Categorical Foundations for Extended Compositional Distributional Models of Meaning

MSc Thesis *(Afstudeerscriptie)*

written by

Gijs Wijnholds
(born May 6th, 1990 in Zwolle, The Netherlands)

under the supervision of **Prof Dr Michael Moortgat** and **Dr Raquel Fernandez**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:
December 19, 2014

Members of the Thesis Committee:
Dr Maria Aloni (Chair)
Prof Dr Michael Moortgat
Dr Raquel Fernandez
Dr Nick Bezhanishvili
Dr Richard Moot
Dr Mehrnoosh Sadrzadeh

Institute for Logic, Language and Computation

Abstract

Compositional distributional models of meaning were introduced by Coecke et al. (2010, 2013) with the aim of reconciling the theory of distributional meaning in terms of vector space semantics with the theory of compositional interpretation as one finds it in typelogical grammars. The particular typelogical formalisms employed by Coecke et al. (pregroup grammars, Lambek calculus) have a recognizing capacity equivalent to context-free grammars. It is well known, however, that natural languages exhibit patterns that require expressivity beyond context-free (Huybregts, 1984; Shieber, 1987). The aim of this thesis, then, is to investigate extensions of compositional distributional models of meaning that result from using typelogical grammars with enhanced expressivity. To this end, we give a categorical characterization of the Lambek-Grishin Calculus (see Moortgat (2007, 2009) and references there) and its constituting subsystems in terms of *linear distributive categories* borrowing a categorification technique from Lambek (1968). We develop a language to reason graphically about morphism structure and equality in terms of *string diagrams*. Finally, we show that finite-dimensional vector spaces are also an instance of linear distributive categories, which creates the possibility of *extended* compositional distributional models of meaning.

Acknowledgements

Firstly, I would like to express my gratitude towards my first supervisor, Michael Moortgat, for having the patience to supervise this project. Without his constant support, whether it be content wise or TEXnically, I would not have been capable of writing this thesis. Second of all, my thanks go out to my second supervisor, Raquel Fernandez, for encouraging me to take a break every now and then, and for giving helpful comments on earlier drafts of this document.

I would like to thank the remainder of my thesis committee, Maria Aloni, Nick Bezhanishvili, Mehrnoosh Sadrzadeh and Richard Moot for taking the time to read this lengthy document and to attend my thesis defense. Special thanks go out to John Baez and Peter Selinger for providing helpful comments and insights regarding string diagrams.

I also wish to thank Ulle Endriss, Michael Franke, and Raquel Fernandez for making practical arrangements to help me finish the Master of Logic after almost a year of absence. Lastly, I wish to thank Tanja Kassenaar and Gina Beekelaar from the ILLC staff for assisting me whenever necessary with any practical issues.

Furthermore, I owe special thanks to the following friends: Jim Keyni, for showing me around in Amsterdam and for the visits in Utrecht. Rob, for the endless amount of table tennis matches. Bram and Jeroen, for showing me a different side of academia. Sander, for all the fun.

I wish to thank my parents and my brother for all of the love and support they have shown throughout my life and for helping me get back on my feet again.

Finally, I especially want to thank Harma for having stood by me for the last couple of years and making my life in general a more pleasant experience.

Contents

Introduction

The analysis of natural language can be subdivided in several parts: that of the analysis of *patterns*, which we call syntax, and the analysis of *meaning association*, which we call semantics. Beyond syntax and semantics proper, there is the realm of pragmatics, the analysis of *meaning in context* rather than a conventional, static meaning. For the purpose of this study, syntax and semantics are already enough of a challenge. Form and meaning should not be considered in isolation; it is a common understanding that these two aspects of natural language are highly interdependent. Providing the link between syntax and semantics is providing the *syntax-semantics interface*, a method describing how the process of putting together syntactic patterns provides information as to how the meaning of these patterns should be assembled. A crucial, desirable feature of the interface between form and meaning is *compositionality*, which roughly states that

The meaning of a complex expression is given by the meaning of its constituent expressions and the way in which they are combined.

The categorial approach to grammatical analysis is based on the idea that linguistic expressions are assigned *types*; the *logic* for the grammatical type system then determines what the syntactically well-formed combinations of expressions are. The syntax-semantics interface is modelled along the lines of the *Curry-Howard correspondence*. Orinigally developed in the context of intuitionistic logic, the CH correspondence allows one to associate logical derivations with terms of the lambda calculus, hence the slogan 'proofs as programs'. In the application to grammars, the terms associated with a derivation serve as 'semantic recipes' prescribing how the meaning of a complex expression is to be computed out of the meaning of its constituent parts. A standard way of setting up semantic *models* for typelogical grammars is to adopt the set-theoretic view of Montague Grammar (after (Montague, 1970b,a)): one assumes a fixed domain of entities and of truth-values on which one then defines set-theoretical constructions that give the desired meanings of complex expressions via compositionality. The problem with this approach is that the set-theoretic interpretation of the basic expressions (words) is *predefined*.

A seemingly opposite approach to semantics is that of distributional semantics, which is based on the principle that "You shall know a word by the company it keeps" (Firth, 1957): one extracts, from a large corpus, co-occurrence counts for words and so builds vectors that represent the meaning of words. In this way, the *similarity* of words can be measured through an appropriate inner product. Distributional semantics avoids the key problem of Montagovian semantics that there be predefined meanings associated to basic expressions, reinstating the empirical nature of linguistic meaning. However, compositionality is now not guaranteed: there is no automatic way of defining how the semantics of basic expressions should be combined to form the meaning of larger, more

complicated expressions.

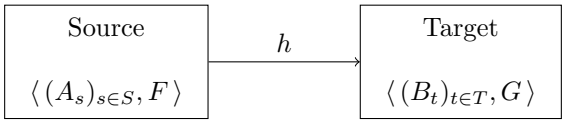
A first question that arises then, is how to combine typelogical grammar with its nice mathematical properties with the distributional view on lexical semantics. That such a combination is indeed possible is show by recent research of Coecke et al who rely on the similar mathematical structure of pregroups (a particular typelogical grammar) and finite dimensional vector spaces (Coecke et al., 2010) or on the similar mathematical structure of Lambek monoids and finite dimensional vector spaces (Coecke et al., 2013). Such similar structure implicitly relies on an extension of the Curry-Howard correspondence, initiated by Lambek and Scott in their book (Lambek and Scott, 1988), on which we will elaborate below.

Combining a substructural logic such as the Lambek Calculus with vector space semantics gives models that we will call *basic compositional distributional models of meaning*. Basic, because they rely on the Lambek Calculus, the typical "logic of grammatical composition". We shall consider deploying this very logic a weakness of the model, for the reason that the patterns that we are able to describe with this logic do not encompass all possible patterns present in natural language. It has indeed been argued that the context-free languages, the class of patterns describable by context-free grammars as well as Lambek grammars, lack the necessary expressivity (Huybregts, 1984; Shieber, 1987). We therefore raise the following problem, which will be the central theme in this study: *how can we extend compositional distributional models of meaning in such a way that we can describe patterns beyond context-freeness and associate meaning to them?*

Our approach in this thesis will then be to consider *extensions* of the Lambek Calculus that have the proper expressivity, including at least the *mildly context-sensitive languages* (Joshi et al., 1990). The sought extensions should exhibit a mathematical structure similar to that of finite dimensional vector spaces, making it possible to define *extended compositional distributional models of meaning*. Before outlining the structure of this thesis, we give some context to place the problem in its proper setting.

Compositionality and Type-Logical Grammar

The intuitive view of compositionality that we gave at the beginning of this section assumes that (a) the meaning of the basic lexical expressions is given and that (b) the meaning of non-basic expressions can be systematically obtained from "the way in which they are combined" syntactically. This intuitive view is made more precise in (Hendriks, 2001), elaborating on (Montague, 1970b). In short, Syntax and Semantics are modelled as multisorted algebras, and compositional interpretation takes the form of a *homomorphism*. i.e. a mapping from source (syntax) to target (semantics) that respects the sorts and the operations. In a picture:



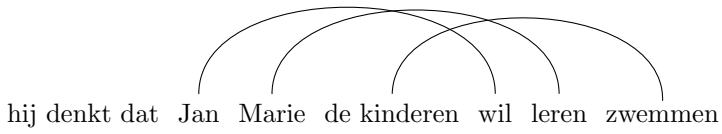
$$h(f(a_1, \ldots, a_n)) = g(h(a_1), \ldots, h(a_n))$$

where g is the semantic operation at the target end corresponding to the syntactic operation f.

The benefit of compositionality is immediate: only the semantics of basic expressions is needed to obtain the semantics for larger, complex expressions. This implies that one only needs a finite specification of a *dictionary* in order to generate an infinite amount of linguistic structures together with the corresponding interpretations.

Typelogical grammar precisely assumes compositionality as being a homomorphism from the derivational term algebra to semantics, having a logical system as the grammatical framework, on which one can easily graft a semantics that follows the structure of the complex expressions in the language. The problem then resides in lexical semantics: how does one attribute a meaning to single words? A standard tool, initiated by Montague in the '70s (Montague, 1970b), is to employ a set-theoretic lexical semantics, in which one assumes a domain of entities and a domain of truth-values on which relations are defined. The meaning of the word *man* in this setting would be precisely the set of entities that are men, or equivalently, the characteristic function that maps all men to truth value 1 and all other entities to truth value 0; the meaning of the word *the* in combinatin with a noun is a function that picks out the unique individual that has the property denoted by the noun if there is such a unique individual, and nothing otherwise.

Non-local composition is a pervasive feature of language. Typical examples include non-periphal extraction and crossing dependencies. The former is exemplified by a expression such as "the book that John found in the library". The relative pronoun "that" in this case has to establish a semantic dependency with the direct object of "found", but this direct object is hidden within the relative clause, and inaccessible for external inspection. An example of crossing dependencies in Dutch is "(Ik weet) dat Jan Marie de kinderen zag leren zwemmen" (I know that John saw Mary teaching the kids how to swim). In this case the semantic dependencies can be represented by the following picture



which is a typical example of a pattern unrecognizable by context-free grammar, the copy language w^2.

There have been several proposals to deal with non-local composition, all trying to find the proper balance between computational complexity and expressivity. Examples are Tree Adjoining Grammars (Joshi, 1985), Multiple Context-Free Grammars (Seki et al., 1991) and Minimalist Grammars (Stabler, 2011). These are examples of proposals that aim to describe the *Mildly Context-Sensitive Languages*. On the side of typelogical grammar there have been several developments since the '80s, all extending the Lambek Calculus in some way (see (Moortgat, 2011) for an overview). Here we find multimodal systems (Moortgat, 1996), Displacement Calculus (Morrill et al., 2011), Combinatory Categorial Grammar (Steedman, 2000) and the Lambek-Grishin Calculus (Moortgat, 2009). Although the Lambek Calculus itself has a nice categorical characterization, the categorical structure of these extensions is not very well understood. The aim then, is to find a nice categorical description of at least one of these extensions. We will focus on the Lambek-Grishin Calculus, a symmetric extension of the Lambek Calculus. The benefit of focusing on this system is that it also has a nice categorical interpretation in terms of *linear distributive bi-clopen categories*, a concept

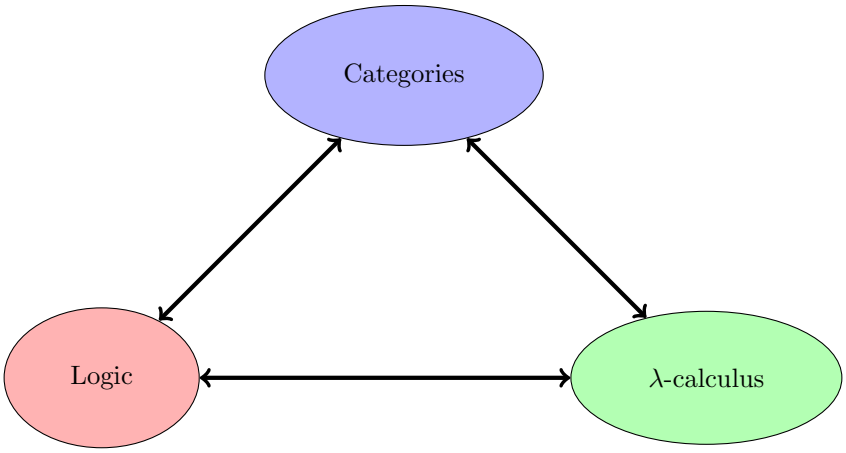
related to but in key aspects different from that of Cockett and Seely (1997b). We will define linear distributive bi-clopen categories in the second part of the thesis.

A Classic: the CHL Correspondence

As said above, the Curry-Howard correspondence[1] states an isomorphim between proofs in intuitionistic propositional logic and terms of the simply typed lambda calculus. The following table gives an impression of how the different concepts of logic relate to the different concepts in lambda calculus:

Logic	Lambda Calculus
formula	type
proof	program
normalization	β-reduction

As was shown by Lambek and Scott (1988), this correspondence can be elevated to the level of *categories*, and hence it goes by the name *Curry-Howard-Lambek* (CHL) correspondence. It establishes an *equivalence of categories*[2] between cartesian closed categories and typed lambda calculi with products. The great benefit of applying such a correspondence to other kinds of categories is that other kinds of logics can be seen to (categorically) be "essentially the same as" their associated category, meaning that we can also broaden our options for a syntax-semantics interface that incorporates compositionality via homomorphic passages from the type logic to the associated semantic category. The CHL correspondence is nicely shown in the following picture:



The corresponding table of concepts is shown below:

[1] Extensively reviewed by Sørensen and Urzyczyin (2006)
[2] This is a loose version of an isomorphism to be defined in Chapter 1.

Logic	Category Theory	Lambda Calculi
formula	object	type
proof	morphism	program
equivalence of proofs	morphism equality	equivalence of programs

The Lambek Calculus is a substructural logic that is both *linear* and *ordered*: it lacks the rules of weakening, contraction and exchange and splits implication into a left and right implication (thereby respecting the order of composition). So, what then are the ingredients for a CHL correspondence for the Lambek Calculus and its relatives? Because of its linearity and ordering, it is immediate the corresponding lambda calculus and type of category must accomodate this. Although we will not touch the first part of the correspondence (simply because it is not immediately relevant), Wansing has developed a lambda calculus that correspond to the Lambek Calculus (Wansing, 1992). On the categorical side we will show in this study that various kinds of *closed categories* will be suitable for interpreting the different incarnations of the Lambek Calculus.

The implications of the CHL correspondence are that one can, instead of interpreting a logic in its corresponding lambda calculus, use *any kind* of mathematical structure that is an instance of the corresponding category. We will see that this makes it possible to interpret a typelogical system such as the Lambek-Grishin Calculus in finite dimensional vector spaces, thus realizing an extended compositional distributional model of meaning.

Graphical Reasoning in Logic and Categories

With the introduction of linear logic (Girard, 1987) came the introduction of *proof nets*. Proof nets are graphical representations of sequent proofs that remove spurious ambiguity: going from sequent systems to natural deduction requires a many-to-one mapping that thus identifies a great deal of sequent proofs. Proof nets avoid this by implicitly representing several sequent proofs by the same net. Proof nets for the Lambek Calculus have been studied intensively (Roorda, 1991; Moot, 2002) and consequently, proof nets have been developed for the multimodal Lambek Calculus (Moot and Puite, 2002) and for the Lambek-Grishin Calculus (Moortgat and Moot, 2012).

On the side of categories, several graphical representations have been examined under the name of *string diagrams*. Here, the morphisms of the category in question can be represented graphically and one defines the appropriate equations on diagrams in order to have a *coherent* (i.e. sound and complete) language to reason graphically instead of chaining equations. A nice introduction to graphical reasoning in categories is (Selinger, 2011).

Obviously, considering the CHL correspondence, there is a close connection between proof nets and string diagrams, as research has shown (Straßburger and Lamarche, 2004; Blute et al., 1996). Given that one enforces the proper equations on proof nets, it can be shown that they will form the morphisms of the free category in question. For instance, the proof nets for multiplicative linear logic with units generate the free *-autonomous category (Straßburger and Lamarche, 2004). In this thesis we discuss graphical calculi for various kinds of closed categories. We first review the graphical calculus developed for monoidal closed categories by Baez and Stay (2011). Our new contribution will be to consider non-associative systems, which allow us to represent information

not just as strings but as binary trees. As a consequence, we need a whole new concept of graphical language, which we fully develop for these closed categories.

Recap: Problem Statement in Context

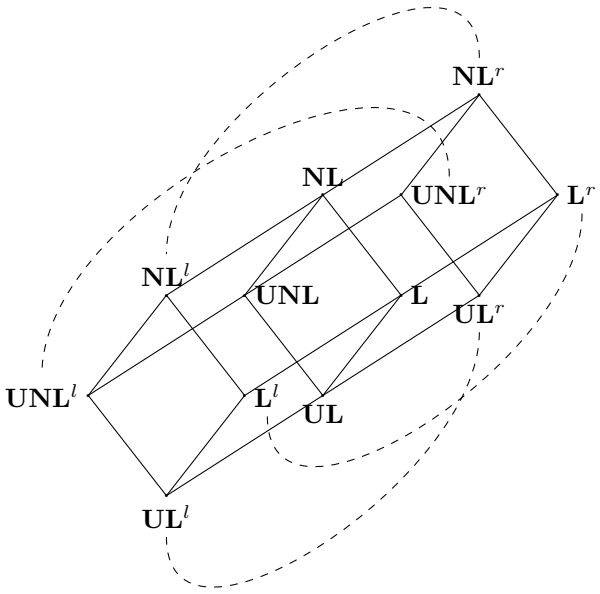
To summarize, the problem we have raised is as follows: because the existing compositional distributional models of meaning are to weak in terms of their capabilities to analyze natural language patterns, there is a need for *extended compositional distributional models of meaning*. Having the Curry-Howard-Lambek correspondence as our guiding light, we find that to develop such models we require the following: we should find a suitable extension of the Lambek Calculus, powerful enough to describe the mildly context-sensitive languages and exhibiting a categorical structure that is *interpretable* in finite dimensional vector spaces. Next to these objectives, we want to develop graphical languages for the categorical structures we find, and explore a bit the semantics of the basic and the extended models.

