/\, np, np \,/\, n, n \Rightarrow s$: proof structure, abstract proof structure and contraction.

sequent calculus	proof structure	conversion
axiom	axiomatic formula	—
cut	cut formula	—
two-premise rule	tensor link	—
one-premise rule	cotensor link	contraction
interaction rule	—	rewrite

The invertible one-premise rules correspond to both a link and a contraction and the interaction rules are invisible in the proof structure, appearing only in the conversion sequence.

With a bit of extra effort in the sequentialization proof (and the exclusion of cuts on axioms) we can show that these correspondences are 1-to-1, that is each axiomatic formula in a proof net corresponds to exactly one axiom rule in the sequent proof, each non-invertible two-premise rule corresponds to exactly one link in the proof net and each invertible one-premise rule to exactly one link in the proof net and exactly one contraction in its conversion sequence.

Summing up, the proof net approach offers the following benefits in comparison to sequent proof search.

- Parsing. Whereas for sequent proof search the structure of the sequent has to be given, the contraction sequence that identifies a proof structure as a proof net actually *computes* this structure.
- Removal of spurious ambiguity. Proof nets, like (product-free) natural deduction, have different proof objects only for proofs of a judgement which differ essentially. The combinatorial possibilities for such readings, which are obtained by finding a complete matching of the premise and conclusion atomic formulas, can easily be enumerated for a given sequence of formulas.
- Display rules compiled away. The tensor trees associated with well-formed proof nets represent a class of sequents which is equivalent up to the display postulates.

3 Proof nets and focused display calculus

The spurious non-determinism of naive backward-chaining proof search can also be addressed within the sequent calculus itself, by introducing an appropriate notion of 'normal' derivations. In §3.1, we introduce **fLG**, a focused version of the sequent calculus for **LG**. In §3.2, we then study how to interpret focused derivations from a proof net perspective.

3.1 fLG: focused display calculus

The strategy of focusing has been well-studied in the context of linear logic, starting with the work of Andreoli Andreoli (2001). It is based on the distinction between *asynchronous* and *synchronous* non-atomic formulas. The introduction rule for the main connective of an asynchronous formula is *invertible*; it is non-invertible for the synchronous formulas. Backward chaining focused proof search starts with an asynchronous phase where invertible rules are applied deterministically until no more candidate formulas remain. At that point, a non-deterministic choice for a synchronous formula must be made: this formula is put 'in focus', and decomposed in its subformulae by means of non-invertible rules until no more non-invertible rules are applicable, at which point one reenters an asynchronous phase. The main result of Andreoli (2001) is that focused proofs are complete for linear logic.

Focused proof search for the Lambek-Grishin calculus has been studied by Bastenhof (2011) who uses a one-sided presentation of the calculus. In this section, we implement his focusing regime in the context of the two-sided sequent format of Bernardi and Moortgat (2010). We proceed in two steps. First we introduce **fLG**, the focused version of the sequent calculus of §2.1. **fLG** makes a distinction between focused and unfocused judgements, and has a set of inference rules to switch between these two. **fLG** comes with a term language that is in Curry-Howard correspondence with its derivations. This term language is a directional refinement of the $\overline{\lambda}\mu\widetilde{\mu}$ language of Curien and Herbelin (2000).

The second step is to give a constructive interpretation for **LG** derivations by means of a continuation-passing-style translation: a mapping $\lceil\cdot\rceil$ that sends derivations of the multiple-conclusion source logic to (natural deduction) proofs in a fragment of single-conclusion intuitionistic Linear Logic MILL (in the categorial terminology: **LP**). For the translation of Bastenhof (2011) that we follow here, the target fragment has linear products and negation $A^{\perp}$, i.e. a restricted form of linear implication $A \multimap \perp$, where $\perp$ is a distinguished atomic type, the response type. Focused source derivations then can be shown to correspond to distinct *normal* natural deduction proofs in the target calculus.

$$\mathbf{fLG}^{\mathcal{A}}_{/,\otimes,\backslash,\oslash,\oplus,\ominus} \xrightarrow{\lceil\cdot\rceil} \mathbf{LP}^{\mathcal{A}\cup\{\perp\}}_{\otimes,\cdot^{\perp}} \quad \left(\xrightarrow{\cdot^{\ell}} \mathbf{IL}^{\{e,t\}}_{\times,\to}\right)$$

For the linguistic illustrations in §3.2, we compose the CPS translation $\lceil\cdot\rceil$ with a second mapping $\cdot^{\ell}$, that establishes the connection with Montague-style semantic representations. This mapping sends the linear constructs to their intuitionistic counterparts, and allows *non-linear* meaning recipes for the translation of the lexical constants.

fLG: proofs and terms In the Curry-Howard proofs-as-programs tradition, we set up **fLG** starting from a term language for which the sequent logic then provides the type system. The term language encodes the *logical* steps of a derivation (left and right introduction rules, and the new set of left and right (de)focusing rules, to be introduced below); structural rules (residuation, distributivity) leave no trace in the proof terms.

Sequent structures, as in §2.1, are built out of formulas. Input formulas now are labeled with variables $x, y, z, \ldots$, output formulas with covariables $\alpha, \beta, \gamma, \ldots$.

To implement the focusing regime, we allow sequents to have one displayed formula *in focus*. Writing the focused formula in a box, **fLG** will have to deal with three types of judgements: sequents with no formula in focus (we'll call these *structural*), and sequents with a succedent or antecedent formula in focus.

$$X \vdash Y \qquad X \vdash \boxed{A} \qquad \boxed{A} \vdash Y$$

Corresponding to the types of sequents, the term language has three types of expressions: *commands*, *values* and *contexts* respectively. For commands, we use the metavariables c, C, for values v, V, for contexts e, E. The typing rules below provide the motivation for the subclassification.

$$\begin{gathered} v ::= \mu\alpha.C \mid V \quad ; \quad V ::= x \mid v_1 \otimes v_2 \mid v \oslash e \mid e \oplus v \\ e ::= \widetilde{\mu}x.C \mid E \quad ; \quad E ::= \alpha \mid e_1 \oplus e_2 \mid v \backslash e \mid e / v \\ c ::= \langle x \upharpoonleft E \rangle \mid \langle V \upharpoonright \alpha \rangle \\ C ::= c \mid \tfrac{x\ y}{z}.C \mid \tfrac{x\ \beta}{z}.C \mid \tfrac{\beta\ x}{z}.C \mid \tfrac{\alpha\ \beta}{\gamma}.C \mid \tfrac{x\ \beta}{\gamma}.C \mid \tfrac{\beta\ x}{\gamma}.C \end{gathered} \tag{7}$$

Typing rules To enforce the alternation between asynchronous and synchronous phases of focused proof search, formulas are associated with a polarity: *positive* for non-atomic formulas with invertible left introduction rule: $A \otimes B$, $A \oslash B$, $B \oplus A$; *negative* for non-atomic formulas with invertible right introduction rule: $A \oplus B$, $A \backslash B$, B / A. For atomic formulas, one can fix an arbitrary polarity. Different choices lead to different prooftheoretic behaviour (and to different interpretations, once we turn to the CPS translation). We will assume that atoms are assigned a bias (positive or negative) in the lexicon. Below the typing rules for **fLG** (restricting attention to the cut-free system).

(Co-)Axiom, (de)focusing First we have the focused version of the axiomatic sequents, and rules for focusing and defocusing which are new with respect to the unfocused presentation of §2.1. There is a polarity restriction on the formula A in these rules: the boxed formula has to be negative for CoAx, $\mu, \widetilde{\mu}^*$; for Ax, $\widetilde{\mu}, \mu^*$ it has to be positive. In the (Co-)Axiom cases, A can be required to be atomic.

$$\frac{}{x : A \vdash \boxed{x : A}}\ \mathrm{Ax} \qquad \frac{}{\boxed{\alpha : A} \vdash \alpha : A}\ \mathrm{CoAx}$$

$$\frac{X \vdash \boxed{V : A}}{\langle V \uparrow \alpha \rangle : (X \vdash \alpha : A)}\ \mu^* \qquad \frac{\boxed{E : A} \vdash X}{\langle x \downarrow E \rangle : (x : A \vdash X)}\ \widetilde{\mu}^* \tag{8}$$

$$\frac{C : (x : A \vdash X)}{\boxed{\widetilde{\mu} x.C : A} \vdash X}\ \widetilde{\mu} \qquad \frac{C : (X \vdash \alpha : A)}{X \vdash \boxed{\mu \alpha.C : A}}\ \mu$$

From a backward-chaining perspective, the $\mu, \widetilde{\mu}$ rules *remove* the focus from a focused succedent or antecedent formula. The result is an unfocused premise sequent, the domain of applicability of the invertible rules, i.e. one enters the asynchronous phase. From the same perspective, the rules $\mu^*, \widetilde{\mu}^*$ place a succedent or antecedent formula in focus, shifting control to the non-invertible rules of the synchronous phase. The $\mu^*, \widetilde{\mu}^*$ rules are in fact instances of Cut where one of the premises is axiomatic.

Invertible rules The term language makes a distinction between simple commands c (the image of the focusing rules $\widetilde{\mu}^*, \mu^*$: $\langle x \downarrow E \rangle, \langle V \uparrow \alpha \rangle$) from extended commands C. The latter start with a sequence of invertible rewrite rules replacing a logical connective by its structural counterpart. We impose the requirement that in the asynchronous phase all formulas to which an invertible rule is applicable are indeed decomposed.