What This Thesis is Not About

This thesis introduces a new framework for compositional distributional models of meaning. An important aspect of these models is their empirical nature; this thesis prepares the way for empirical validation of more refined compositional distributional meaning models. Actually carrying out experiments to validate the power of these models in for instance sense disambiguation and sentence similarity would be a thesis subject of its own: one might fix a grammar and then extract from a corpus the vector space semantics and see how the extended models behave with respect to the basic models. However, this still has the problem of having to predefine the lexical type declarations. Thus, it would be better to also extract the grammar (categorially: the lexicon) out of the corpus via *grammar induction*. Doing grammar induction will obviously also show the difference between different syntactic backbones used. To perform multiple experiments with different set-ups would take quite some time, and we therefore leave it to future work. We will also point this out in our conclusion section.

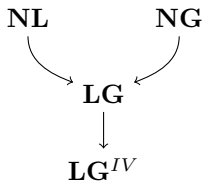
Overview of Type-Logics

This thesis discusses many different type logics and we clarify the relationships between these systems in a picture. The variants of the basic Lambek Calculus are the non-associative Lambek Calculus **NL**, its associative variant **L** and the variants obtained by adding units to either of the former systems, giving the unitary non-associative Lambek Calculus **UNL** and the unitary associative Lambek Calculus **UL**. These four systems give rise to a diamond indicating the relation between them; the connectives of these systems are such that we can split the systems into left and right variants, indicated by superscripting an l or r. The following diagram summarizes all this:



where the dashed lines indicate isomorphisms between systems. The dual Grishin systems are obtained by replacing the **L** by a **G** and give rise to a similar diagram.

Finally, one obtains the Lambek-Grishin system by merging the systems **NL** and **NG** and adding interaction postulates [3] between the two systems:



where the arrows indicate that each system is a part of the system it is pointing to. Of the systems **NL** and **L** (and their unital variants) it is known that they are complete with respect to (unital) *residuated groupoids* and (unital) *residuated semigroups* respectively (see Buszkowski (1986)). For an overview of algebraic semantics for substructural logics in general, see the book by Galatos et al. (2007). We will show in this thesis that the logics under discussion all correspond to certain categorical notions to be defined in the first chapter of each part. Just as the system **MLL** of multiplicative linear logic with units corresponds to *-autonomous categories (Blute and Scott,

[3] The particular interaction postulates we will consider are the type IV interactions amongst a range of possible postulates formulated by Grishin (1983)

2004) and intuitionistic propositional logic corresponds to cartesian closed categories (Lambek and Scott, 1988), we establish correspondences according to the following tables:

$\mathbf{NL}^{(l/r)}$	left/right/bi-closed tensor categories ($\mathbf{(L/R/B)CC}_{st}$)
$\mathbf{UNL}^{(l/r)}$	left/right/bi-closed unitary tensor categories ($\mathbf{U(L/R/B)CC}_{st}$)
$\mathbf{L}^{(l/r)}$	left/right/bi-closed associative tensor categories ($\mathbf{A(L/R/B)CC}_{st}$)
$\mathbf{UL}^{(l/r)}$	left/right/bi-closed monoidal categories ($\mathbf{M(L/R/B)CC}_{st}$)

$\mathbf{NG}^{(l/r)}$	left/right/bi-open tensor categories ($\mathbf{(L/R/B)OC}_{st}$)
$\mathbf{UNG}^{(l/r)}$	left/right/bi-open unitary tensor categories ($\mathbf{U(L/R/B)OC}_{st}$)
$\mathbf{G}^{(l/r)}$	left/right/bi-open associative tensor categories ($\mathbf{A(L/R/B)OC}_{st}$)
$\mathbf{UG}^{(l/r)}$	left/right/bi-open monoidal categories ($\mathbf{M(L/R/B)OC}_{st}$)

Finally the last table shows the correspondences for the Lambek-Grishin system:

$\mathbf{LG}_{\emptyset}$	bi-clopen tensor categories ($\mathbf{BCOC}_{st}$)
$\mathbf{LG}^{IV}$	linear distributive tensor categories ($\mathbf{LDTC}_{st}$)

Contributions and Structure of the Thesis

In this thesis, we develop a uniform framework for doing compositional distributional semantics guided by the work of Coecke et al. (Coecke et al., 2010, 2013). More specifically, in part I we review and expand where necessary the theory of basic categorical compositional distributional models of meaning by the following chapters:

Chapter 1 We introduce basic category theory and categories with additional structure.

Chapter 2 We introduce graphical languages for categories with additional structure. The first contribution is the development of a coherent graphical language for non-associative systems.

Chapter 3 We introduce Lambek's Syntactic Calculus and present its "categorification".

Chapter 4 We review finite-dimensional vector spaces as a system of doing distributional semantics and show how a compositional distributional model of meaning could be obtained.

Part II of the thesis tries to replicate the structure of part I while giving an extension of the basic model. Thus, we extend the theory of categorical foundations for compositional distributional models of meaning within the following chapters:

Chapter 5 We introduce co-closed (or open) categories and non-associative linearly distributive categories, the latter in line with the development of the Lambek-Grishin Calculus.

Chapter 6 We associate a graphical language with linearly distributive categories and prove a coherence theorem for it, our other novel contribution to the theory of string diagrams.

Chapter 7 We review the Lambek-Grishin Calculus and investigate its categorical structure.

Chapter 8 We investigate how the categorified Lambek-Grishin Calculus can be interpreted in finite-dimensional vector space semantics.

Part I

Basic Compositional Distributional Models of Meaning

Chapter 1

Categories

In this chapter, we review basic category theory and various kinds of *closed* categories and their functors. The very basics of category theory are outlined, following the definitions from Blute and Scott (2004) and Awodey (2006). Then we move on to define *closed tensor categories* and *monoidal closed categories*, the latter following the definition of Selinger (2011). Finally we define *functors with structure* and discuss the symmetry between left and right closed categories.

A category essentially is an *abstraction* over mathematical structures: it contains objects and arrows between objects, the latter of which can be composed to construct new arrows. Additionally some evident axioms need to be satisfied: there must be *identity* arrows for every object and composition of arrows should be *associative*. From this concept of category, one can go on to define *arrows between categories*, these are called *functors*. Then one will want to define *arrows between functors*, to be called *natural transformations*. The nice thing about the theory of categories is that we can view functors as arrows between categories, but also as the arrows of a category that has categories as objects, or we may even think of them as objects of a category, the arrows now being the natural transformations. These shifts in viewpoint are characteristic (and may lead to confusion) for category theory. Some additional concepts include that of *adjunction*, which will be a key concept in the rest of this thesis, and the concept of *monads*, which we use to illustrate the categorical structure of the type logics we will consider. We will start out with the very basic concepts and work our way through categories with extra structure.

1.1 The Basics

The most basic definition in category theory consists of that of category:

Definition 1.1. A category $\mathbf{C}$ consists of:

- A collection of objects $Ob(\mathbf{C})$, denoted by A, B etc.,
- A collection of morphisms $Ar(\mathbf{C})$, denoted by f, g etc.,
- Mappings $dom, cod : Ar(\mathbf{C}) \to Ob(\mathbf{C})$ assigning to each morphism its domain and codomain respectively. We write $f : A \to B$ for a morphism f with $dom(f) = A$ and $cod(f) = B$.
- Identity arrows, i.e. for every object A there is an arrow $id_A : A \to A$,
- Composition of arrows, i.e. for every morphisms $f : A \to B$ and $g : B \to C$ there is a composite morphism $g \circ f : A \to C$.

These data must satisfy the following equations:

$$\begin{array}{ll} h \circ (g \circ f) = (h \circ g) \circ f & \text{for } f : A \to B,\ g : B \to C,\ h : C \to D, \\ f \circ id_A = f = id_B \circ f & \text{for } f : A \to B. \end{array}$$

We define, for a category $\mathbf{C}$ and two objects A, B in $Ob(\mathbf{C})$, the *Hom-set* of A and B as $Hom^{\mathbf{C}}(A, B) := \{f \in Ar(\mathbf{C}) \mid f : A \to B\}$.

For any $f : A \to B$ in $\mathbf{C}$, we say that f is an isomorphism when there exists a two-sided inverse, i.e. a $g : B \to A$ such that $g \circ f = id_A$ and $f \circ g = id_B$.

We will introduce the notion of *opposite* or *dual* category as a preliminary for duality:

Definition 1.2 (Dual Category)**.** Given a category $\mathbf{C}$, its dual category $\mathbf{C}^{\mathrm{op}}$ is given by considering the following construction:

- The objects $Ob(\mathbf{C}^{\mathrm{op}})$ are precisely $Ob(\mathbf{C})$,
- The morphisms $Ar(\mathbf{C}^{\mathrm{op}})$ are precisely $Ar(\mathbf{C})$,
- The mappings dom and cod are interchanged (source becomes target and vice versa),

- Composition is reversed, i.e. $g \circ f$ becomes $f \circ g$.

A lot of examples of categories involve certain mathematical structures and homomorphisms between these structures. One could also think of a category as a structure, and define the structure homomorphisms categorically. These are called functors:

Definition 1.3. A (covariant) functor $F : \mathbf{C} \to \mathbf{D}$ is a mapping that assigns to each object A in $Ob(\mathbf{C})$ an object $F(A)$ in $Ob(\mathbf{D})$ and to each morphism $f : A \to B$ in $Ar(\mathbf{C})$ a morphism $F(f) : F(A) \to F(B)$ in $Ar(\mathbf{D})$ such that the following hold:

1. $F(g \circ f) = F(g) \circ F(f)$,
2. $F(id_A) = id_{F(A)}$.

Besides a regular functor (the lifting of an direction preserving homomorphism), there are also direction reversing homomorphisms in category theory, called contravariant functors:

Definition 1.4. A contravariant functor $F : \mathbf{C} \to \mathbf{D}$ is a mapping that assigns to each object A in $Ob(\mathbf{C})$ an object $F(A)$ in $Ob(\mathbf{D})$ and to each morphism $f : A \to B$ in $Ar(\mathbf{C})$ a morphism $F(f) : F(B) \to F(A)$ in $Ar(\mathbf{D})$ such that the following hold:

1. $F(g \circ f) = F(f) \circ F(g)$,
2. $F(id_A) = id_{F(A)}$.

Since we have introduced the notion of dual category, we note here that a contravariant functor $F : \mathbf{C} \to \mathbf{D}$ is the same as a covariant functor $F : \mathbf{C}^{\mathrm{op}} \to \mathbf{D}$.

Similarly to the case of morphisms, there are identity functors and there exists associative composition of functors. So, it makes sense to define *isomorphisms of categories*: we say that $F : \mathbf{C} \to \mathbf{D}$ is an isomorphism of categories if there exists a functor $G : \mathbf{D} \to \mathbf{C}$ such that $G \circ F = Id_{\mathbf{C}}$ and $F \circ G = Id_{\mathbf{D}}$.

Finally, we wish to reserve a special place for *bifunctors*, functors that take two arguments. We denote such a functor by $F : \mathbf{C} \times \mathbf{D} \to \mathbf{E}$ where $\mathbf{C} \times \mathbf{D}$ is the *product category*, the category that has pairs of objects as objects and pairs of morphisms as morphisms. As a consequence, we express the functorial restrictions (or bifunctoriality) as the following two restrictions:

- Identities should be preserved, so $F(id_{(A,B)}) = id_{F(A,B)}$,
- Composition should be preserved, so $F((k, h) \circ (g, f)) = F(k, h) \circ F(g, f)$.

Next are the "arrows between functors", or *natural transformations*:

Definition 1.5. For two functors $F, G : \mathbf{C} \to \mathbf{D}$, a natural transformation $\theta : F \to G$ is a family of morphisms $\theta_A : F(A) \to G(A)$ (one for every A in $Ob(\mathbf{C})$) such that for any $f : A \to B$ the equation $\theta_B \circ F(f) = G(f) \circ \theta_A$ holds, i.e. the following diagram commutes:

$$\begin{array}{ccc} F(A) & \xrightarrow{\theta_A} & G(A) \\ {\scriptstyle F(f)}\downarrow & & \downarrow{\scriptstyle G(f)} \\ F(B) & \xrightarrow[\theta_B]{} & G(B) \end{array}$$

For functors $F, G : \mathbf{C} \to \mathbf{D}$, a natural transformation $\theta : F \to G$ is a natural isomorphism if for every A in $Ob(\mathbf{C})$ we have that $\theta_A : F(A) \to G(A)$ is an isomorphism. We write $F \cong G$ to say that F is naturally isomorphic to G.

Because many categories are not necessarily isomorphic, but rather isomorphic *up to natural isomorphism*, we need to define the concept of an *equivalence of categories*:

An equivalence of categories consists of a functor $F : \mathbf{C} \to \mathbf{D}$ and a functor $G : \mathbf{D} \to \mathbf{C}$ such that $G \circ F \cong Id_{\mathbf{C}}$ and $F \circ G \cong Id_{\mathbf{D}}$. We denote the equivalence of categories by $\mathbf{C} \cong \mathbf{D}$.

We now turn to the most important categorical concept for our purposes and perhaps the most important concept in basic category theory: the concept of *adjunction*. Adjunction covers *Galois connections* in order theory, but it also captures the behaviour of the universal quantifier versus that of the existential quantifier in first-order logic (Awodey, 2006, Section 9.5). We will proceed to give three equivalent definitions of adjunction, however we will mostly use the *Hom-set definition*:

Definition 1.6 (Hom-set Adjunction)**.** Given two categories $\mathbf{C}$ and $\mathbf{D}$, an adjunction between two functors $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{D} \to \mathbf{C}$ consists of a natural isomorphism $\varphi : Hom_{\mathbf{D}}(FA, B) \cong Hom_{\mathbf{C}}(A, GB)$.

The other definitions are given in terms of the *unit* or *co-unit* of the adjunction together with a universal mapping property:

Definition 1.7 (Unit Adjunction)**.** Given two categories $\mathbf{C}$ and $\mathbf{D}$, an adjunction between two functors $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{D} \to \mathbf{C}$ consists of a natural transformation $\eta : Id_{\mathbf{C}} \to G \circ F$ such that for any object A in $\mathbf{C}$ and B in $\mathbf{D}$ and any morphism $f : A \to G(B)$, there exists a unique $g : F(A) \to B$ such that $f = G(g) \circ \eta_A$.

Definition 1.8 (Co-Unit Adjunction)**.** Given two categories $\mathbf{C}$ and $\mathbf{D}$, an adjunction between two functors $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{D} \to \mathbf{C}$ consists of a natural transformation $\epsilon : F \circ G \to Id_{\mathbf{D}}$ such that for any object A in $\mathbf{C}$ and B in $\mathbf{D}$ and any morphism $g : F(A) \to B$ there exists a unique $f : A \to G(B)$ such that $g = \epsilon_B \circ F(f)$.

See (Awodey, 2006, Chapter 9) for a proof that these definitions are in fact equivalent.

The concept of a *monad*, also called a *triple* or *standard construction*, might be seen as the generalization of the concept of a *closure operator* in order theory:

Definition 1.9. A monad on a category $\mathbf{C}$ is a triple (T, η, μ) where $T : \mathbf{C} \to \mathbf{C}$ is an endofunctor and $\eta : Id_C \to T$ and $\mu : T \circ T \to T$ are natural transformations such that the following diagrams commute for every object A:

$$\begin{array}{ccc} T(T(T(A))) & \xrightarrow{T(\mu_A)} & T(T(A)) \\ \downarrow{\scriptstyle \mu_{T(A)}} & & \downarrow{\scriptstyle \mu_A} \\ T(T(A)) & \xrightarrow[\mu_A]{} & T(A) \end{array}$$

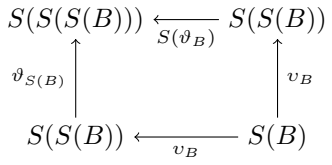
$$\begin{array}{ccc} T(A) & \xrightarrow{T(\eta_A)} & T(T(A)) \\ & {\scriptstyle id_{T(A)}} \searrow \quad \swarrow {\scriptstyle \mu_A} & \\ & T(A) & \end{array} \qquad \begin{array}{ccc} T(A) & \xrightarrow{\eta_{T(A)}} & T(T(A)) \\ & {\scriptstyle id_{T(A)}} \searrow \quad \swarrow {\scriptstyle \mu_A} & \\ & T(A) & \end{array}$$

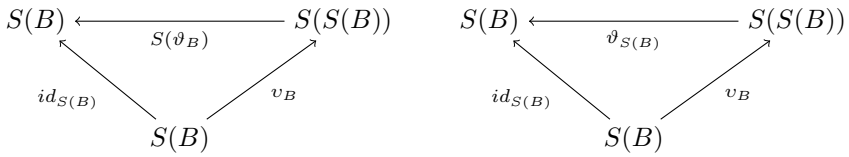
The striking thing about mondads is that two adjoint functors always define a monad! Note that in the following proposition we use the notation $G\epsilon F$: this is the natural transformation defined for A in $Ob(\mathbf{C})$ as $G(\epsilon_{F(A)})$.

Proposition 1.1. *Given two functors $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{D} \to \mathbf{C}$ that are adjoint, the triple $(G \circ F, \eta, G\epsilon F)$ where η is the unit of the adjunction and ϵ is the co-unit of the adjunction is a monad.*

Next to considering closure operators and monads as their generalization, we want to note that the dual notion, that of an *interior operator*, is generalized by the dual notion of a *comonad*:

Definition 1.10. A comonad on a category $\mathbf{C}$ is a triple (S, ϑ, υ) where $T : \mathbf{C} \to \mathbf{C}$ is an endofunctor and $\vartheta : S \to Id_C$ and $\upsilon : T \to T \circ T$ are natural transformations such that the following diagrams commute for every object B:





We then get that the reversed composition $F \circ G$ for adjoint functors F and G gives rise to a comonad:

Proposition 1.2. *Given two functors $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{D} \to \mathbf{C}$ that are adjoint, the triple $(F \circ G, \epsilon, F\eta G)$, where η is the unit of the adjunction and ϵ is the co-unit of the adjunction, is a comonad.*

In the next section, we will look at *categories with additional structure*.

1.2 Monoidal and Closed Categories

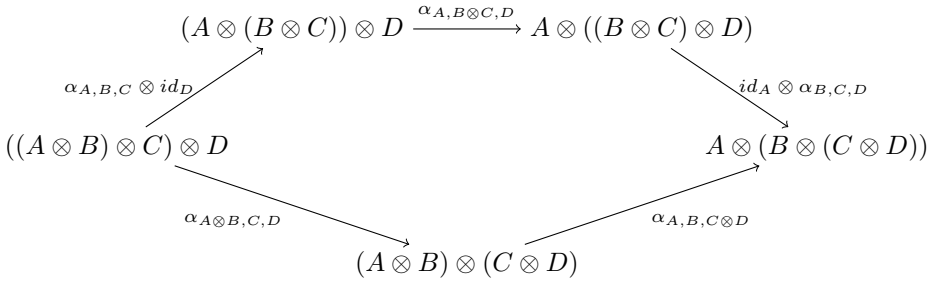
A standard concept of a category with extra structure is that of a *monoidal category*: a category that exhibits the structure of a monoid. However, for our purposes we need to simplify this definition as we want to consider categories with extra structure but without associativity or units. To this end we define *tensor categories*[1] as categories that have a "tensor" without extra coherence axioms whatsoever:

Definition 1.11. A tensor category is a category $\mathbf{C}$ equipped with a bifunctor $\otimes : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$.