$$\frac{C : (x : A \cdot \otimes \cdot y : B \vdash X)}{\frac{x\ y}{z}.C : (z : A \otimes B \vdash X)}\ \otimes L \qquad \frac{C : (X \vdash \alpha : A \cdot \oplus \cdot \beta : B)}{\frac{\alpha\ \beta}{\gamma}.C : (X \vdash \gamma : A \oplus B)}\ \oplus R$$

$$\frac{C : (x : A \cdot \oslash \cdot \beta : B \vdash X)}{\frac{x\ \beta}{z}.C : (z : A \oslash B \vdash X)}\ \oslash L \qquad \frac{C : (X \vdash x : A \cdot \backslash \cdot \beta : B)}{\frac{x\ \beta}{\gamma}.C : (X \vdash \gamma : A \backslash B)}\ \backslash R \tag{9}$$

$$\frac{C : (\beta : B \cdot \oslash \cdot x : A \vdash X)}{\frac{\beta\ x}{z}.C : (z : B \oslash A \vdash X)}\ \oslash L \qquad \frac{C : (X \vdash \beta : B \cdot / \cdot x : A)}{\frac{\beta\ x}{\gamma}.C : (X \vdash \gamma : B / A)}\ / R$$

Non-invertible rules When a positive (negative) formula has been brought into focus in the succedent (antecedent), one is committed to transfer the focus to its subformulae.

$$\frac{\boxed{e_1 : B} \vdash Y \qquad \boxed{e_2 : A} \vdash X}{\boxed{e_1 \oplus e_2 : B \oplus A} \vdash Y \cdot \oplus \cdot X}\,\oplus L \qquad \frac{X \vdash \boxed{v_1 : A} \qquad Y \vdash \boxed{v_2 : B}}{X \cdot \otimes \cdot Y \vdash \boxed{v_1 \otimes v_2 : A \otimes B}}\,\otimes R$$

$$\frac{X \vdash \boxed{v : A} \qquad \boxed{e : B} \vdash Y}{\boxed{v \backslash e : A \backslash B} \vdash X \cdot \backslash \cdot Y}\,\backslash L \qquad \frac{X \vdash \boxed{v : A} \qquad \boxed{e : B} \vdash Y}{X \cdot \oslash \cdot Y \vdash \boxed{v \oslash e : A \oslash B}}\,\oslash R \tag{10}$$

$$\frac{\boxed{e : B} \vdash Y \qquad X \vdash \boxed{v : A}}{\boxed{e / v : B / A} \vdash Y \cdot / \cdot X}\,/ L \qquad \frac{\boxed{e : B} \vdash Y \qquad X \vdash \boxed{v : A}}{Y \cdot \oslash \cdot X \vdash \boxed{e \oslash v : B \oslash A}}\,\oslash R$$

Derived inference rules: focus shifting To highlight the correspondence with the algorithm for proof net construction to be discussed in §2.2, we will use a derived rule format for shifting between a conclusion and premise focused formula. A branch from $(\widetilde{\mu}^*)$ via a sequence (possibly empty) of structural rules and rewrite rules to (μ) is compiled in a derived inference rule with the $\widetilde{\mu}^*$ restrictions on A and the μ restrictions on B.

$$\frac{\dfrac{\dfrac{\boxed{E : A} \vdash Y}{\langle x \uparrow E \rangle : (x : A \vdash Y)}\,\widetilde{\mu}^*}{\vdots}}{\dfrac{\dfrac{(\textit{res}, \textit{distr}, \textit{rewrite})}{\vdots}}{\dfrac{(\div)\langle x \uparrow E \rangle : (X \vdash \beta : B)}{X \vdash \boxed{\mu\beta.(\div)\langle x \uparrow E \rangle : B}}\,\mu}} \qquad \rightsquigarrow \qquad \frac{\boxed{E : A} \vdash Y}{X \vdash \boxed{\mu\beta.(\div)\langle x \uparrow E \rangle : B}}\,\leftrightarrows$$

For the combinations of $\mu^*, \widetilde{\mu}^*$ and $\mu, \widetilde{\mu}$, this results in the focus shifting rules below. We leave it to the reader to add the terms.

$$\frac{\boxed{A} \vdash Y}{X \vdash \boxed{B}} \leftrightharpoons \qquad \frac{X' \vdash \boxed{A}}{X \vdash \boxed{B}} \Rightarrow \qquad \frac{X \vdash \boxed{A}}{\boxed{B} \vdash Y} \rightleftharpoons \qquad \frac{\boxed{A} \vdash Y'}{\boxed{B} \vdash Y} \Leftarrow \tag{11}$$

Example 5. We illustrate the effect of the focusing regime with some alternative ways of assigning a polarity bias to atomic formulas with a simple Subject-Transitive Verb-Object sentence. Examples with lexical material filled in would be 'everyone seeks/finds a unicorn'.

$$(np/n \otimes n) \cdot \otimes \cdot ((np\backslash s)/np \cdot \otimes \cdot (np/n \cdot \otimes \cdot n)) \vdash s \tag{12}$$

For the Object we have a Determiner-Noun combination. For the Subject, we take a product type $(np/n)\otimes n$, so that we have a chance to illustrate the working of the asynchronous phase of the derivation. In the discussion of Figure 12, we saw that (12) has multiple proofs in the unfocused sequent calculus, but only one proof net, i.e. one way of matching the premise and conclusion atoms.

What about the focused calculus **fLG**? Before answering this question, we have to decide on the polarization of the atomic types. Suppose we give them uniform negative bias. There is only one focused proof then, with proof term (13): 'goal driven', top-down, to use parsing terminology. In the proof term, we write tv for the transitive verb; det for the object determiner; noun for the object common noun; subj for the subject noun phrase.

$$\begin{gathered}\mu\beta.(\frac{y\ z}{\mathsf{subj}}.\langle\ \mathsf{tv} \upharpoonright ((Q \setminus \beta) \,/\, Q')\ \rangle) \quad \text{with} \\ Q : \mu\gamma.\langle\ y \upharpoonright (\gamma \,/\, \mu\gamma'.\langle\ z \upharpoonright \gamma'\rangle)\rangle\ ,\ Q' : \mu\alpha.\langle\ \mathsf{det} \upharpoonright (\alpha \,/\, \mu\alpha'.\langle\ \mathsf{noun} \upharpoonright \alpha'\rangle)\rangle\end{gathered} \tag{13}$$

As an alternative, suppose basic type s keeps its negative bias, resetting the sentence continuation for each clausal domain, but the other basic types are assigned positive bias. We now have *two* focused derivations: 'data driven', bottom-up. To make sense of this difference, we will have to look at the CPS translation of these proofs, to be introduced below.

$$\mu\alpha.(\frac{x'\ z}{\mathsf{subj}}.\langle\ x' \upharpoonright (\widetilde{\mu}x.\langle\ \mathsf{det} \upharpoonright (\widetilde{\mu}y.\langle\ \mathsf{tv} \upharpoonright ((x \setminus \alpha) \,/\, y)\ \rangle \,/\, \mathsf{noun})\ \rangle \,/\, z)\ \rangle) \tag{14}$$

Table 1: CPS translation: non-atomic types

pol(·)							
A	B	$\lceil A \otimes B \rceil$	$\lceil A/B \rceil$	$\lceil B\backslash A \rceil$	$\lceil A \oplus B \rceil$	$\lceil A \oslash B \rceil$	$\lceil B \oplus A \rceil$
−	−	$\lceil A \rceil^\perp \otimes \lceil B \rceil^\perp$	$\lceil A \rceil \otimes \lceil B \rceil^\perp$	$\lceil B \rceil^\perp \otimes \lceil A \rceil$	$\lceil A \rceil \otimes \lceil B \rceil$	$\lceil A \rceil^\perp \otimes \lceil B \rceil$	$\lceil B \rceil \otimes \lceil A \rceil^\perp$
−	+	$\lceil A \rceil^\perp \otimes \lceil B \rceil$	$\lceil A \rceil \otimes \lceil B \rceil$	$\lceil B \rceil \otimes \lceil A \rceil$	$\lceil A \rceil \otimes \lceil B \rceil^\perp$	$\lceil A \rceil^\perp \otimes \lceil B \rceil^\perp$	$\lceil B \rceil^\perp \otimes \lceil A \rceil^\perp$
+	−	$\lceil A \rceil \otimes \lceil B \rceil^\perp$	$\lceil A \rceil^\perp \otimes \lceil B \rceil^\perp$	$\lceil B \rceil^\perp \otimes \lceil A \rceil^\perp$	$\lceil A \rceil^\perp \otimes \lceil B \rceil$	$\lceil A \rceil \otimes \lceil B \rceil$	$\lceil B \rceil \otimes \lceil A \rceil$
+	+	$\lceil A \rceil \otimes \lceil B \rceil$	$\lceil A \rceil^\perp \otimes \lceil B \rceil$	$\lceil B \rceil \otimes \lceil A \rceil^\perp$	$\lceil A \rceil^\perp \otimes \lceil B \rceil^\perp$	$\lceil A \rceil \otimes \lceil B \rceil^\perp$	$\lceil B \rceil^\perp \otimes \lceil A \rceil$

$$\mu\alpha.(\frac{x'\ z}{\mathsf{subj}}.\langle\ \mathsf{det} \upharpoonright (\widetilde{\mu}y.\langle\ x' \upharpoonright (\widetilde{\mu}x.\langle\ \mathsf{tv} \upharpoonright ((x \setminus \alpha) / y)\ \rangle / z)\ \rangle / \mathsf{noun})\ \rangle) \tag{15}$$

CPS translation Let us turn then to the translation that associates the proofs of the multiple-conclusion source logic **fLG** with a constructive interpretation, i.e. a linear lambda term of the target logic MILL/**LP**. CPS translations for **LG** were introduced in Bernardi and Moortgat (2007; 2010), who adapt the call-by-value and call-by-name regimes of Curien and Herbelin (2000) to a directional environment. The translation of Bastenhof (2011) (following Girard (1991)) is an improvement in that it avoids the 'administrative redexes' of the earlier approaches: the image of **LG** source derivations, under the mapping from Bastenhof (2011) that we present below, are *normal* **LP** terms.

The target language, on the type level, has the same atoms as the source language, and in addition a distinguished atom $\perp$, the response type. Complex types are linear products $- \otimes -$ and a defined negation $A^\perp \doteq A \multimap \perp$. The CPS translation $\lceil \cdot \rceil$ maps **fLG** source types, sequents and their proof terms to the target types and terms in Curry-Howard correspondence with normal natural deduction proofs.