[1]The term tensor category seems to be used to refer to a monoidal category. Despite the confusion, we found it the most logical way to refer to a category that has a "tensor".

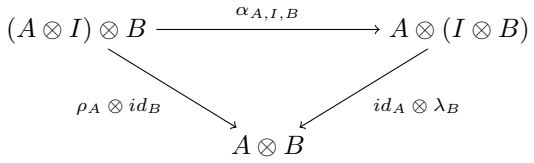
We can then go on to *add* associativity or units, giving us the following definitions:

Definition 1.12. An associative tensor category (or non-unitary monoidal category) is a tensor category $(\mathbf{C}, \otimes)$ equipped with an isomorphism natural in A, B, C specified by $\alpha_{A,B,C} : (A \otimes B) \otimes C \to A \otimes (B \otimes C)$ where the following diagram commutes:



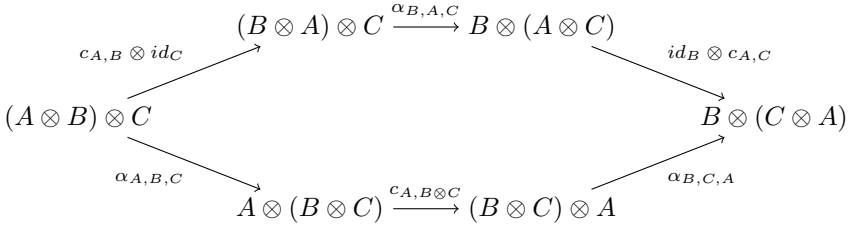
Definition 1.13. A unitary tensor category (or non-associative monoidal category) is a tensor category $(\mathbf{C}, \otimes)$ with a distinguished unit object I and natural isomorphisms specified by $\lambda_A : I \otimes A \to A$ and $\rho_A : A \otimes I \to A$.

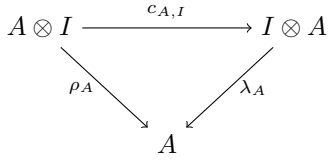
Definition 1.14. A monoidal category (or associative unitary tensor category) is an associative unitary tensor category $(\mathbf{C}, \otimes, \alpha, I, \lambda, \rho)$ where the following diagram commutes:



We have now sketched one dimension in our landscape of categories: adding a tensor and then picking a unit object or associativity as extra features to ultimately obtain a monoidal category. There are also monoidal categories where the tensor behaves as a commutative product:

Definition 1.15. A symmetric monoidal category is a monoidal category $(\mathbf{C}, \otimes, \alpha, I, \lambda, \rho)$ equipped with natural isomorphisms specified by $c_{A,B} : A \otimes B \to B \otimes A$ such that $c_{B,A} \circ c_{A,B} = id_{A\otimes B}$ and such that the following diagrams commute:





We now want to define the *closure* of our categories. This is achieved by adding bifunctors that are left/right adjoint to the tensor:

Definition 1.16. A left closed tensor category is a tensor category $(\mathbf{C}, \otimes)$ equipped with a bifunctor $\Rightarrow\colon \mathbf{C}^{\mathrm{op}} \times \mathbf{C} \to \mathbf{C}$ (i.e. contravariant in its first argument, covariant in its second argument) together with natural isomorphism specified by $\beta_{A,B,C} : Hom^{\mathbf{C}}(A \otimes B, C) \to Hom^{\mathbf{C}}(B, A \Rightarrow C)$.

Definition 1.17. A right closed tensor category is tensor category $(\mathbf{C}, \otimes)$ equipped with a bifunctor $\Leftarrow\colon \mathbf{C} \times \mathbf{C}^{\mathrm{op}} \to \mathbf{C}$ together with a natural isomorphism specified by $\gamma_{A,B,C} : Hom^{\mathbf{C}}(A \otimes B, C) \to Hom^{\mathbf{C}}(A, C \Leftarrow B)$.

Definition 1.18. A bi-closed tensor category is a tensor category $(\mathbf{C}, \otimes)$ that is both left and right closed.

Left closed, right closed, and bi-closed associative tensor/monoidal categories are defined analogously. Note, however, that a symmetric monoidal category is by definition bi-closed when it is either left or right closed. For example, suppose we have a left closed symmetric monoidal category $(\mathbf{C}, \otimes, \alpha, I, \lambda, \rho, c, \Rightarrow, \beta)$. Define $B \Leftarrow A := A \Rightarrow B$ and $\gamma := f \mapsto \beta(f \circ c)$. It is easy to show that this defines a right closed structure on $\mathbf{C}$.

We have already noted that adjoints give rise to monads and comonads. However, there are another two monads that arise in bi-closed categories:

Let $(\mathbf{C}, \otimes, \Rightarrow, \beta, \Leftarrow, \gamma)$ be a bi-closed category. Define, for any object D in $Ob(\mathbf{C})$ the functor $D \Leftarrow (\_ \Rightarrow D)$ (resp. $(D \Leftarrow \_) \Rightarrow D$) that sends objects A to $D \Leftarrow (A \Rightarrow D)$ ($(D \Leftarrow A) \Rightarrow D$) and sends maps $f : A \to B$ to $id_D \Leftarrow (f \Rightarrow id_D)$ ($(id_D \Leftarrow f) \Rightarrow id_D$).

Now define the natural transformations $\eta : Id_{\mathbf{C}} \to D \Leftarrow (\_ \Rightarrow D)$ by $\eta_A := \gamma(\beta^{-1}(id_{A \Rightarrow D}))$ and $\mu : D \Leftarrow ((D \Leftarrow (\_ \Rightarrow D)) \Rightarrow D)$ by $\mu_A := id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow (id_A \Rightarrow D)})))$ (and similarly for the functor $(D \Leftarrow \_) \Rightarrow D$.

Proposition 1.3. *The triple* $(D \Leftarrow (\_ \Rightarrow D), \eta, \mu)$ *defines a monad on* $\mathbf{C}$.

Proof. For the square diagram we have

$$
\begin{aligned}
&(id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)})))) \circ (id_D \Leftarrow ((id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)})))) \Rightarrow id_D)) \\
&= id_D \Leftarrow ((id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)})))) \Rightarrow id_D \circ \beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)}))) \\
&= id_D \Leftarrow ((id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)})))) \Rightarrow id_D \circ \beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)})) \circ id_{A \Rightarrow D}) \\
&= id_D \Leftarrow (\beta(id_D \circ \gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)}) \circ ((id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)})))) \oplus id_{A \Rightarrow D}))) \\
&= id_D \Leftarrow (\beta(\gamma^{-1}((id_D \Leftarrow (id_{A \Rightarrow D})) \circ id_{D \Leftarrow (A \Rightarrow D)} \circ (id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)}))))))) \\
&= id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)} \circ id_{D \Leftarrow (A \Rightarrow D)} \circ (id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)}))))))) \\
&= id_D \Leftarrow (\beta(\gamma^{-1}((id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)})))) \circ id_{D \Leftarrow ((D \Leftarrow (A \Rightarrow D)) \Rightarrow D)} \circ id_{D \Leftarrow ((D \Leftarrow (A \Rightarrow D)) \Rightarrow D)}))) \\
&= id_D \Leftarrow (\beta(id_D \circ \gamma^{-1}(id_{D \Leftarrow ((D \Leftarrow (A \Rightarrow D)) \Rightarrow D)}) \circ (id_{D \Leftarrow ((D \Leftarrow (A \Rightarrow D)) \Rightarrow D)} \otimes \beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)}))))) \\
&= id_D \Leftarrow ((id_{D \Leftarrow ((D \Leftarrow (A \Rightarrow D)) \Rightarrow D)} \Rightarrow id_D) \circ \beta(\gamma^{-1}(id_{D \Leftarrow ((D \Leftarrow (A \Rightarrow D)) \Rightarrow D)})) \circ \beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)}))) \\
&= id_D \Leftarrow (id_{(D \Leftarrow ((D \Leftarrow (A \Rightarrow D)) \Rightarrow D)) \Rightarrow D} \circ \beta(\gamma^{-1}(id_{D \Leftarrow ((D \Leftarrow (A \Rightarrow D)) \Rightarrow D)})) \circ \beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)}))) \\
&= id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow ((D \Leftarrow (A \Rightarrow D)) \Rightarrow D)})) \circ \beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)}))) \\
&= (id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)})))) \circ (id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow ((D \Leftarrow (A \Rightarrow D)) \Rightarrow D)}))))
\end{aligned}
$$

For the first triangle diagram, we have

$$
\begin{array}{ll}
(id_D \Leftarrow (\beta(\gamma^{-1}(id_{D\Leftarrow(A\Rightarrow D)})))) \circ (id_D \Leftarrow ((\gamma(\beta^{-1}(id_{A\Rightarrow D}))) \Rightarrow id_D)) & \\
= id_D \Leftarrow (((\gamma(\beta^{-1}(id_{A\Rightarrow D}))) \Rightarrow id_D) \circ \beta(\gamma^{-1}(id_{D\Leftarrow(A\Rightarrow D)}))) & \text{bifunctoriality of } \Leftarrow \\
= id_D \Leftarrow (((\gamma(\beta^{-1}(id_{A\Rightarrow D}))) \Rightarrow id_D) \circ \beta(\gamma^{-1}(id_{D\Leftarrow(A\Rightarrow D)})) \circ id_{A\Rightarrow D}) & \text{identity axiom} \\
= id_D \Leftarrow (\beta(id_D \circ \gamma^{-1}(id_{D\Leftarrow(A\Rightarrow D)}) \circ (\gamma(\beta^{-1}(id_{A\Rightarrow D})) \oplus id_{A\Rightarrow D}))) & \text{naturality of } \beta \\
= id_D \Leftarrow (\beta(\gamma^{-1}((id_D \Leftarrow id_{A\Rightarrow D}) \circ id_{D\Leftarrow(A\Rightarrow D)} \circ \gamma(\beta^{-1}(id_{A\Rightarrow D}))))) & \text{naturality of } \gamma^{-1} \\
= id_D \Leftarrow (\beta(\gamma^{-1}(id_{D\Leftarrow(A\Rightarrow D)} \circ id_{D\Leftarrow(A\Rightarrow D)} \circ \gamma(\beta^{-1}(id_{A\Rightarrow D}))))) & \text{bifunctoriality of } \Leftarrow \\
= id_D \Leftarrow (\beta(\gamma^{-1}(\gamma(\beta^{-1} id_{A\Rightarrow D})))) & \text{identity axiom twice} \\
= id_D \Leftarrow id_{A\Rightarrow D} & \text{iso property of } \beta \text{ and } \gamma \\
= id_{D\Leftarrow(A\Rightarrow D)} & \text{bifunctoriality of } \Leftarrow
\end{array}
$$

Finally, for the second triangle diagram, we have

$$
\begin{array}{ll}
(id_D \Leftarrow (\beta(\gamma^{-1}(id_{D\Leftarrow(A\Rightarrow D)})))) \circ \gamma(\beta^{-1}(id_{(D\Leftarrow(A\Rightarrow D))\Rightarrow D})) & \\
= (id_D \Leftarrow (\beta(\gamma^{-1}(id_{D\Leftarrow(A\Rightarrow D)})))) \circ \gamma(\beta^{-1}(id_{(D\Leftarrow(A\Rightarrow D))\Rightarrow D})) \circ id_{D\Leftarrow(A\Rightarrow D)} & \text{identity axiom} \\
= \gamma(id_D \circ \beta^{-1}(id_{(D\Leftarrow(A\Rightarrow D))\Rightarrow D})) \circ (id_{D\Leftarrow(A\Rightarrow D)} \otimes \beta(\gamma^{-1}(id_{D\Leftarrow(A\Rightarrow D)}))) & \text{naturality of } \gamma \\
= \gamma(\beta^{-1}((id_{D\Leftarrow(A\Rightarrow D)} \Rightarrow id_D) \circ id_{(D\Leftarrow(A\Rightarrow D))\Rightarrow D} \circ \beta(\gamma^{-1}(id_{D\Leftarrow(A\Rightarrow D)})))) & \text{naturality of } \beta^{-1} \\
= \gamma(\beta^{-1}(id_{(D\Leftarrow(A\Rightarrow D))\Rightarrow D} \circ id_{(D\Leftarrow(A\Rightarrow D))\Rightarrow D} \circ \beta(\gamma^{-1}(id_{D\Leftarrow(A\Rightarrow D)})))) & \text{bifunctoriality of } \Rightarrow \\
= \gamma(\beta^{-1}(\beta(\gamma^{-1} id_{D\Leftarrow(A\Rightarrow D)}))) & \text{identity axiom twice} \\
= id_{D\Leftarrow(A\Rightarrow D)} & \text{iso property of } \beta \text{ and } \gamma
\end{array}
$$

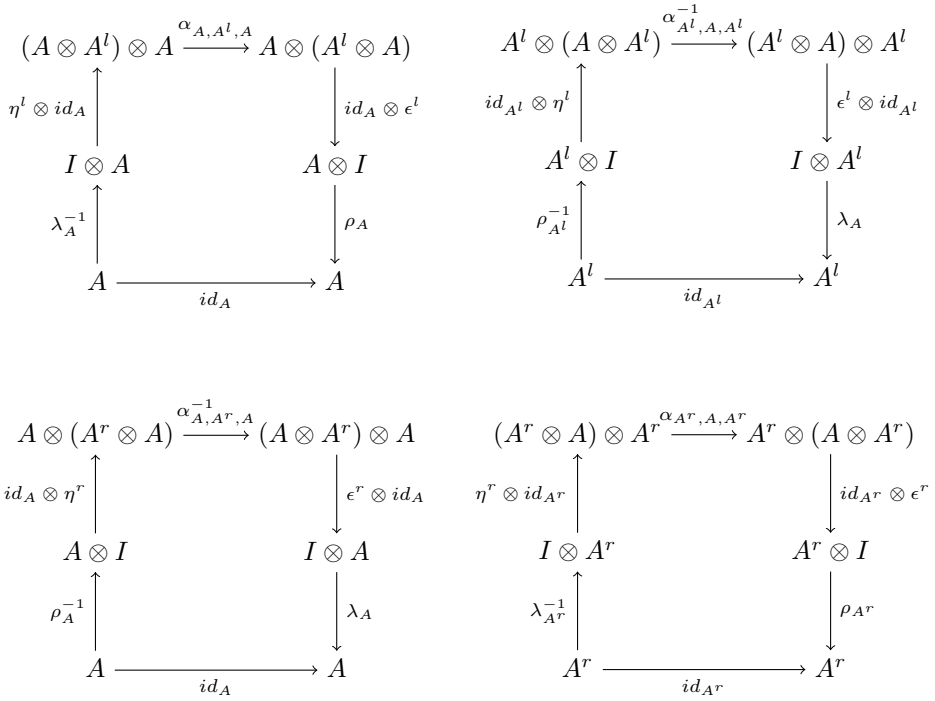
□

Next to the closed categories we have already considered, there are special cases of a monoidal closed category, where each object has a left/right *dual object*. When the underlying category is non-symmetric, such a category is called *autonomous* or *rigid* but in the presence of symmetry these categories are called *compact closed*:

Definition 1.19. An autonomous category is a monoidal category $(\mathbf{C}, \otimes, \alpha, I, \lambda, \rho)$ such that for every object A in $Ob(\mathbf{C})$ there exist objects A^l and A^r (called left and right adjoints) and for every A there exist morphisms

$$\eta^l : I \to A \otimes A^l \quad \epsilon^l : A^l \otimes A \to I \quad \eta^r : I \to A^r \otimes A \quad \epsilon^r : A \otimes A^r \to I$$

Such that the following diagrams commute:



Autonomous categories as defined above should actually be called bi-autonomous categories as they contain both left and right dual objects. It is of course obvious how left/right autonomous categories should be defined. In the case that the monoidal category is also symmetric, the left and right dual objects collapse into one dual object (up to isomorphism) and we speak of a compact closed category.

An interesting property of autonomous and compact closed categories is that they form bi-closed categories by setting $A \Rightarrow B := A^l \otimes B$ and $B \Leftarrow A := B \otimes A^r$, giving rise to the following two propositions:

Proposition 1.4. *Every bi-autonomous category is a bi-closed tensor category.*

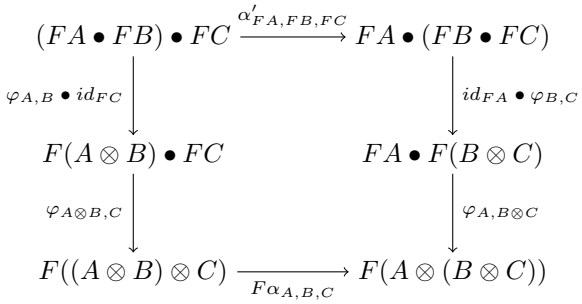
Proposition 1.5. *Every compact closed category is a bi-closed symmetric monoidal category.*

In the following section, we consider *functors with structure* for use between the various kinds of categories we have considered so far.

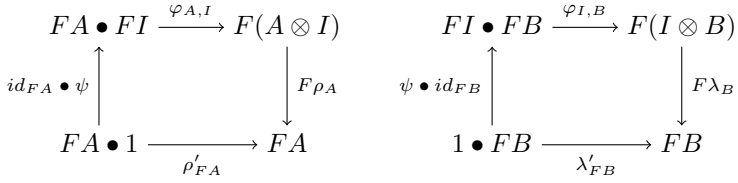
1.3 Monoidal and Closed Functors

Definition 1.20. For $(\mathbf{C}, \otimes)$ and $(\mathbf{D}, \bullet)$ tensor categories, a tensor functor is a functor $F : \mathbf{C} \to \mathbf{D}$ such that there exists a natural transformation specified by $\varphi_{A,B} : FA \bullet FB \to F(A \otimes B)$.