Types For positive atoms, $\lceil p \rceil = p$, for negative atoms $\lceil p \rceil = p^\perp$. For complex types, the value of $\lceil \cdot \rceil$ depends on the polarities of the subtypes as shown in Table 1.

Terms The action of $\lceil\cdot\rceil$ on terms is given in (16). We write $\widetilde{x}, \widetilde{\alpha}$ for the target variables corresponding to source x, α. The (de)focusing rules correspond to application/abstraction in the target language. Non-invertible (two premise) rules are mapped to linear pair terms; invertible rewrite rules to the matching deconstructor, the **case** construct (φ, ψ, ξ metavariables for the the (co)variables involved).

$$\begin{array}{rcl} \textit{(co)var} & \lceil x\rceil = \widetilde{x} \quad ; \quad \lceil\alpha\rceil = \widetilde{\alpha} & \\ \textit{linear application} & \lceil\langle x \rangle E\rangle\rceil = (\widetilde{x}\,\lceil E\rceil) \quad ; \quad \lceil\langle V \rangle \alpha\rangle\rceil = (\widetilde{\alpha}\,\lceil V\rceil) & \\ \textit{linear abstraction} & \lceil\widetilde{\mu}x.C\rceil = \lambda\widetilde{x}.\lceil C\rceil \quad ; \quad \lceil\mu\alpha.C\rceil = \lambda\widetilde{\alpha}.\lceil C\rceil & (16) \\ \textit{linear pair} & \lceil\varphi\#\psi\rceil = \langle\lceil\varphi\rceil, \lceil\psi\rceil\rangle \quad (\# \in \{\otimes, /, \backslash, \oplus, \oslash, \oplus\}) & \\ \textit{case} & \lceil{}^{\varphi\ \psi}_{\xi}.C\rceil = \textbf{case}\ \widetilde{\xi}\ \textbf{of}\ \langle\widetilde{\varphi}, \widetilde{\psi}\rangle.\lceil C\rceil & \end{array}$$

Sequents For sequent hypotheses/conclusions, we have

$\mathsf{pol}(A)$	$\lceil x : A\rceil$	$\lceil\alpha : A\rceil$
$+$	$\widetilde{x} : \lceil A\rceil$	$\widetilde{\alpha} : \lceil A\rceil^{\perp}$
$-$	$\widetilde{x} : \lceil A\rceil^{\perp}$	$\widetilde{\alpha} : \lceil A\rceil$

(17)

Table 1 then specifies how the translation extends to sequents (replace logical connectives by their structural counterparts, and target $\otimes$ by the comma for multiset union).

$$\begin{array}{rcl} \lceil C : (X \vdash Y)\rceil & = & \lceil X\rceil, \lceil Y\rceil \vdash_{\mathbf{LP}} \lceil C\rceil : \perp \\ \left\lceil X \vdash \boxed{v : A}\right\rceil & = & \lceil X\rceil \vdash_{\mathbf{LP}} \lceil v\rceil : \lceil A\rceil \\ \left\lceil \boxed{e : A} \vdash Y\right\rceil & = & \lceil Y\rceil \vdash_{\mathbf{LP}} \lceil e\rceil : \lceil A\rceil^{\perp} \end{array} \qquad (18)$$

Illustrations We return to our sample derivations. In (19) one finds the CPS image of the source types for transitive verb and determiner under the different assignments of bias to the atomic subformulas, and the composition with $\cdot^{\ell}$, assuming $np^{\ell} = e$ (entities), $s^{\ell} = \perp^{\ell} = t$ (truth values) and $n^{\ell} = e \to t$ (sets of

Table 2: Constants: lexical translations

$(np^+\backslash s^-)/np^+$	finds	$\lambda\langle\langle x, c\rangle, y\rangle.(c\ (\textsc{find}^{eet}\ y\ x))$
$(np^+/n^+)\otimes n^+$	everyone	$\langle\lambda\langle x, y\rangle.(\forall\ \lambda z.(\Rightarrow\ (y\ z)\ (x\ z))), \textsc{person}^{et}\rangle$
np^+/n^+	some	$\lambda\langle x, y\rangle.(\exists\ \lambda z.(\wedge\ (y\ z)\ (x\ z)))$
n^+	unicorn	$\textsc{unicorn}^{et}$
$(np^-\backslash s^-)/np^-$	needs	$\lambda\langle\langle q, c\rangle, q'\rangle.(q\ \lambda x.(\textsc{need}^{((et)t)et}\ q'\ x))$
$(np^-/n^-)\otimes n^-$	everyone	$\langle\lambda\langle x, w\rangle.(\forall\ \lambda z.(\Rightarrow\ (w\ \lambda y.(y\ z))\ (x\ z))), \lambda k.(k\ \textsc{person}^{et})\rangle$
np^-/n^-	some	$\lambda\langle x, w\rangle.(\exists\ \lambda z.(\wedge\ (w\ \lambda y.(y\ z))\ (x\ z)))$
n^-	unicorn	$\lambda k.(k\ \textsc{unicorn}^{et})$

entities). For the lexical constants of the illustration, Table 2 gives $\cdot^\ell$ translations compatible with the typing. In Table 3, these lexical recipes are substituted for the parameters of the CPS translation.