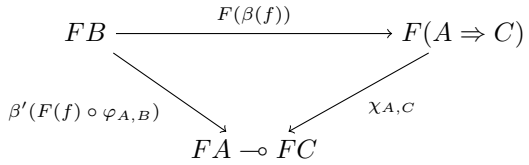
Definition 1.21. For $(\mathbf{C}, \otimes, \alpha)$ and $(\mathbf{D}, \bullet, \alpha')$, an associative tensor functor is a tensor functor where the natural transformation specified by $\varphi_{A,B} : FA \bullet FB \to F(A \otimes B)$ satisfies the commuting of the following diagram:



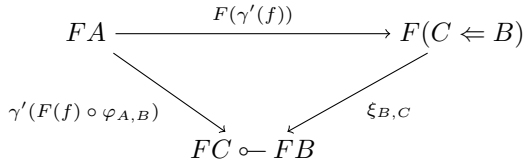
Definition 1.22. For $(\mathbf{C}, \otimes, \alpha, I, \lambda, \rho)$ and $(\mathbf{D}, \bullet, \alpha', 1, \lambda', \rho')$ monoidal categories, a monoidal functor is an associative tensor functor $F : \mathbf{C} \to \mathbf{D}$ with associated natural transformation specified by $\varphi_{A,B} : FA \bullet FB \to F(A \otimes B)$ and there exists a morphism $\psi : 1 \to FI$ such that additionally the following diagrams commute:



Definition 1.23. For $(\mathbf{C}, \otimes, \Rightarrow, \beta)$ and $(\mathbf{D}, \bullet, \multimap, \beta')$ left closed tensor categories, a left closed tensor functor is a tensor functor F with associated natural transformation specified by $\varphi_{A,B} : FA \bullet FB \to F(A \otimes B)$ such that there additionally exists a natural transformation specified by $\chi_{A,B} : F(A \Rightarrow B) \to FA \multimap FB$ such that for every $f : A \otimes B \to C$ in $\mathbf{C}$, we have that the following diagram commutes:



Definition 1.24. For $(\mathbf{C}, \otimes, \Leftarrow, \gamma)$ and $(\mathbf{D}, \bullet, \circ\!\!-, \gamma')$ right closed tensor categories, a right closed tensor functor is a tensor functor F with associated natural transformation specified by $\varphi_{A,B} : FA \bullet FB \to F(A \otimes B)$ such that there additionally exists a natural transformation specified by $\xi_{A,B} : F(B \Leftarrow A) \to FB \circ\!\!- FA$ such that for every $f : A \otimes B \to C$ in $\mathbf{C}$, we have that the following diagram commutes:



Definition 1.25. For $(\mathbf{C}, \otimes, \Rightarrow, \Leftarrow, \beta, \gamma)$ and $(\mathbf{D}, \bullet, \multimap, \circ\!\!-, \beta', \gamma')$ bi-closed tensor categories, a bi-closed tensor functor is a left and right closed tensor functor F.

In the final section of this chapter, we review the symmetry exhibited between left and right closed categories.

1.4 Symmetry

There is an obvious symmetry between a left and right closed category, whether it be a tensor category or a monoidal one. The symmetry involves swapping the $\Rightarrow$ and $\Leftarrow$ functors. So, let $(\mathbf{C}, \otimes, \Rightarrow, \beta)$ be a left closed tensor category and let $(\mathbf{D}, \otimes, \Leftarrow, \gamma)$ be a right closed tensor category such that $\mathbf{C}$ and $\mathbf{D}$ have the same objects. Define the following functor $S : \mathbf{C} \to \mathbf{D}$:

$$\begin{array}{lcl} S(A) & = & A \\ S(A \otimes B) & = & S(B) \otimes S(A) \\ S(A \Rightarrow B) & = & S(B) \Leftarrow S(A) \\ S(id_A) & = & id_{S(A)} \\ S(g \circ f) & = & S(g) \circ S(f) \\ S(f \otimes g) & = & S(g) \otimes S(f) \\ S(f \Rightarrow g) & = & S(g) \Leftarrow S(f) \\ S(\beta(f)) & = & \gamma(S(f)) \end{array}$$

To show that this is in fact an isomorphism of categories, define the following functor $S' : \mathbf{D} \to \mathbf{C}$:

$$\begin{array}{lcl} S'(A) & = & A \\ S'(A \otimes B) & = & S'(B) \otimes S'(A) \\ S'(B \Leftarrow A) & = & S'(A) \Rightarrow S'(B) \\ S'(id_A) & = & id_{S'(A)} \\ S'(g \circ f) & = & S'(g) \circ S'(f) \\ S'(f \otimes g) & = & S'(g) \otimes S'(f) \\ S'(g \Leftarrow f) & = & S'(f) \Rightarrow S'(g) \\ S'(\gamma(f)) & = & \beta(S'(f)) \end{array}$$

It is an easy exercise to check that $S \circ S' = Id_{\mathbf{D}}$ and $S' \circ S = Id_{\mathbf{C}}$.

It is immediate that we can extend the symmetry functors S, S' between left and right closed categories to an endofunctor on bi-closed categories.

Chapter 2

Graphical Languages

In this chapter, we will look at *graphical languages*: languages that we can use to reason graphically about morphism equality in categories with a certain structure. We will look at the connection between proof nets and graphical languages, and define a graphical language for closed tensor categories. We will then devote attention to proving *coherence* (i.e. soundness and completeness) for this graphical language by means of a *freeness* theorem. We conclude with some critical suggestions about the relation between graphical languages with and without associativity.

2.1 Graphical Languages: an Introduction

Reasoning about morphism equality in categories usually proceeds by drawing commutative diagrams or explicitly writing down chains of equations. The problem with the latter form is immediate: it is very tedious to write down and to check whether each step is correct. One problem with commutative diagrams is that they only reflect the *typing* of morphisms, i.e. their domain and codomain. Nothing is said about the structure of the morphisms. So, we may very well ask ourselves *how can we represent (the structure of) morphisms and their equations graphically?*

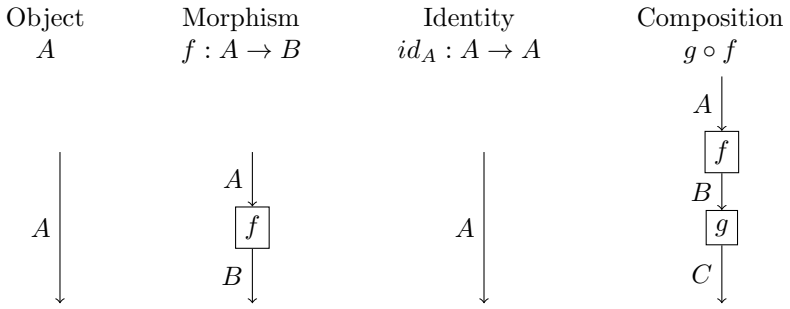
The short answer is: just define a graphical language for the category you like! The sad news is that one will want to show *coherence*, i.e. soundness and completeness of the graphical language with respect to the category it is stated for. This is usually quite hard, as it requires a considerable amount of topology.

Nevertheless, graphical languages have already been developed for monoidal categories (Joyal and Street, 1991a) and various kinds of monoidal categories with additional structure (Joyal and Street, 1991b), together with coherence proofs. For monoidal closed categories, there is a graphical language (Baez and Stay, 2011), but coherence has not been proven for it (John Baez, personal communication). A nice survey of the various graphical languages is due to Selinger (Selinger, 2011).

We will review the existing graphical languages for monoidal categories and their extension to monoidal closed categories. Then will ask ourselves what happens when we drop associativity and the units and get to the point where we want a graphical language for closed tensor categories. We provide an answer in terms of *proof nets* and show coherence for a graphical language of proof nets for bi-closed tensor categories.

2.2 Graphical Languages for Monoidal Categories

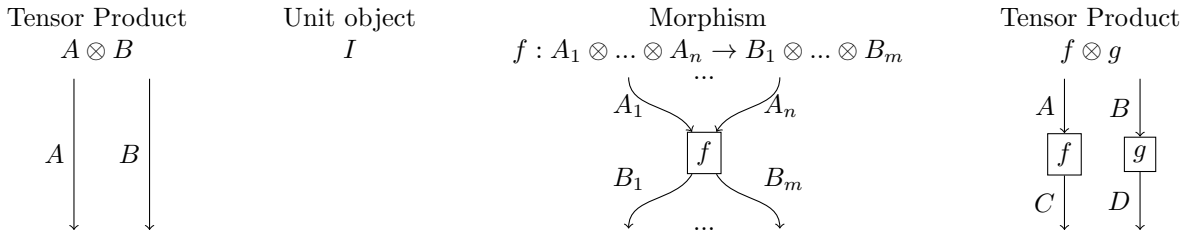
We start out by representing objects in a category as labelled wires and morphisms as *boxes* (with their name on it) that have an incoming and an outgoing wire (resp. the domain and codomain). The identity morphism is then visualized as an ongoing wire without a box in between. Composition is defined as juxtaposing two diagrams. All this is summarized in the following figure:



The categorical axioms are then automatically present: the identity axiom is fulfilled as we may choose how long we make the wires, associativity of composition is fullfilled as the juxtaposition of diagrams does not distinguish between what was glued together first and what was glued together second.

2.2.1 Going Monoidal

The next step is to consider the tensor product of monoidal categories: we can draw objects $A \otimes B$ simply by drawing them next to each other. The unit object is represent by the empty wire, and the tensor product of morphisms is represented by drawing the morphisms next to each other. This is all in the next figure:



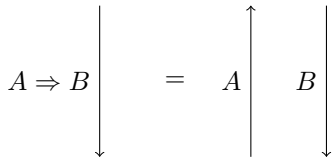
Associativity of the tensor is now automatically satisfied. Also, the unit laws are immediately satisfied as we don't bother to draw the units. In fact, we get the following coherence theorem for this graphical language:

Theorem 2.1 ((Selinger, 2011),Thm. 1.3,(Joyal and Street, 1991a), Thm. 1.2)**.** *A well-formed equation between morphisms in a monoidal category follows from the axioms if and only if it holds, up to planar isotopy, in the corresponding graphical language.*

For a detailed exposition of the proof, see Joyal and Street's original paper (Joyal and Street, 1991a). For a somewhat clearer but less detailed exposition, see the survey paper of Selinger (2011).

2.2.2 Closing the Category

We will now consider the graphical language for monoidal closed categories proposed by Baez and Stay (2011). To realize the extension of graphical languages for monoidal categories to those that have internal homs, one needs a graphical representation of objects $A \Rightarrow B$ and $B \Leftarrow A$. Intuitively, one might want to draw the object $A \Rightarrow B$ as an arrow going up next to an arrow going down, as in



However, as Baez and Stay note, in the general case where the monoidal closed category is not compact, arrows pointing up are not allowed. To resolve this issue but still maintain the intuitive idea of an arrow going upwards, one draws a *clasp* connecting the upwards pointed arrow to the downward pointed arrow. So for bi-closed monoidal categories, we extend the graphical language for monoidal categories with the constructs of Figure 2.1.

Now we only need to describe the effect of β and γ on morphisms graphically. The effect of β and β^{-1} is shown in Figure 2.2, the graphical representation of the action of γ and γ^{-1} is completely symmetrical.

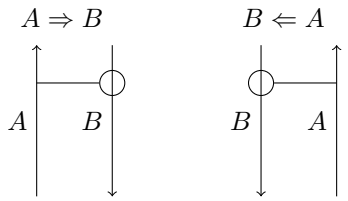


Figure 2.1: Language Constructs for Closedness in Monoidal Categories

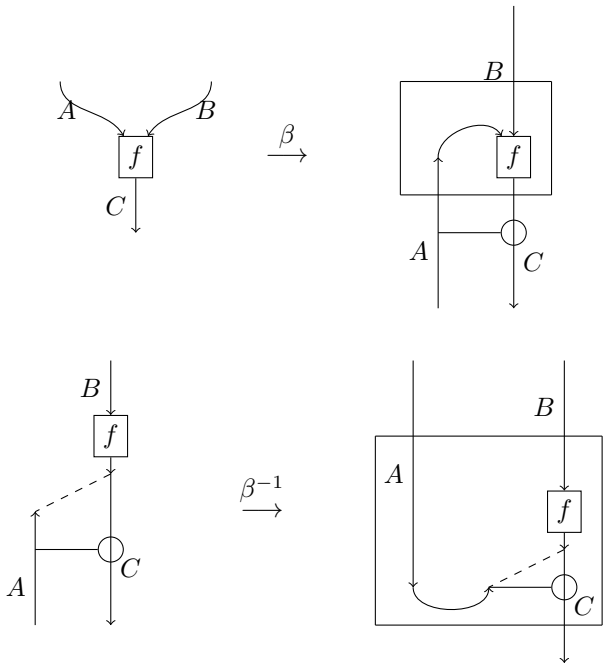


Figure 2.2: Currying and Uncurrying in Monoidal Closed Categories

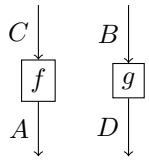
Note that we have actually *bent around* arrows in order to keep morphisms going down. As in the general (the non-compact) case this is not allowed, we draw a box around the "illegal" constructs.

2.2.3 A Problem With the Clasp Language

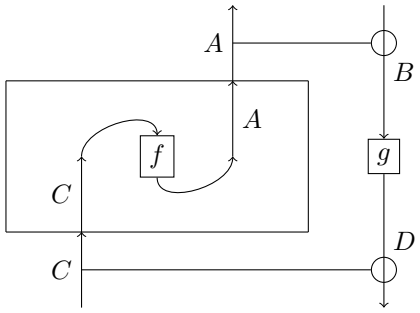
Unfortunately, there is no coherence proof for the clasp language of Baez and Stay. At least the original authors have not tried to prove coherence for it (John Baez, pers. comm.). So in this subsection we will discuss some issues regarding the clasp language.

Firstly, we may well ask ourselves how to represent the effect of the $\Rightarrow$ functor on two maps.

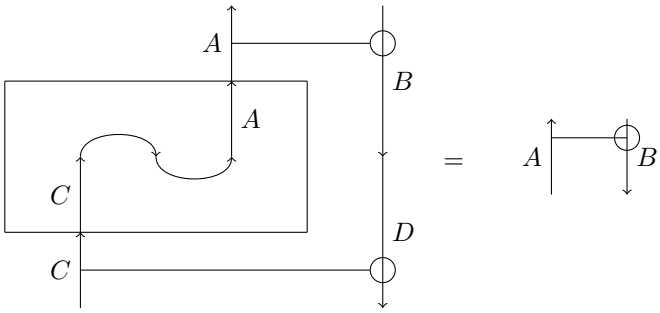
Given morphisms $f : C \to A$ and $g : B \to D$ represented by



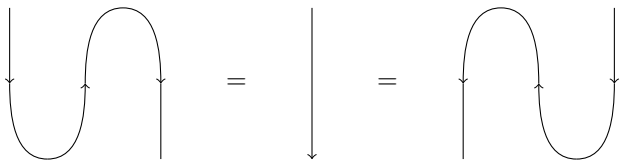
it makes sense to define $f \Rightarrow g$ as follows:



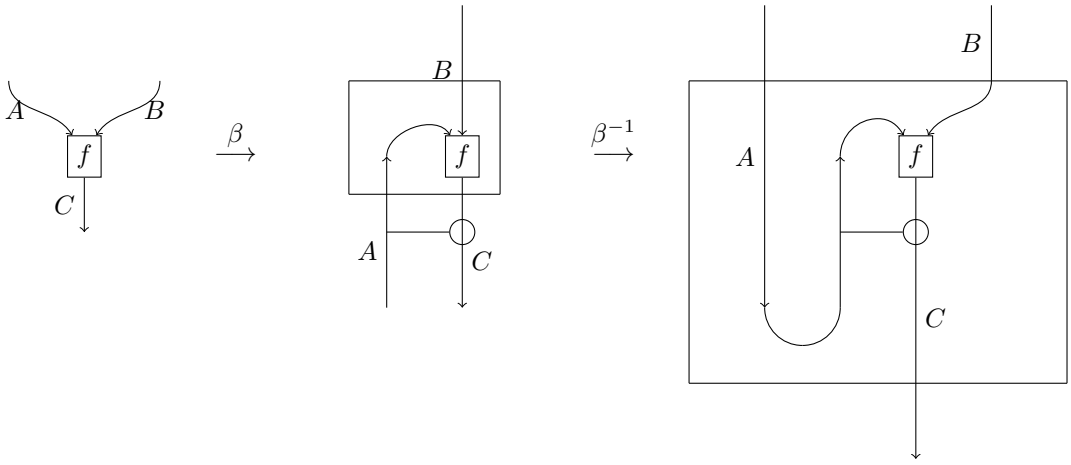
Adopting this graphical representation, consider the fact that $id_A \Rightarrow id_B = id_{A \Rightarrow B}$ should be satisfied. This would mean that we should be able to derive



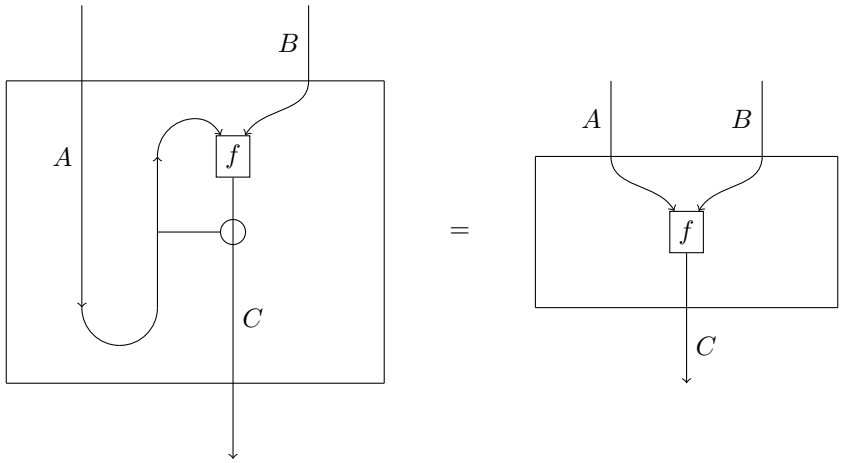
But the only way to do this is to require the graphical yanking equations



Now consider the fact that β and γ should be natural isomorphisms: this means that the effect of consecutively currying and uncurrying some morphism should return something that is derivationally equal to the original morphims. But the currying and uncurrying of a morphism $f : A \otimes B \to C$ looks as follows:

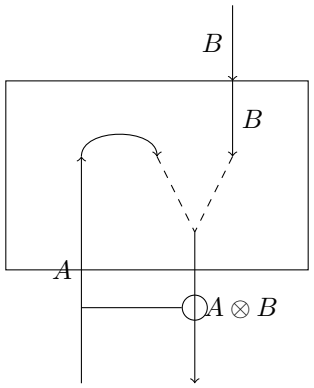


And again, the only way to make β an isomorphism is to require graphical yanking, in which case we would get



Fortunately, one can actually allow yanking when one allows it only inside a box. This means effectively that inside a box the situation is compact, while outside a box it is purely monoidal. It should however be noted that when boxes are drawn every time there is an arrow bent around, allowing yanking only within a box basically amounts to global yanking. So it might be interesting to see a definitive proof of why this does not pose a problem.