	LG	$\lceil\cdot\rceil^\perp$	$(\lceil\cdot\rceil^\perp)^\ell$
a.	$(np^+\backslash s^-)/np^+$	$((np\otimes s^\perp)\otimes np)^\perp$	$((e\times(tt))\times e)\to t$
b.	np^+/n^+	$(np^\perp\otimes n)^\perp$	$((et)\times(et))\to t$
c.	$(np^-\backslash s^-)/np^-$	$((np^{\perp\perp}\otimes s^\perp)\otimes np^{\perp\perp})^\perp$	$((((et)t)\times(tt))\times((et)t))\to t$
d.	np^-/n^-	$(np^\perp\otimes n^{\perp\perp})^\perp$	$((et)\times(((et)t)t))\to t$

(19)

Table 3: Compositional translations

$$\lceil(13)\rceil = \lambda\widetilde{\beta}.(\mathsf{case\ subj}^\ell\ \mathsf{of}\ \langle\widetilde{y},\widetilde{z}\rangle.(\mathsf{tv}^\ell\ \langle\langle\lambda\widetilde{\gamma}.(\widetilde{y}\ \langle\widetilde{\gamma},\lambda\widetilde{\gamma}'.(\widetilde{z}\ \widetilde{\gamma}')\rangle),\widetilde{\beta}\rangle,\lambda\widetilde{\alpha}.(\mathsf{det}^\ell\ \langle\widetilde{\alpha},\lambda\widetilde{\alpha}'.(\mathsf{noun}^\ell\ \widetilde{\alpha}')\rangle)\rangle)))$$

$$\lceil(13)\rceil^\ell = \lambda c.(\forall\ \lambda x.((\Rightarrow\ (\mathsf{person}\ x))\ (c\ ((\textsc{needs}\ \lambda w.(\exists\ \lambda y.((\wedge\ (\mathsf{unicorn}\ y))\ (w\ y))))\ x))))$$

$$\lceil(14)\rceil = \lambda\widetilde{\alpha}.(\mathsf{case\ subj}^\ell\ \mathsf{of}\ \langle\widetilde{x}',\widetilde{z}\rangle.(\widetilde{x}'\ \langle\lambda\widetilde{x}.(\mathsf{det}^\ell\ \langle\lambda\widetilde{y}.(\mathsf{tv}^\ell\ \langle\langle\widetilde{x},\widetilde{\alpha}\rangle,\widetilde{y}\rangle),\mathsf{noun}^\ell\rangle),\widetilde{z}\rangle))$$

$$\lceil(14)\rceil^\ell = \lambda c.(\forall\ \lambda x.((\Rightarrow\ (\textsc{person}\ x))\ (\exists\ \lambda y.((\wedge\ (\textsc{unicorn}\ y))\ (c\ ((\textsc{likes}\ y)\ x))))))$$

$$\lceil(15)\rceil = \lambda\widetilde{\alpha}.(\mathsf{case\ subj}^\ell\ \mathsf{of}\ \langle\widetilde{x}',\widetilde{z}\rangle.(\mathsf{det}^\ell\ \langle\lambda\widetilde{y}.(\widetilde{x}'\ \langle\lambda\widetilde{x}.(\mathsf{tv}^\ell\ \langle\langle\widetilde{x},\widetilde{\alpha}\rangle,\widetilde{y}\rangle),\widetilde{z}\rangle),\mathsf{noun}^\ell\rangle))$$

$$\lceil(15)\rceil^\ell = \lambda c.(\exists\ \lambda y.((\wedge\ (\textsc{unicorn}\ y))\ (\forall\ \lambda x.((\Rightarrow\ (\textsc{person}\ x))\ (c\ ((\textsc{likes}\ y)\ x))))))$$

3.2 Proof nets and focusing

We saw in §3.1 that **fLG** may allow multiple derivations from one and the same set of (co)axiom judgements. These derivations would be identified under the proof net perspective of §2.2. To establish the correspondence with **fLG** derivations, we introduce term-labeled proof nets, and show how a proof term can be read off from the *composition graph* associated with a net.

Our approach is comparable to the algorithm of de Groote and Retoré (1996), which computes a linear lambda term from a traversal of the dynamic graph associated with a proof net for a derivation in the Lambek calculus **L**. For single-conclusion **L**, the term associated with a given proof net is unique; in the case of multiple-conclusion **LG** the term computation algorithm may associate more than one term with a proof net. These multiple results will then be shown to correspond to the derivational ambiguity of focused proof search.

Reduction tree In order to analyse the structure of a conversion sequence in more detail, we introduce the notion of a proof net *component*:

Definition 3.1. Given a proof net P, a *component* C of P is a maximal subnet of P containing only tensor links.