Finally, there is a problem with the concept of clasps. Consider the co-evaluation morphism $\mathit{co\text{-}ev} : B \to A \Rightarrow (A \otimes B)$. Given that $A \Rightarrow (A \otimes B)$ is not the same as $(A \Rightarrow A) \otimes B$, the only means to represent co-evaluation would be to explicitly *merge* B and C, as in the following picture:



Clearly, there are some problems with the clasp language as we have considered it. Either certain constructs should be made explicit to be able to have a coherent graphical language based on clasps, or it should not be attempted to recover a coherent language from the clasp diagrams.

In the next section we will develop a graphical language for closed tensor categories and prove coherence for it.

2.3 Graphical Languages for Closed Tensor Categories

In this section, we develop a graphical language for closed tensor categories. That is, categories that do have a tensor and left and right internal homs but for which the tensor lacks associativity. We will see in the next chapter that these categories correspond to non-associative Lambek Calculi. We will start out with a brief review of proof nets for the latter calculi and then go on to define *proof net categories* and state a freeness theorem about them, providing coherence for our graphical language.

2.3.1 Proof Nets versus Graphical Languages

Proof nets were originally devised by Girard (1987) as a system that enables one to visualize proofs in a succinct way: proofs that are syntactically different but are more or less the same are associated with the same proof net. After its introduction, proof nets for different types of logics have been extensively studied and in particular proof net systems have been developed for the different incarnations of Lambek's syntactic calculus and its extensions (Roorda, 1991; Moot, 2002; Moot and Puite, 2002; Moortgat and Moot, 2012). These proof nets are developed more or less along the same lines: firstly, one should define the notion of a *proof structure*, a graph made up from certain *links*, both tensor and par links. Next is the definition of *correctness criteria*: criteria that may be used to establish which proof structures will correspond to sequent proofs and which will not correspond to a sequent proof. The proof structures satisfying the correctness criteria are called *proof nets*. The original correctness criterium of Girard for proof structures of linear logic was the notion of a *long-trip* criterion, stating that a proof structure is a proof net if one is able to produce a certain traversal of the proof structure. For the multiplicative fragment of linear logic, the correctness criteria are stated *statically*: one should consider all *switchings* of par links, and for each possible switching, the resulting structure should be *acyclic* and *connected*.

The key difference between proof nets and graphical languages is that the former are based on sequent systems, and as such facilitate multiple "inputs" whereas morphisms in a category only have one input (i.e. the domain). So even though we take inspiration from proof nets in the development of our graphical language (we will define links, proof structures and correctness criteria) the graphical language is still very different from the proof net representation. Another difference is that proof nets can be used to give a graphical proof of cut elimination, as is done by Moot (2002, Section 4.4) for the case of multiplicative linear logic.

We will develop our proof net language taking inspiration from the work of Blute et al. (1996); Cockett and Seely (1997a) and we will prove coherence according to the method outlined in Selinger (2011).

2.3.2 Signatures, Interpretations and Free Categories

Definition 2.1. A bi-closed tensor signature $\Sigma = (\Sigma_0, \Sigma_1, dom, cod)$ consists of:

- a set Σ_0 of object variables,
- a set Σ_1 of morphism variables,
- two maps $dom, cod : \Sigma_1 \to CT(\Sigma_0)$.

where $CT(\Sigma_0)$ is the free $(\otimes, \Rightarrow, \Leftarrow)$-algebra generated by Σ_0.

Definition 2.2. Given a bi-closed tensor signature Σ and a closed tensor category $\mathbf{C}$, an interpretation $i : \Sigma \to \mathbf{C}$ consists of:

- an object map $i_0 : \Sigma_0 \to Ob(\mathbf{C})$ such that

$$\begin{aligned} i_0(A \otimes B) &= i_0(A) \otimes i_0(B) \\ i_0(A \Rightarrow B) &= i_0(A) \Rightarrow i_0(B) \\ i_0(B \Leftarrow A) &= i_0(B) \Leftarrow i_0(A), \end{aligned}$$

- for every $f \in \Sigma_1$ a morphism $i_1(f) : i_0(dom(f)) \to i_0(cod(f))$.

Definition 2.3. A bi-closed tensor category $\mathbf{C}$ is a free bi-closed tensor category over a bi-closed tensor signature Σ if there is an interpretation $i : \Sigma \to \mathbf{C}$ such that for any bi-closed tensor category $\mathbf{D}$ and bi-closed tensor interpretation $j : \Sigma \to \mathbf{D}$, there is a unique bi-closed tensor functor $F : \mathbf{C} \to \mathbf{D}$ such that $j = F \circ i$.

We will develop a graphical language as a *proof net category*. Showing that for any bi-closed tensor category, the associated proof net category is the free one means that all equations in the category hold if and only if they hold in the graphical language and as such, coherence will have been proven.

2.3.3 Sequent Calculus Categorified

The coherence proof we aim to prove is based on a translation of proof nets (to be defined subsequently) into sequent proofs, which in turn are translated into categorical morphisms. Thus, we must define the sequent calculus (and an equivalence relation on sequent proofs that we will use). The following definitions are borrowed from (Bastenhof, 2013):

Definition 2.4 (Formulae)**.** Given a set of atomic formulae At, the set of formulae is defined as follows:

$$A, B := p \mid A \otimes B \mid A\backslash B \mid B/A \text{ for } p \in At.$$

Next to defining formulae are structures, which will be used on the left-hand side of the turnstile in sequent proofs:

Definition 2.5 (Structures)**.** Structures are defined over formulas using a binary merger:

$$\Gamma, \Delta := A \mid (\Gamma \bullet \Delta)$$

To ease our reading of the sequent calculus rules, we define contexts, which are structures with a unique hole in them, where we in turn can place structures in:

Definition 2.6 (Contexts)**.** A context is a structure with a unique occurrence of a hole []:

$$\Gamma[], \Delta[] := [] \mid (\Gamma[] \bullet \Delta) \mid (\Gamma \bullet \Delta[])$$

We write $\Gamma[\Delta]$ for replacing the hole [] in Γ by Δ.
We are now ready to define the rules of the sequent calculus for the system **NL**:

Definition 2.7 (Sequent Calculus)**.** The sequent calculus presentation of **NL** is as follows:

$$\frac{}{A \vdash A}\ Id \qquad \frac{\Delta \vdash B \quad \Gamma[B] \vdash A}{\Gamma[\Delta] \vdash A}\ Cut$$

$$\frac{\Gamma[A \bullet B] \vdash C}{\Gamma[A \otimes B] \vdash C}\ \otimes L \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma \bullet \Delta \vdash A \otimes B}\ \otimes R$$

$$\frac{\Delta \vdash B \quad \Gamma[A] \vdash C}{\Gamma[\Delta \bullet B\backslash A] \vdash C}\ \backslash L \qquad \frac{B \bullet \Gamma \vdash A}{\Gamma \vdash B\backslash A}\ \backslash R$$

$$\frac{\Delta \vdash B \quad \Gamma[A] \vdash C}{\Gamma[A/B \bullet \Delta] \vdash C}\ /L \qquad \frac{\Gamma \bullet B \vdash A}{\Gamma \vdash A/B}\ /R$$

In order to "categorify" the sequent calculus, we define an equivalence relation on proofs. For this purpose, note that we can write down proofs as bracketed strings instead of drawing a whole proof tree. This is done by writing down the rule's name and, in brackets the proofs that the rule acts upon, in the order they are listed in sequent rule. For instance, $\otimes L(\otimes R(Id(A), Id(B)))$ is a proof of $A \otimes B \vdash A \otimes B$. We denote by D_i (for $i \in \mathbb{N}$) arbitrary proofs where we use the notation $LHS(D_i) = \Gamma$ and $RHS(D_i) = A$ to denote the components of the sequent that D_i is a proof of.
Our equivalence relation follows *identity unfolding* and *cut-elimination*:

Definition 2.8. We define the following equivalence relation on sequent proofs:

- Identity unfolding, by which we mean

$$\otimes L(\otimes R(Id(A), Id(B))) \equiv Id(A \otimes B)$$
$$\backslash R(\backslash L(Id(A), Id(B))) \equiv Id(A\backslash B)$$
$$/R(/L(Id(A), Id(B))) \equiv Id(B/A)$$

- Cut-elimination base case, by which we mean

$$Cut(D_1, Id(A)) \equiv D_1$$
$$Cut(Id(B), D_1) \equiv D_1$$

- Principal cut-elimination, by which we mean

$$Cut(\otimes R(D_1, D_2), \otimes L(D_3)) \equiv Cut(D_2, Cut(D_1, D_3))$$
$$Cut(\otimes R(D_1, D_2), \otimes L(D_3)) \equiv Cut(D_1, Cut(D_2, D_3))$$
$$Cut(\backslash R(D_1), \backslash L(D_2, D_3)) \equiv Cut(Cut(D_2, D_1), D_3)$$
$$Cut(\backslash R(D_1), \backslash L(D_2, D_3)) \equiv Cut(D_2, Cut(D_1, D_3))$$
$$Cut(/R(D_1), /L(D_2, D_3)) \equiv Cut(Cut(D_2, D_1), D_3)$$
$$Cut(/R(D_1), /L(D_2, D_3)) \equiv Cut(D_2, Cut(D_1, D_3))$$

- Permutative cut-elimination, by which we mean

$$Cut(\otimes L(D_1), D_2) \equiv \otimes L(Cut(D_1, D_2))$$

$$Cut(\backslash L(D_1, D_2), D_3) \equiv \backslash L(D_1, Cut(D_2, D_3))$$

$$Cut(/L(D_1, D_2), D_3) \equiv /L(D_1, Cut(D_2, D_3))$$

$$Cut(D_1, \otimes L(D_2)) \equiv \otimes L(Cut(D_1, D_2))$$

$$Cut(D_1, \backslash R(D_2)) \equiv \backslash R(Cut(D_1, D_2))$$

$$Cut(D_1, /R(D_2)) \equiv /R(Cut(D_1, D_2))$$

$$Cut(D_1, \otimes R(D_2, D_3)) \equiv \otimes R(D_2, Cut(D_1, D_3))$$
when $RHS(D_1) = C$ and $LHS(D_3) = \Gamma'[C]$

$$Cut(D_1, \otimes R(D_2, D_3)) \equiv \otimes R(Cut(D_1, D_2), D_3)$$
when $RHS(D_1) = C$ and $LHS(D_2) = \Gamma[C]$

$$Cut(D_1, \backslash L(D_2, D_3)) \equiv \backslash L(D_2, Cut(D_1, D_3))$$
when $RHS(D_1) = C$ and $LHS(D_3) = \Gamma[C][A]$

$$Cut(D_1, \backslash L(D_2, D_3)) \equiv \backslash L(Cut(D_1, D_2), D_3)$$
when $RHS(D_1) = C$ and $LHS(D_2) = \Gamma'[C]$

$$Cut(D_1, /L(D_2, D_3)) \equiv /L(D_2, Cut(D_1, D_3))$$
when $RHS(D_1) = C$ and $LHS(D_3) = \Gamma[C][A]$

$$Cut(D_1, /L(D_2, D_3)) \equiv /L(Cut(D_1, D_2), D_3)$$
when $RHS(D_1) = C$ and $LHS(D_2) = \Gamma'[C]$

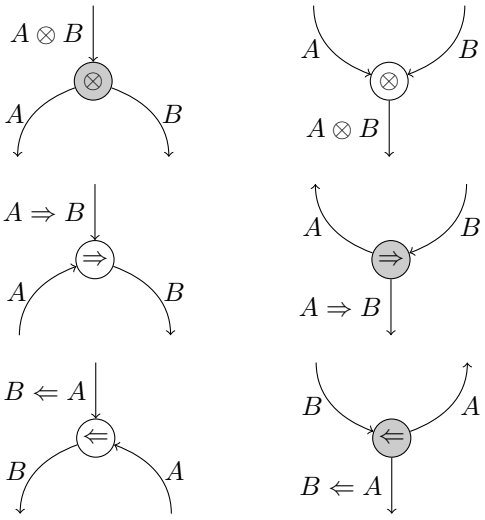
2.3.4 Proof Nets Defined

We will now define *proof nets* that we will prove to correspond to bi-closed tensor categories in the sense that well-formed equations between morphisms in a bi-closed tensor category hold if and only if they hold in their graphical language. The idea is that we can build up arbitrary proof structures (possible proof nets) by gluing together *links* and that some correctness criteria define a subclass of *proof nets*. We define some critical equations on proof structures that will give us the right tool to reason about bi-closed tensor categories graphically.

We start out with a formal definition of *links*, the basic building blocks of proof structures. Links come in two flavors: as tensor links, and as cotensor links[1]:

Definition 2.9. A labelled link is a tuple (t, i, o) where t is the type of the link, either tensor or cotensor, i is a list of input formulas of the link and o is the list of output formulas of the link.

The links for our graphical language include a tensor and cotensor link for each connective in the formula language: one link for *construction* and one link for *destruction*. We can visualize these links as little graphs that a node containing the connective under consideration and which is drawn either white or gray depending on the type of the link. The input and output formulas are then drawn as ingoing and outgoing wires, respectively. It might be clear that a constructive $\otimes$-link binds two formulas A and B together into the formula $A \otimes B$ whereas the destructive $\otimes$-link splits the two formulas. Following this analogy, it is not hard to imagine that the links will look as follows:



We want to build larger graphs out of these links, but must take care here: the resulting graph should have one unique input and one unique output, remniscent of the fact that morphisms always have one object as their domain and one object as their codomain. Moreover, the graph should be connected and *well-typed*: is should not be possible to combine two formulas A and B to form

[1]The terms *tensor* and *cotensor* are not intended to refer to any categorical notion; rather, they are intended as a reminiscent of the distinction between *tensor* and *par* links in proof nets for linear logic. The distinction, of course, also has a practical function when defining correctness criteria.

$A \otimes B$ and then decompose it as if it were another formula (for instance $A \Rightarrow B$). The following definition takes care of these prerequisites:

Definition 2.10. A proof structure is a connected graph made by the given links such that every output wire is the input wire to another link and vice versa except for a unique input wire and a unique output wire.

Because not every proof structure will correspond to an existing morphism, we need to distinguish in the class of proof structures those that will translate nicely use correctness criteria. Firstly, in order to be a proof net, the proof structure should be *planar*. Secondly, the input and output wires must be edges of the unique *external face* of the proof structure. Finally, the proof structure must satisfy *operator balance* and the *return cycle requirement*. All of these definitions follow below:

Definition 2.11 (Planarity)**.** A proof structure satisfies the planary constraint if it contains no crossing wires.

Definition 2.12 (External Face Requirement)**.** A planar proof structure satisfies the external face requirement if the unique input wire and and output wire are in the unique external face of the graph.

Definition 2.13 (Operator Balance)**.** A proof structure satisfies operator balance if every (undirected) cycle contains an equal number of tensor and cotensor nodes.

Definition 2.14 (Return Cycle Requirement)**.** A proof structure satisfies the return cycle requirement if the following three properties hold:

1. For every $\Rightarrow$ cotensor node, there is a directed path from the node through its left output, returning at the node,
2. For every $\Leftarrow$ cotensor node, there is a directed path from the node through its right output, returning at the node,
3. For every $\otimes$ cotensor node, there is no directed path from the node through one of its outputs returning at the node.

We can now simply define proof nets as follows:

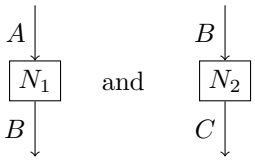
Definition 2.15 (Proof Nets)**.** A proof structure is a proof net iff it satisfies the planarity constraint, the external face requirement, operator balance and the return cycle requirement.

Our next goal is to show that we can also define proof nets inductively. We start with an inductive definition and proceed by showing that it in fact defines the whole class of proof nets. We use the notation N^* for a net that should be drawn upside-down, i.e. mirrored vertically along an imaginary axis.