From a proof net, we can obtain its components by simply erasing all cotensor links. The components will be the connected components (in the graph-theoretic sense) of the resulting graph. To simplify the following discussion, unless otherwise indicated, we will use the word *component* to refer only to components containing at least one tensor link.

When P is a proof net (and therefore converts to a tensor tree using a sequence ρ of conversions and contractions) the components of P can bee seen as a parallel representation of the synchronous phases in sequent proof search. In ρ, all interaction rules operate in one component C, the cotensor rules and the corresponding contractions join two different components (though the component connected to the main vertex can be trivial here). When multiple cotensor links have both active tentacles attached to a single component (Figure 10 shows an example), we apply all contractions simultaneously, repeating this process until no further contractions apply.

So instead of seeing ρ as a *sequence* of reductions, we can see it as a rooted *tree* of reductions: the initial components are its leaves (synchronous phases) and

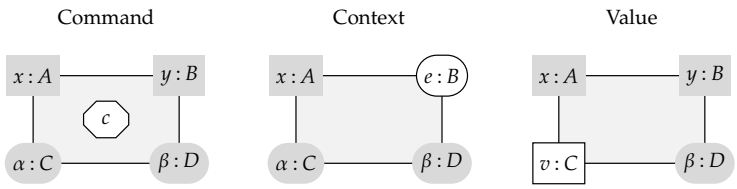


Figure 13: Proof nets with term labels: commands, context and values

the contractions, which join components, are its branches (the branches from the active components to their parents correspond to asynchronous phases) and the final tree — a single component — is its root (we will see an example in Figure 19 below). Note that this same observation is essential to the cut elimination proof of Moot and Puite (2002).

Nets and term labeling When assigning a term label to a proof net, our algorithms will assign labels to larger and larger subnets of a given proof net, until we have computed a term for the complete proof net. Like in the sequent calculus, we distinguish between subnets which are commands, contexts and values. Figure 13 shows how we will distinguish these visually: the main formula of a subnet is drawn white, other formulas are drawn in light gray, values are drawn inside a rectangle, contexts inside an oval.

Figure 14 gives the term-labeled version of the proof net links corresponding to the logical rules of the sequent calculus. The flow of information is shown by the arrows: information flow is always from the active formulas to the main formula of a link, and as a consequence the complex term can be assigned either to a conclusion or to a premise of the link. This is the crucial difference with term labeling for the single-conclusion Lambek calculus, where the complex term is always assigned to a conclusion. The cotensor rules, operating on commands, indicate the prefix for the command corresponding to the term assignment for the rule (we will see later how commands are formed).

The proof term of an **LG** derivation is computed on the basis of the *composition graph* associated with its proof net.

Definition 3.2. Given a proof net P, the associated *composition graph* $cg(P)$ is obtained as follows.