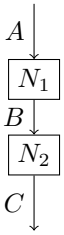
Definition 2.16 (Proof Nets Inductively)**.** The class of proof nets is defined inductively as follows:

- **Identity.** The identity proof net for arbitrary A is given by

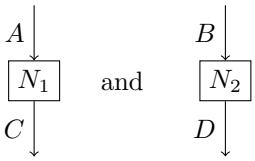
- **Composition.** Given two proof nets



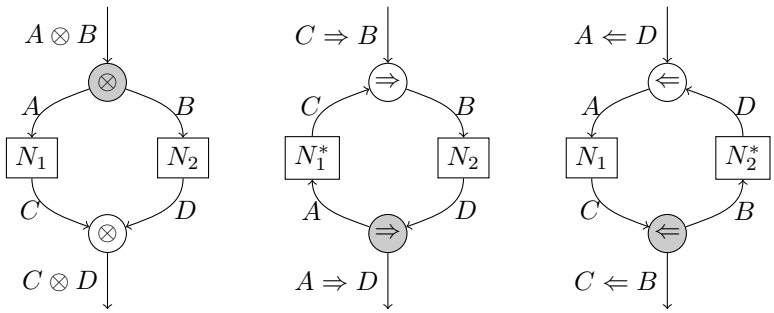
the following is a proof net:



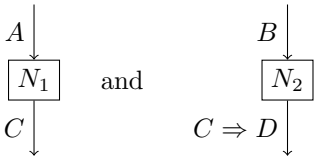
- **Monotonicity.** Given two proof nets



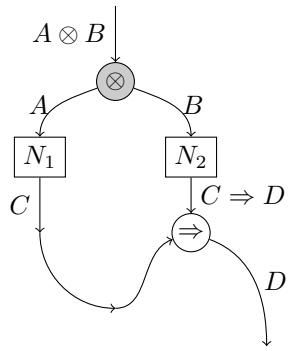
the following are proof nets:



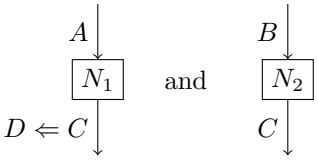
- **Generalized Left Application.** Given two proof nets



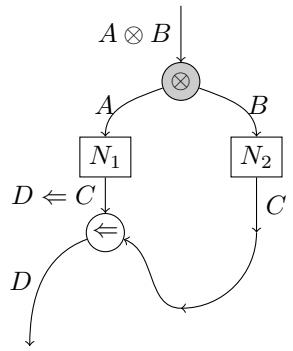
the following is a proof net:



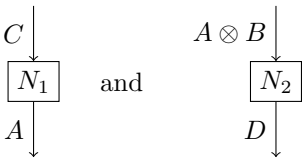
- **Generalized Right Application.** Given two proof nets



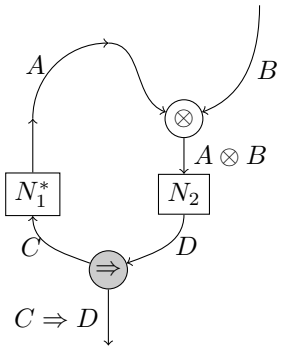
the following is a proof net:



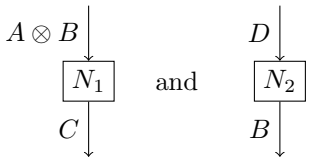
- **Generalized Left Co-Application.** Given two proof nets



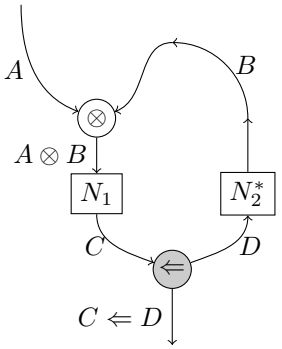
the following is a proof net:



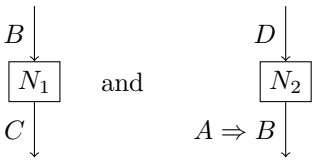
- **Generalized Right Co-Application.** Given two proof nets



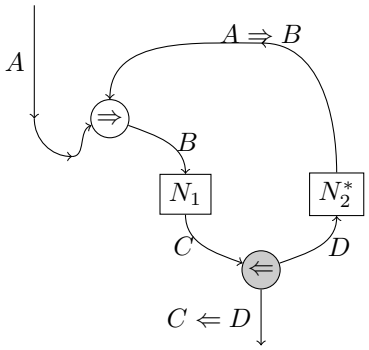
the following is a proof net:



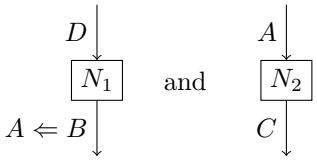
- **Generalized Left Lifting.** Given two proof nets



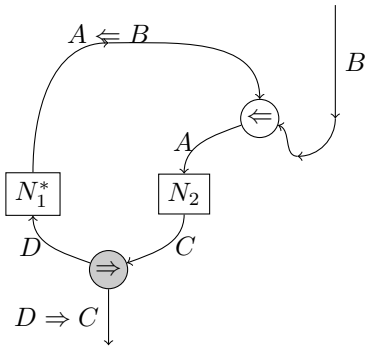
the following is a proof net:



- **Generalized Right Lifting.** Given two proof nets



the following is a proof net:



- Nothing else is a proof net.

Now we want to show that our inductive definition is correct, i.e. it defines precisely those proof structures that are proof nets. Before we prove this theorem, we need to show some additional properties of proof nets.

Property 2.1. *Every proof structure has an equal number of tensor and cotensor nodes.*

Proof. By definition, every proof structure has a unique input and a unique output. By well-formedness, an output of a link cannot be an input to the same link. Now, as each tensor node has 2 inputs and 1 output, and each cotensor node has 1 input and 2 outputs, this means that the *in degree* (i.e. the number of unconnected inputs) for a proof structure with n tensor nodes and m cotensor

nodes is $2n+m-(2m+n-1) = n-m+1$ while the *out degree* is $n+2m-(m+2n-1) = m-n+1$. Now suppose that $n \neq m$. Then either the in or the out degree is 0 or less, which is impossible by the assumption that there is one unique input and one unique output. Hence, $n = m$. □

Property 2.2. *Every connected, well-typed graph made up from the given links has a directed path from any input to any output.*

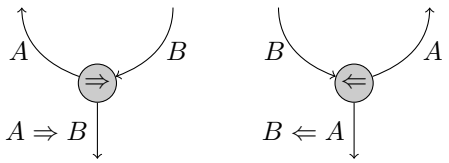
Proof. By induction on the number of nodes in a graph. The base case of zero nodes is trivial as the graph must consist of a single wire. For the inductive case, suppose that each graph with n nodes has a directed path from any input to any output, and consider a proof structure with $n+1$ nodes. Now consider any input wire and any output wire. By connectedness, the input wire must enter some node. As every node has at least one output, consider the outputs of the node. If one of the outputs is the selected output wire, we are done. Otherwise proceed by considering one of the output wires of the node and proceed using the induction hypothesis. □

Property 2.3 (Locality)**.** *For any proof net, each of its subgraphs with a unique in- and output, all correctness criteria hold, i.e. each of these subgraphs are proof nets as well.*

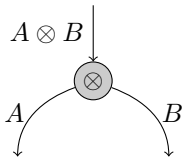
Proof. Obviously, every subgraph of a proof net with a unique in- and output is a proof structure. It is also immediate that planarity and operator balance hold locally, since a violation of these in the subgraph would mean a violation in the whole graph. A similar reasoning holds for the return cycle requirement. Hence, we may conclude that the subgraph must be a proof net. □

Theorem 2.2. *A proof structure is a proof net if and only if it is generated by the inductive definition of proof nets.*

Proof. The converse is easily shown by inspecting the generating rules and seeing that they preserve operator balance and the return cycle requirement. For the left to right direction, things are more difficult. We proceed by induction on the number of nodes. As proof nets have an equal number of tensor and cotensor nodes, we will try to show that we can take off one tensor and one cotensor node at a time to proceed with the induction hypothesis. The base case is trivial, as it can only be a proof net consisting of a single wire, which is also the base case of the inductive definition.
For the inductive case, suppose that any proof net with up to n nodes can be inductively generated, and consider a proof net N with $n+2$ nodes. First of all, because of the external face requirement, we can always identify a *top node* and a *bottom node* as outermost nodes of the proof net. Secondly we note that a proof net cannot have as top node (i.e. the node with the unique loose input wire) the following two links:



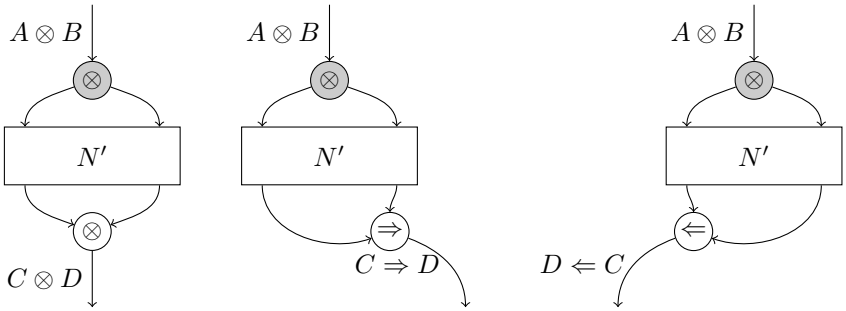
This is because the left (resp. right) output wire cannot possibly give a directed path to the (loose) input, hence the return cycle requirement is violated. For the same reason, a proof net cannot have as a bottom node (i.e. the node with the unique loose output wire) the following link:



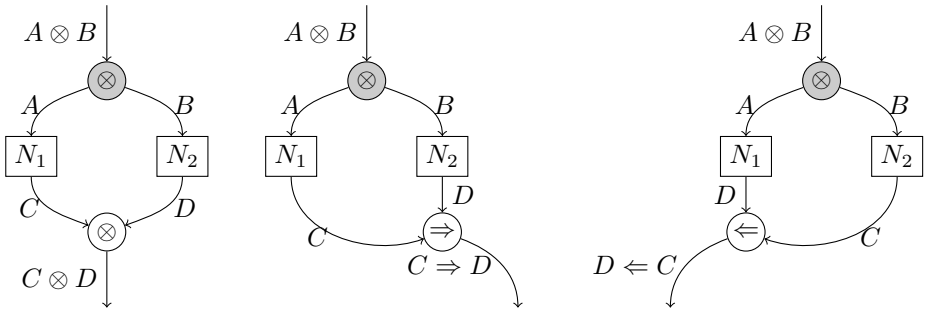
This is because, by property 2.2, there will always be a path from the outputs of this link, back to the node, and hence the return cycle requirement will be violated.

We will now proceed by case distinction considering different cases where N starts with a certain link and ends with a certain link (although there are a lot of cases to go through, we will treat them systematically):

1. The case where N starts with a cotensor node and ends with a tensor node: so N is of one of the following forms:



In this case, we claim that N is either composite or actually of the form

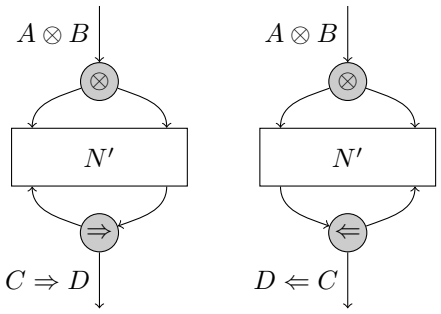


where N_1 and N_2 are proof nets.

Now we know that N must be either such that there is a *splitting wire*, i.e. a wire where we can split N into two subgraphs by only splitting at that wire. In this case, we have by locality that the top and bottom parts must be proof nets, and moreover, as they both contain at least 1 node, they at least contain 2 nodes, meaning that both parts cannot contain more than n nodes. Thus, we can apply the induction hypothesis and use the composition case of the inductive definition of proof nets.

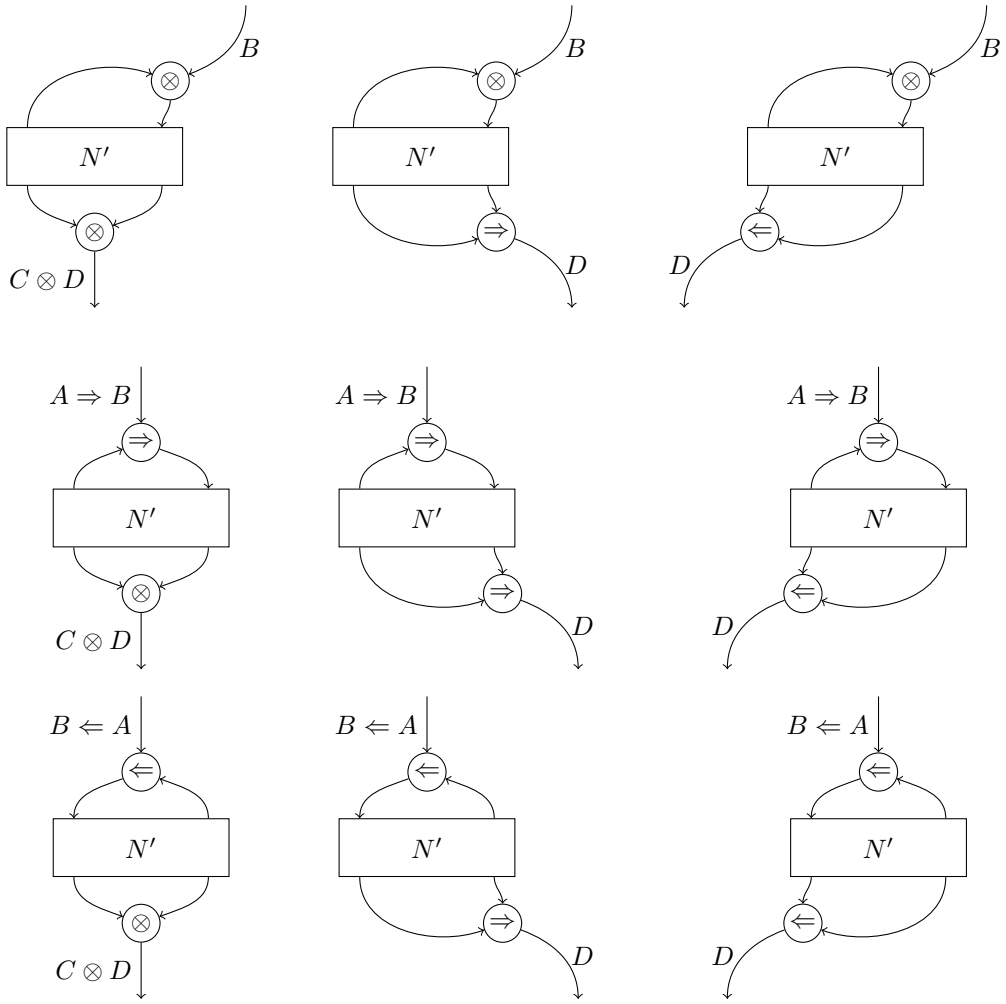
Now suppose that N does not contain a splitting wire. We show disjointness of the left and right part of the proof net by contradiction. Suppose the left and right part are indeed connected, butmnot in a way such that the whole proof net N is of a composite form. Suppose that there is a connection from left to right. This means that in the right part, there must be one tensor node extra and in the left part, there must be a cotensor node extra (simply because of the number of in- and outputs in the left and right part). However, by connectedness, the tensor and cotensor nodes are both connected to each other as well as connected to the top (cotensor) node, which induces a cycle without operator balance. Hence, we have arrived at a contradiction.

2. The cases where N starts with a cotensor node and ends with a cotensor node. These are the following (note that the left (resp. right) output of the $\Rightarrow$ (resp. $\Leftarrow$) node cannot be the end of the graph as that would violate the return cycle requirement):



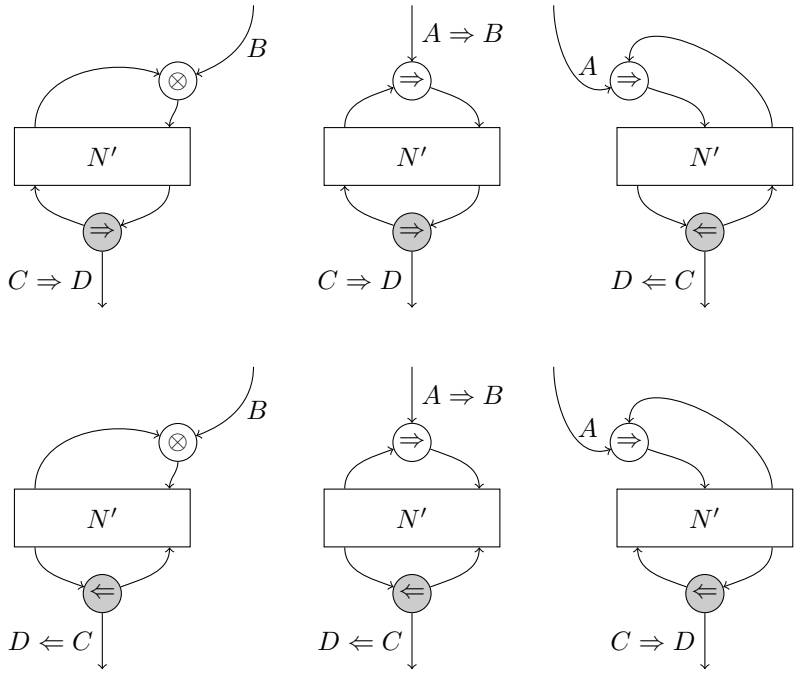
Again, we know that these proof nets either contain a splitting wire, and hence they are of a composite form, in which case we can apply the induction hypothesis, or they do not contain a splitting wire. In the latter case we know that there must be at least two tensor nodes in N', and asthe graph is not of a composite form, we must have either that (i) the left and right part are disjoint or (ii) the left and right part are not disjoint. Case (i) is not possible as this would mean (in the left graph) that there are two inputs and no output, which is not possible. So we must consider case (ii). But in this case, by connectedness and the fact there must be two tensor nodes, there must be a cycle with two tensor nodes and one cotensor node, which violates operator balance. Hence, these graphs can only be proof nets by virtue of composition, which is present in the inductive definition.

3. The cases where N starts with a tensor node and ends in a tensor node. These are the following cases (note that the cases where the left input of the starting $\otimes$ node are the unique loose input are symmetric and are thus treated analogously. Similar for the cases where the left (resp. right) input of a $\Rightarrow$ (resp. $\Leftarrow$) node is the unique loose input):

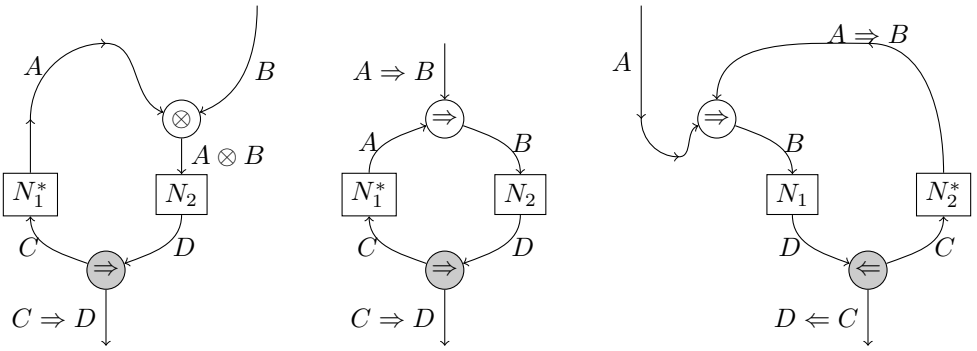


Again, we claim that these structures can only be proof nets by virtue of composition: the left and right side cannot be disjoint, because then one of the parts has two outputs and no input, which is not possible. So the left and right side must be connected. As there should be at least two cotensor nodes in N' and as these must be connected, we get that there must be a cycle violating operator balance.

4. Finally, we have the cases where N starts with a tensor node and ends with a cotensor node. These are the following (note that we have ommitted, for the sake of space, some symmetric cases here):



Again, the case of composition is handled by the inductive definition. We claim that the bottom three nets cannot in fact be proof nets if they are not composite. For each of these diagrams it holds that there must be at least one tensor node on one side, and one cotensor node on the other side, and these sides must then be connected. But this induces a cycle that does not satisfy operator balance. Hence, we have contradicted the assumption that these are proof nets.
We claim that the top three nets are of the form



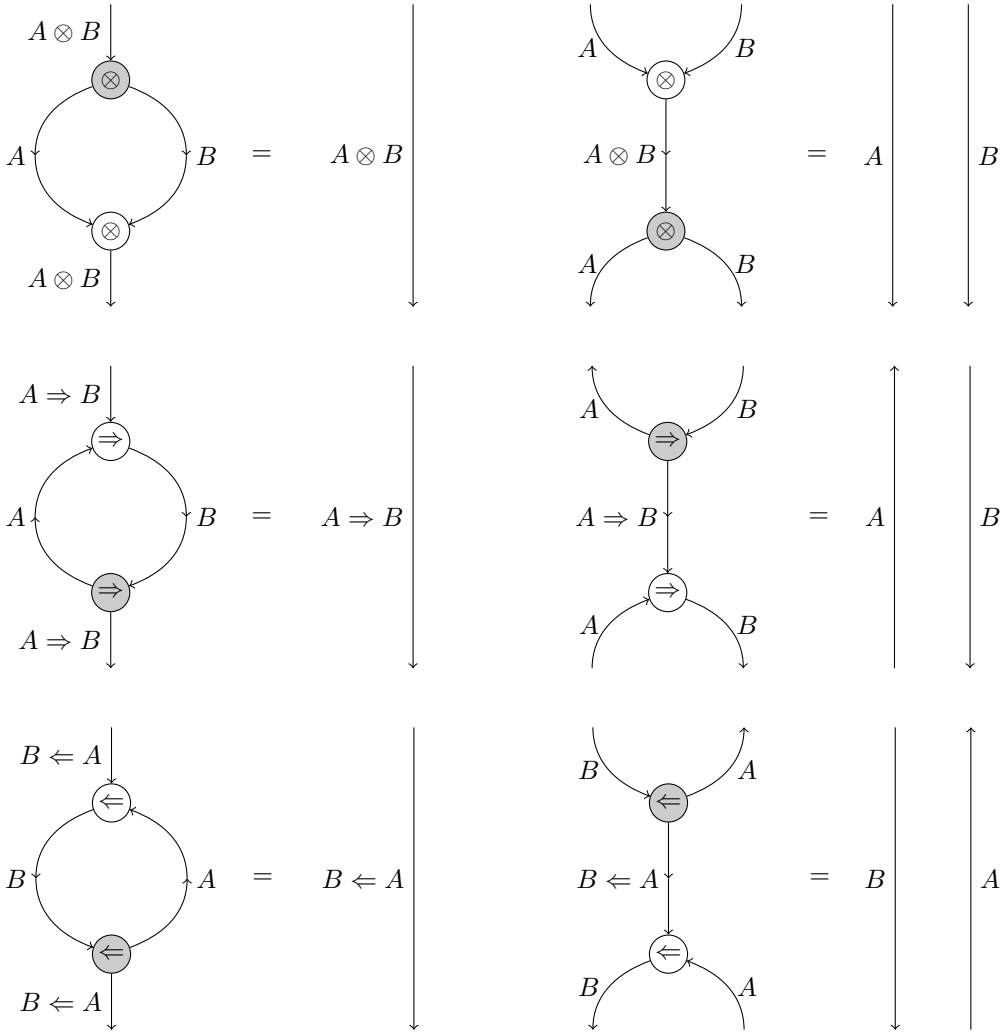
Again, all we need to do is show disjointness of the left and right parts. We proceed again by contradiction. But the assumption the left and right part are indeed connected means admitting that there must be a cycle violating operator balance. Hence, the left and right part are disjoint, and as they each have one input and one output, they have an equal number of tensor and cotensor nodes. Furthermore, as the correctness criteria also apply locally, we may conclude that N_1 and N_2

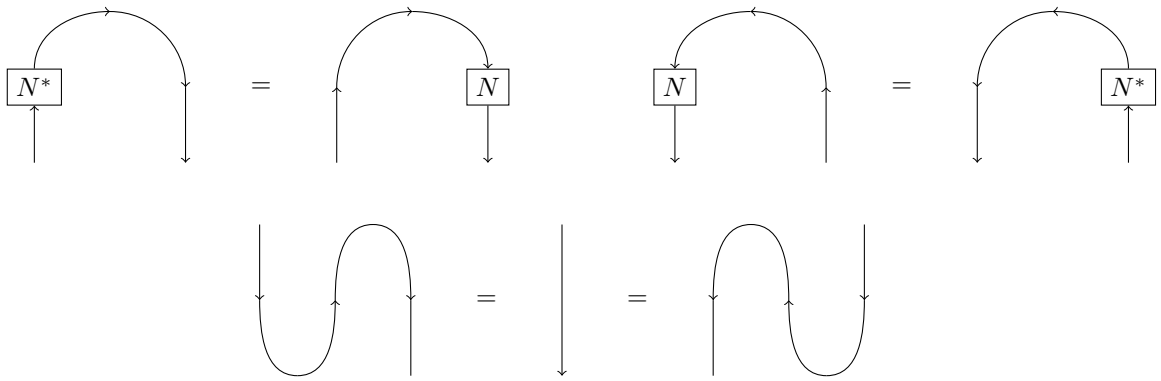
are proof nets. Applying the induction hypothesis on these nets, and given that these constructions are cases in the inductive definition, we are done. □

Now that we have a correct inductive definition of proof nets, it's time to define *equality* on them.

2.3.5 Equations on Proof Nets

We will define an equivalence relation on proof nets, that will allow for a categorical interpretation of the equivalence classes of nets. In addition to the standard requirement of reflexivity, transitivity, and symmetry, we require the following equations:

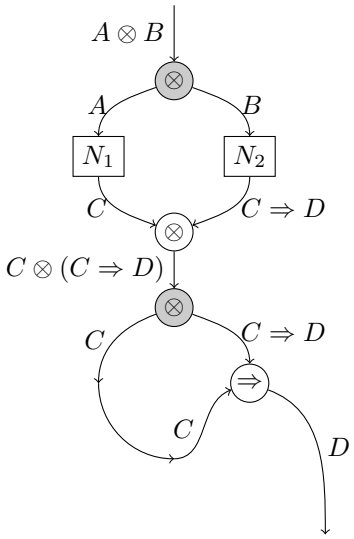




It is interesting to note that we can now simplify our inductive definition of proof nets, reminiscent of Došen's axiomatization of the non-associative Lambek Calculus (Došen, 1988). We will indeed show that one can do with only the first four generating rules of proof nets, namely composition and the three monotonicity rules. We will then need to add four base cases, namely (non-generalized) application and co-application nets. What we get is the following lemma:

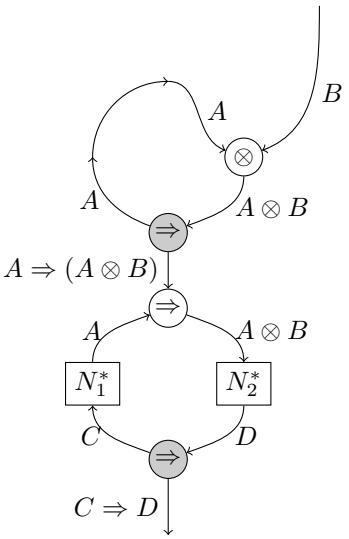
Lemma 2.1. *Given the proof net equations, all proof nets can be generated by the identity net, non-generalized application and co-application, composition and monotonicity.*

Proof. We only need to show that we can derive generalized application, generalized co-application and generalized lifting. We will only consider the left version of these rules, the right variant being symmetric. Generalized left application is obtained by composing $\otimes$ monotonicity and non-generalized application:

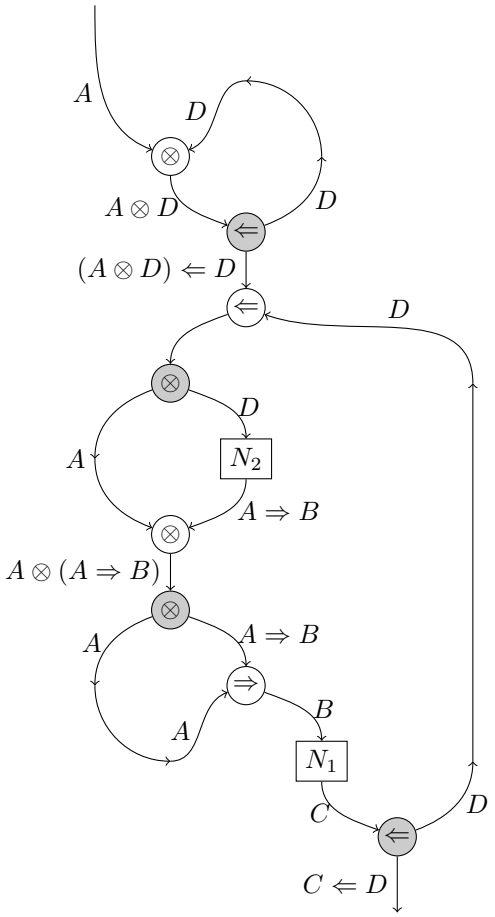


which clearly is equal to generalized left application. We can obtain generalized left co-application

by a similar process, namely by composing non-generalized left co-application with $\Rightarrow$ monotonicity. This gives us the following net:



which is equal to generalized left co-application. Finally we need to show that we can derive generalized left lifting. This is somewhat more complex, but amounts to composing non-generalized right co-application with $\Leftarrow$ monotonicity applied to the identity net of D and the composition of $\otimes$ monotonicity on N_2 and non-generalized left application. In other words, we get the following net:



and by eliminating the appropriate nodes, one can see that this net equals generalized left lifting. □

Although the Došen axiomatization for proof nets is considerably more simple, we keep the original sequent calculus presentation of **NL** because it allows for a translation into sequent proofs *by the link*, i.e. we can define the translation in terms of nodes of a proof net instead of just mapping the basic application and co-application nets to the application and co-application rules.

2.3.6 Sequentialization

We will now make the connection between proof nets and sequent proofs explicit, by a process called *sequentialization*, i.e. a translation from proof nets into sequent proofs. We present a translation and show that every proof net translates properly into a sequent proof of a sequent of the form $A \vdash B$. The translation is given in Figure 2.3.

An important lemma we want to prove is that every proof net translates to a valid sequent proof. This is called the *sequentialization lemma*:

Lemma 2.2. *Every proof net sequentializes.*

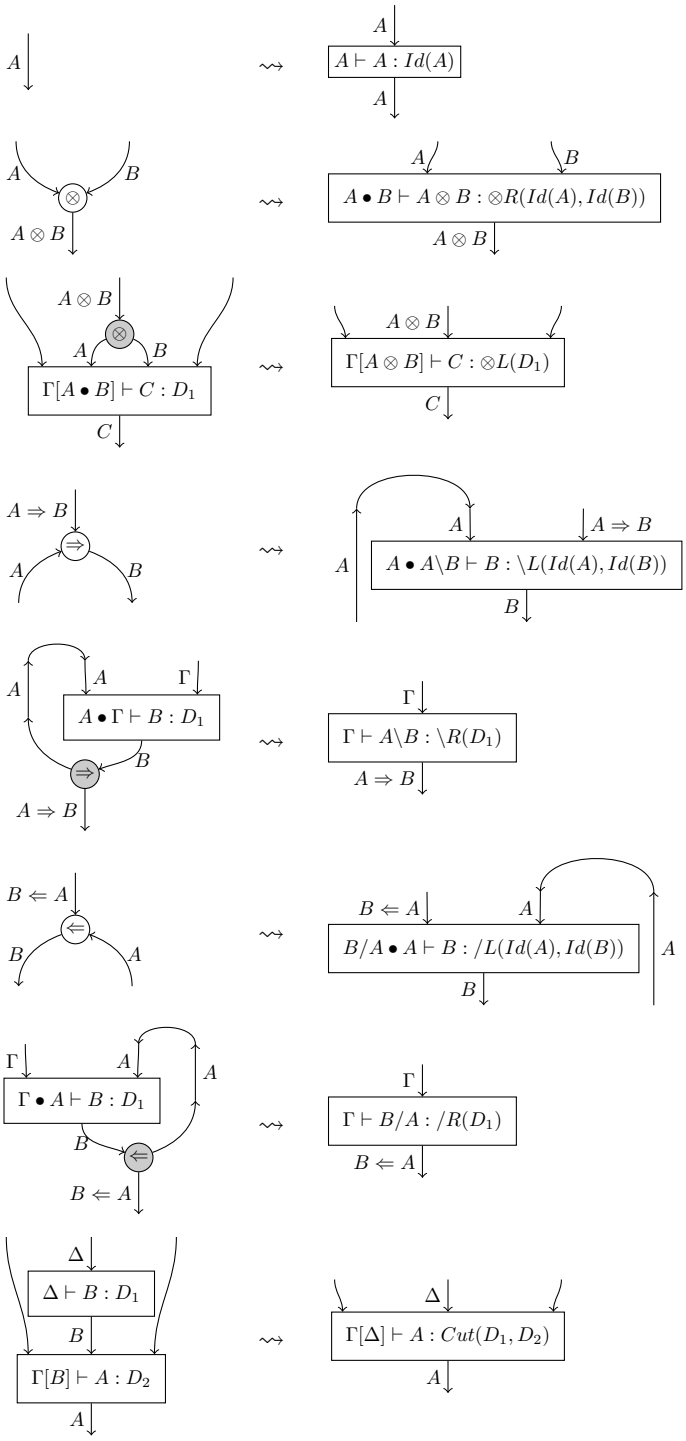


Figure 2.3: Sequentialization for proof nets

Proof. By induction on proof nets. For the base case, the identity proof net obviously sequentializes to the identity proof. For the inductive cases we have the following:

- Suppose N_1 and N_2 sequentialize to $A \vdash B : D_1$ and $B \vdash C : D_2$, respectively. Composition sequentializes by a cut, giving $A \vdash C : Cut(D_1, D_2)$.
- Suppose N_1 and N_2 sequentialize to $A \vdash C : D_1$ and $B \vdash D : D_2$, respectively. Then the new $\otimes$ proof net sequentializes as follows: The bottom sequentializes to $C \bullet D \vdash C \otimes D : \otimes R(Id(C), Id(D))$ and we can choose to either cut in D_1 first and then D_2 or the other way around, giving respectively

  $$Cut(D_2, Cut(D_1, \otimes R(Id(C), Id(D))))$$

  or

  $$Cut(D_1, Cut(D_2, \otimes R(Id(C), Id(D))))$$

  . Finally, the top has one sequent proof below it and the whole sequentializes to either

  $$A \otimes B \vdash C \otimes D : \otimes L(Cut(D_2, Cut(D_1, \otimes R(Id(C), Id(D)))))$$

  or

  $$A \otimes B \vdash C \otimes D : \otimes L(Cut(D_1, Cut(D_2, \otimes(Id(C), Id(D)))))$$

  . The new $\Rightarrow$ proof net sequentializes as follows: the top sequentializes by itself into $C \bullet C \backslash B \vdash B : \backslash L(Id(C), Id(B))$ whereafter N_1^* can be mirrored again into N_1. Then we get a choice to first cut D_1 with the top and then with D_2 or the other way around, leaving us with either

  $$A \bullet C \backslash B \vdash D : Cut(Cut(D_1, \backslash L(Id(C), Id(B))), D_2)$$

  or

  $$A \bullet C \backslash B \vdash D : Cut(D_1, Cut(\backslash L(Id(C), Id(B)), D_2))$$

  . Finally, the bottom can sequentialize the whole to either

  $$C \backslash B \vdash A \backslash D : \backslash R(Cut(Cut(D_1, \backslash L(Id(C), Id(B))), D_2))$$

  or

  $$C \backslash B \vdash A \backslash D : \backslash R(Cut(D_1, Cut(\backslash L(Id(C), Id(B)), D_2)))$$

  . The new $\Leftarrow$ proof net sequentializes in a similar way to either

  $$A / D \vdash C / B : / R(Cut(Cut(D_2, / L(Id(D), Id(A)), D_1))$$

  or

  $$A / D \vdash C / B : / R(Cut(D_2, Cut(/ L(Id(D), Id(A)), D_1)))$$

  .
- Suppose N_1 and N_2 sequentialize to $A \vdash C : D_1$ and $B \vdash C \backslash D : D_2$, respectively. Then the new $\otimes, \Rightarrow$ proof net sequentializes as follows: the bottom sequentializes to $C \bullet C \backslash D \vdash D : \backslash L(Id(C), Id(D))$ whereafter we can again choose to cut in either D_1 first and then D_2 or the other way around, leaving us with either

  $$A \bullet B \vdash D : Cut(D_2, Cut(D_1, \backslash L(Id(C), Id(D))))$$

or

$$A \bullet B \vdash D : Cut(D_1, Cut(D_2, \backslash L(Id(C), Id(D))))$$

. Then, the top part sequentializes the whole into either

$$A \otimes B \vdash D : \otimes L(Cut(D_2, Cut(D_1, \backslash L(Id(C), Id(D)))))$$

or

$$A \otimes B \vdash D : \otimes L(Cut(D_1, Cut(D_2, \backslash L(Id(C), Id(D)))))$$

.

- The case of the new $\otimes, \Leftarrow$ proof net is similar to the previous item.
- Suppose N_1 and N_2 sequentialize to $C \vdash A : D_1$ and $A \otimes B \vdash D : D_2$, respectively. Then the new $\otimes, \Rightarrow$ proof net sequentializes as follows: The top part with the $\otimes$ node sequentializes to $A \bullet B \vdash A \otimes B : \otimes R(Id(A), Id(B))$, whereafter N_1^* can get mirrored again into N_1, and we choose to cut either to get

$$C \bullet B \vdash D : Cut(Cut(D_1, \otimes R(Id(A), Id(B))), D_2)$$

or

$$C \bullet B \vdash D : Cut(D_1, Cut(\otimes R(Id(A), Id(B)), D_2))$$

. Then, the bottom part sequentializes the whole into either

$$B \vdash C \backslash D : \backslash R(Cut(Cut(D_1, \otimes R(Id(A), Id(B))), D_2))$$

or

$$B \vdash C \backslash D : \backslash R(Cut(D_1, Cut(\otimes R(Id(A), Id(B)), D_2)))$$

.

- The case of the new $\otimes, \Leftarrow$ proof net is similar to the previous item.
- Suppose N_1 and N_2 sequentialize to $B \vdash C : D_1$ and $D \vdash A \backslash B : D_2$, respectively. Then the new $\Rightarrow, \Leftarrow$ proof net sequentializes as follows: The top part with the $\Rightarrow$ node sequentializes to $A \bullet A \backslash B \vdash B : \backslash L(Id(A), Id(B))$ whereafter the incoming A wire can be straightened by the left snake equation, and N_2^* can be mirrored into N_2. Then we choose which cut to perform first, resulting in either

$$A \bullet D \vdash C : Cut(Cut(D_2, \backslash L(Id(A), Id(B))), D_1)$$

or

$$A \bullet D \vdash C : Cut(D_2, Cut(\backslash L(Id(A), Id(B)), D_1))$$

. Then, the bottom part sequentializes with the whole into either

$$A \vdash C / D : / R(Cut(Cut(D_2, \backslash L(Id(A), Id(B))), D_1))$$

or

$$A \vdash C / D : / R(Cut(D_2, Cut(\backslash L(Id(A), Id(B)), D_1)))$$

.