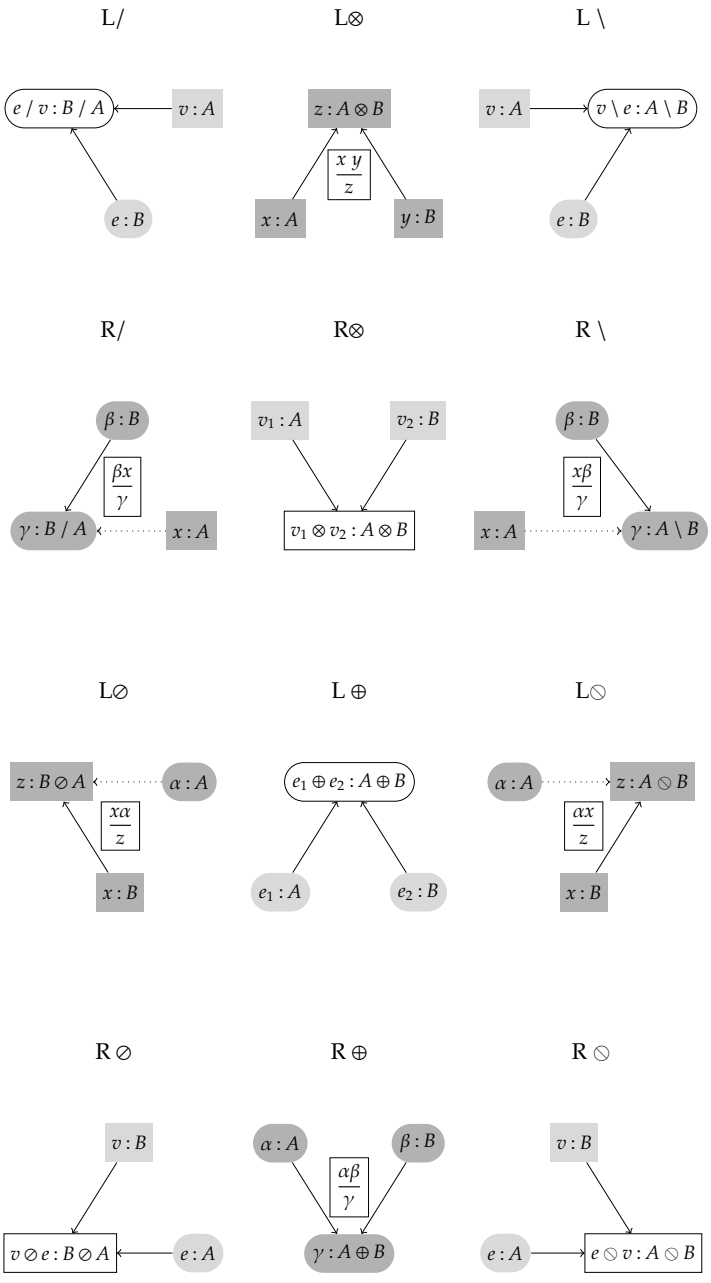


Figure 14: **LG** links with term labeling

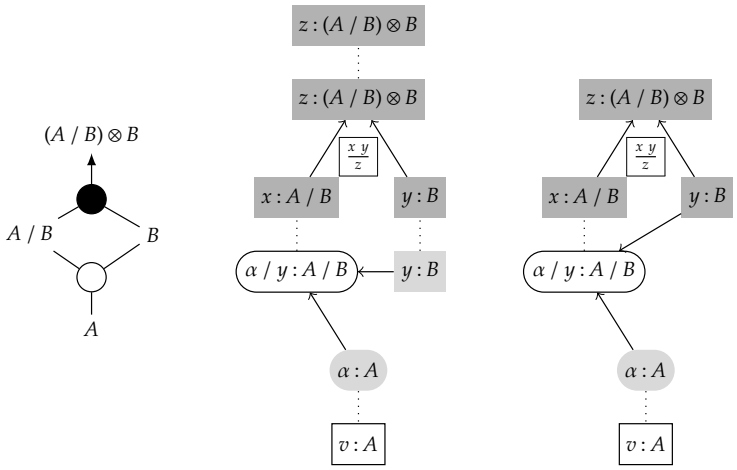


Figure 15: Proof net, initial composition graph, reduced composition graph.

1. all vertices of P with formula label A are expanded into *axiom links*: edges connecting two vertices with formula label A; all links are replaced by the corresponding links of Figure 14;
2. all vertices in this new structure are assigned atomic terms of the correct type (variable or covariable) and the terms for the tensor rules are propagated from the active formulas to the main formula;
3. all axiom links connecting terms of the same type (value or context) are collapsed.

Figure 15 gives an example of the composition graph associated with a net. In all, the expansion stage gives rise to four types of axiom links, depending on the type of the term assigned to the A premise and the A conclusion. These cases are summarized in Figure 16. The substitution links are collapsed in the final stage of the construction of the composition graph (shown on the right of Figure 15; the command and $\mu/\widetilde{\mu}$ cases are the ones that remain.

Given the composition graph $cg(P)$ associated with a proof net P, we compute terms for it as follows.

1. we compute all maximal subnets of $cg(P)$, which consist of a set of tensor links with a single main formula, marking all these links as visited;

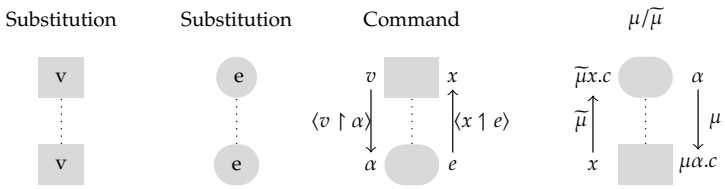


Figure 16: Types of axiom links

2. while $cg(P)$ contains unvisited links do the following:
   (a) follow an unvisited command link attached to a previously calculated maximal subnet, forming a correct command subnet; like before, we restrict to *active* subnets which do not contain (or allow us to reach through an axiom) the main formula of a negative link;
   (b) for each negative link with both active formulas attached to the current command subnet, pass to the main formula of the negative link, forming a new command, repeat this step until no such negative links remain attached;
   (c) follow a μ or $\widetilde{\mu}$ link to a new vertex, forming a larger value or context subnet and replacing the variable previously assigned to the newly visited vertex by the μ value or $\widetilde{\mu}$ context.

The algorithm stays quite close to the focused derivations of the previous section: the maximal subnets of step 1 are *rooted* versions of the components we have used before, with the directions of the arrows potentially splitting components into multiple rooted components (Figure 18 will give an example) and the asynchronous phases, which consisted of one or more contractions for cotensor links, will now consist of a passage through a command link, followed by zero or more cotensor links, followed by either a μ or a $\widetilde{\mu}$ link, the result being a new, larger subnet. The term assignment algorithm is a way to enumerate the non-equivalent proof terms of a net. Given that these terms are isomorphic to focused sequent proofs, it is no coincidence that the computation of the proof terms looks a lot like the sequentialisation algorithm.[4]

Lemma 1. *If P is a proof net (with a pairing of command and $\mu/\widetilde{\mu}$ links) and v is a term calculated for P using this pairing then there is a sequent proof π which is assigned v*

[4]The connection between proof net sequentialisation and focusing for linear logic is explored in Andreoli and Maieli (1999)