- The case of the new $\Leftarrow, \Rightarrow$ proof net is similar to the previous item.

□

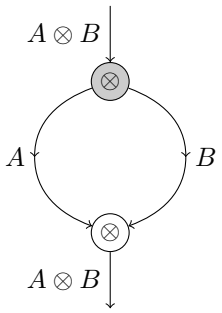
To show that this is in fact a deterministic translation, we prove the following lemma:

Lemma 2.3. *For any two proof nets that are equal, their sequentializations are equivalent.*

Proof. We first show that the sequentialization translation can be made deterministic by showing that the possible choice results for each net are equivalent and thus can be solved by stating that the net is sent to an equivalence class of sequent proofs. We then proceed by showing that all the equations on proof nets effectively are translated into the same equivalence class.
To show that the sequentialization translation can be made deterministic, observe (for instance) that for nets N_1 and N_2 that sequentialize to $A \vdash C : D_1$ and $B \vdash D : D_2$ respectively, we have that

$$\begin{gathered}
\otimes L(Cut(D_2, Cut(D_1, \otimes R(Id(C), Id(D))))) \equiv \\
\otimes L(Cut(D_2, \otimes R(Cut(D_1, Id(C)), Id(D)))) \equiv \\
\otimes L(Cut(D_2, \otimes R(D_1, Id(D)))) \equiv \\
\otimes L(\otimes R(D_1, Cut(D_2, Id(D)))) \equiv \\
\otimes L(\otimes R(D_1, D_2)) \equiv \\
\otimes L(\otimes R(Cut(D_1, Id(C)), D_2)) \equiv \\
\otimes L(Cut(D_1, \otimes R(Id(C), D_2)) \equiv \\
\otimes L(Cut(D_1, \otimes R(Id(C), Cut(D_2, Id(D))))) \equiv \\
\otimes L(Cut(D_1, Cut(D_2, \otimes R(Id(C), Id(D))))).
\end{gathered}$$

This process is similar for all other items in the translation, and we thus invite the reader to do the relevant computations. To see that all equations on proof nets are translated into the same equivalence class, then, we observe that

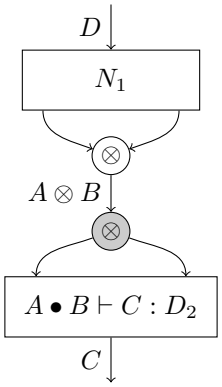


sequentializes to $A \otimes B \vdash A \otimes B : \otimes L(\otimes R(Id(A), Id(B)))$, but we have that

$$\otimes L(\otimes R(Id(A), Id(B))) \equiv Id(A \otimes B)$$

, which in turn is the sequentialization of

This process is similar for the identity cut equations for $\Rightarrow$ and $\Leftarrow$. We now consider the general cut equations, and observe that in



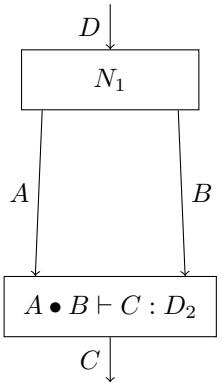
we can sequentialize the cotensor node for $\otimes$ away, giving $\otimes L(D_2)$ whereas the tensor node for $\otimes$ sequentializes to $\otimes R(Id(A), Id(B))$. We can then perform a cut to get

$$Cut(\otimes R(Id(A), Id(B)), \otimes L(D_2)$$

which is equivalent to

$$Cut(Id(A), Cut(Id(B), D_2))$$

which is equivalent to D_2 again, but now we are left with



so we are done. The cases for $\Rightarrow$ and $\Leftarrow$ are similarly treated. We just need to mention here that the final four equations, the two about bending around maps and the snake equations are really just for bookkeeping, i.e. making sure that everything works out well. $\square$

Now that we have established a deterministic, equality-preserving translation from proof nets to sequent proofs, we want to compose this translation with one that maps sequent proofs to morphisms in a bi-closed tensor category. This is done in the next section.

2.3.7 From Sequent Proofs to Categorical Morphisms

We now want to translate sequent proofs that are proofs of sequents of the form $A \vdash B$ (as every sequentialization results in this form) into categorical morphisms. To this end, we define a translation from sequent rules into morphisms. But first, we need some additional notions, as in general, sequents may not have a single formula in the left-hand side. Hence, we define the formula interpretation of structures:

Definition 2.17 (Formula Interpretation)**.** A structure converts to a formula, denoted Γ° as follows:

- $A^\circ = A$,
- $(\Gamma \bullet \Delta)^\circ = \Gamma^\circ \otimes \Delta^\circ$.

We will now show that for every sequent proof $\Gamma \vdash A : D_1$ in **NL** over some set At of atomic formulae, there is a translation $T(D_1) : \Gamma^\circ \to A$ in the bi-closed tensor category given by the free set of objects over At. But before we define the translation, we need an additional lemma:

Lemma 2.4. *For every context $\Gamma[]$ and morphism $f : \Delta^\circ \to A$ there is a morphism $op(f) : \Gamma[\Delta]^\circ \to \Gamma[A]^\circ$.*

Proof. By induction on $\Gamma[]$. The base case is $[]$, and we define $op(f) = f$ there. In the inductive case $(\Gamma'[] \bullet \Delta')$, suppose that there is $op'(f) : \Gamma'[\Delta] \to \Gamma'[A]$. Then define $op(f) = op'(f) \otimes id_{\Delta'^\circ}$. Similarly for the case $(\Gamma' \bullet \Delta'[])$. □

We can now define the translation T, it is given in Figure 2.4.

As we will want to define a full translation from proof nets to categorical morphisms, we will want to show that the equivalence classes of sequent proofs are sent to identical morphisms. In other words, we need to show the following:

Lemma 2.5. *Every equivalence between sequent proofs $D_1 \equiv D_2$ becomes equality of morphisms $T(D_1) = T(D_2)$.*

Proof. We need to show this case by case on the equivalence relation on sequent proofs. We illustrate each of the cases by a table showing the equivalent proofs and their translations.
For the case of identity unfolding we have

$\otimes L(\otimes R(Id(A), Id(B)))$ $id_A \otimes id_B$	$Id(A \otimes B)$ $id_{A\otimes B}$
$\backslash R(\backslash L(Id(A), Id(B)))$ $id_A \Rightarrow id_B$	$Id(A \backslash B)$ $id_{A \Rightarrow B}$
$/R(/L(Id(A), Id(B)))$ $id_B \Leftarrow id_A$	$Id(B/A)$ $id_{B \Leftarrow A}$

where it is clear that the equivalences become correct equations under translation. For the base case of cut-elimination we have

$Cut(D_1, Id(B))$ $id_B \circ T(D_1)$	D_1 $T(D_1)$
$Cut(Id(A), D_1)$ $T(D_1) \circ op(id_A)$	D_1 $T(D_1)$

$$\frac{}{A \vdash A}\ Id \quad \rightsquigarrow \quad id_A : A \to A$$

$$\frac{\Delta \vdash B \quad \Gamma[B] \vdash A}{\Gamma[\Delta] \vdash A}\ Cut \quad \rightsquigarrow \quad \frac{f : \Delta^\circ \to B \quad g : \Gamma[B]^\circ \to A}{g \circ op(f) : \Gamma[\Delta]^\circ \to A}$$

$$\frac{\Gamma[A \bullet B] \vdash C}{\Gamma[A \otimes B] \vdash C}\ \otimes L \quad \rightsquigarrow \quad \frac{f : \Gamma[A \bullet B]^\circ \to C}{f : \Gamma[A \bullet B]^\circ \to C}$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma \bullet \Delta \vdash A \otimes B}\ \otimes R \quad \rightsquigarrow \quad \frac{f : \Gamma^\circ \to A \quad g : \Delta^\circ \to B}{f \otimes g : (\Gamma \bullet \Delta)^\circ \to A \otimes B}$$

$$\frac{\Delta \vdash B \quad \Gamma[A] \vdash C}{\Gamma[\Delta \bullet B \backslash A] \vdash C}\ \backslash L \quad \rightsquigarrow \quad \frac{f : \Delta^\circ \to B \quad g : \Gamma[A]^\circ \to C}{g \circ op(\lhd^{-1}(f \Rightarrow id_A))}$$

$$\frac{B \bullet \Gamma \vdash A}{\Gamma \vdash B \backslash A}\ \backslash R \quad \rightsquigarrow \quad \frac{f : (B \bullet \Gamma)^\circ \to A}{\lhd f : \Gamma^\circ \to B \Rightarrow A}$$

$$\frac{\Delta \vdash B \quad \Gamma[A] \vdash C}{\Gamma[A / B \bullet \Delta] \vdash C}\ / L \quad \rightsquigarrow \quad \frac{f : \Delta^\circ \to B \quad g : \Gamma[A]^\circ \to C}{g \circ op(\rhd^{-1}(id_A \Leftarrow f))}$$

$$\frac{\Gamma \bullet B \to A}{\Gamma \vdash A / B}\ / R \quad \rightsquigarrow \quad \frac{f : (\Gamma \bullet B)^\circ \to A}{\rhd f : \Gamma^\circ \to A \Leftarrow B}$$

Figure 2.4: From Sequent Proofs to Categorical Morphisms

where it is quite obvious that the equivalences becomes correct equations under translation, as *op* will only insert identity morphisms.
For the case of principal cut-elimination we have

$Cut(\otimes R(D_1, D_2), \otimes L(D_3))$ $T(D_3) \circ op_1(T(D_1) \otimes T(D_2))$	$Cut(D_2, Cut(D_1, D_3))$ $T(D_3) \circ op_2(T(D_1)) \circ op_3(T(D_2))$
$Cut(\otimes R(D_1, D_2), \otimes L(D_3))$ $T(D_3) \circ op_4(T(D_1) \otimes T(D_2))$	$Cut(D_1, Cut(D_2, D_3))$ $T(D_3) \circ op_5(T(D_2)) \circ op_6(T(D_1))$
$Cut(\backslash R(D_1), \backslash L(D_2, D_3))$ $T(D_3) \circ op_7(\beta^{-1}(T(D_2) \Rightarrow id_A)) \circ op_8(\beta(T(D_1)))$	$Cut(Cut(D_2, D_1), D_3)$ $T(D_3) \circ op_9(T(D_1) \circ op_{10}(T(D_2)))$
$Cut(\backslash R(D_1), \backslash L(D_2, D_3))$ $T(D_3) \circ op_{11}(\beta^{-1}(T(D_2) \Rightarrow id_A)) \circ op_{12}(\beta(T(D_1)))$	$Cut(D_2, Cut(D_1, D_3))$ $T(D_3) \circ op_{13}(T(D_1)) \circ op_{14}(T(D_2))$
$Cut(/R(D_1), /L(D_2, D_3))$ $T(D_3) \circ op_{15}(\gamma^{-1}(id_A \Leftarrow T(D_2))) \circ op_{16}(\gamma(T(D_1)))$	$Cut(Cut(D_2, D_1), D_3)$ $T(D_3) \circ op_{17}(T(D_1) \circ op_{18}(T(D_2)))$
$Cut(/R(D_1), /L(D_2, D_3))$ $T(D_3) \circ op_{19}(\gamma^{-1}(id_A \Leftarrow T(D_2))) \circ op_{20}(\gamma(T(D_1)))$	$Cut(D_2, Cut(D_1, D_3))$ $T(D_3) \circ op_{21}(T(D_1)) \circ op_{22}(T(D_2))$

where we need to ensure that the translations are actually equal. But observe (for $\otimes$) that

$$\begin{gathered} op_1(T(D_1) \otimes T(D_2)) = \\ op_1((T(D_1) \circ id_{\Gamma'^\circ}) \otimes (id_B \circ T(D_2))) = \\ op_1((T(D_1) \otimes id_B) \circ (id_{\Gamma'^\circ} \otimes T(D_2))) = \\ op_1(T(D_1) \otimes id_B) \circ op_1(id_{\Gamma'^\circ} \otimes T(D_2)) = \\ op_2(T(D_1)) \circ op_3(T(D_2)) \end{gathered}$$

and that

$$\begin{gathered} op_4(T(D_1) \otimes T(D_2)) = \\ op_4((id_A \circ T(D_1)) \otimes (T(D_2) \circ id_{\Delta^\circ})) = \\ op_4((id_A \otimes T(D_2)) \circ (T(D_1) \otimes id_{\Delta^\circ})) = \\ op_4(id_A \otimes T(D_2)) \circ op_4(T(D_1) \otimes id_{\Delta^\circ}) = \\ op_5(T(D_2)) \circ op_6(T(D_1)). \end{gathered}$$

For $\Rightarrow$ (the case of $\Leftarrow$ is symmetric) we observe that

$$\begin{gathered} op_7(\beta^{-1}(T(D_2) \Rightarrow id_A)) \circ op_8(\beta(T(D_1))) = \\ op_7(\beta^{-1}(T(D_2) \Rightarrow id_A)) \circ op_7(id_{\Delta^\circ} \otimes \beta(T(D_1))) = \\ op_7(\beta^{-1}(T(D_2) \Rightarrow id_A) \circ (id_{\Delta^\circ} \otimes \beta(T(D_1)))) = \\ op_7(id_A \circ \beta^{-1}(T(D_2) \Rightarrow id_A) \circ (id_{\Delta^\circ} \otimes \beta(T(D_1)))) = \\ op_7(\beta^{-1}((id_{\Delta^\circ} \Rightarrow id_A) \circ (T(D_2) \Rightarrow id_A) \circ \beta(T(D_1)))) = \\ op_7(\beta^{-1}((T(D_2) \Rightarrow id_A) \circ \beta(T(D_1)) \circ id_{\Gamma'^\circ})) = \\ op_7(\beta^{-1}(\beta(id_A \circ T(D_1) \circ (T(D_2) \otimes id_{\Gamma'^\circ})))) = \\ op_7(T(D_1) \circ (T(D_2) \otimes id_{\Gamma'^\circ})) = \\ op_9(T(D_1) \circ (T(D_2) \otimes id_{\Gamma'^\circ})) = \\ op_9(T(D_1) \circ op_{10}(T(D_2)) = \end{gathered}$$

and that

$$op_{11}(\beta^{-1}(T(D_2) \Rightarrow id_A)) \circ op_{12}(\beta(T(D_1))) =$$
$$op_{11}(\beta^{-1}(T(D_2) \Rightarrow id_A)) \circ op_{11}(id_{\Delta^\circ} \otimes \beta(T(D_1))) =$$
$$op_{11}(\beta^{-1}(T(D_2) \Rightarrow id_A) \circ (id_{\Delta^\circ} \otimes \beta(T(D_1)))) =$$
$$op_{11}(id_A \circ \beta^{-1}(T(D_2) \Rightarrow id_A) \circ (id_{\Delta^\circ} \otimes \beta(T(D_1)))) =$$
$$op_{11}(\beta^{-1}((id_{\Delta^\circ} \Rightarrow id_A) \circ (T(D_2) \Rightarrow id_A) \circ \beta(T(D_1)))) =$$
$$op_{11}(\beta^{-1}((T(D_2) \Rightarrow id_A) \circ \beta(T(D_1)) \circ id_{\Gamma'^\circ})) =$$
$$op_{11}(\beta^{-1}(\beta(id_A \circ T(D_1) \circ (T(D_2) \otimes id_{\Gamma'^\circ})))) =$$
$$op_{11}(T(D_1) \circ (T(D_2) \otimes id_{\Gamma'^\circ})) =$$
$$op_{13}(T(D_1) \circ (T(D_2) \otimes id_{\Gamma'^\circ})) =$$
$$op_{13}(T(D_1)) \circ op_{13}(T(D_2) \otimes id_{\Gamma'^\circ}) =$$
$$op_{13}(T(D_1)) \circ op_{14}(T(D_2)).$$

We proceed with the table for permutative cut-elimination, it is as follows:

$Cut(\otimes L(D_1), D_2)$ $T(D_2) \circ op_1(T(D_1))$	$\otimes L(Cut(D_1, D_2))$ $T(D_2) \circ op_1(T(D_1))$
$Cut(\backslash L(D_1, D_2), D_3)$ $T(D_3) \circ op_2(T(D_2) \circ op_3(\beta^{-1}(T(D_1) \Rightarrow id_A)))$	$\backslash L(D_1, Cut(D_2, D_3))$ $T(D_3) \circ op_4(T(D_2)) \circ op_5(\beta^{-1}(T(D_1) \Rightarrow id_A))$
$Cut(/L(D_1, D_2), D_3)$ $T(D_3) \circ op_6(T(D_2) \circ op_7(\gamma^{-1}(id_A \Leftarrow T(D_1))))$	$/L(D_1, Cut(D_2, D_3))$ $T(D_3) \circ op_8(T(D_2)) \circ op_9(\gamma^{-1}(id_A \Leftarrow T(D_1)))$
$Cut(D_1, \otimes L(D_2))$ $T(D_2) \circ op_{10}(T(D_1))$	$\otimes L(Cut(D_1, D_2))$ $T(D_2) \circ op_{10}(T(D_1))$
$Cut(D_1, \backslash R(D_2))$ $\beta(T(D_2)) \circ op_{11}(T(D_1))$	$\backslash R(Cut(D_1, D_2))$ $\beta(T(D_2) \circ op_{12}(T(D_1)))$
$Cut(D_1, /R(D_2))$ $\gamma(T(D_2)) \circ op_{13}(T(D_1))$	$/R(Cut(D_1, D_2))$ $\gamma(T(D_2) \circ op_{14}(T(D_1)))$
$Cut(D_1, \otimes R(D_2, D_3))$ $(T(D_2) \otimes T(D_3)) \circ op_{15}(T(D_1))$	$\otimes R(D_2, Cut(D_1, D_3))$ $T(D_2) \otimes (T(D_3) \circ op_{16}(T(D_1)))$
$Cut(D_1, \otimes R(D_2, D_3))$ $(T(D_2) \otimes T(D_3)) \circ op_{17}(T(D_1))$	$\otimes R(Cut(D_1, D_2), D_3)$ $(T(D_2) \circ op_{18}(T(D_1))) \otimes T(D_3)$
$Cut(D_1, \backslash L(D_2, D_3))$ $T(D_3) \circ op_{19}(\beta^{-1}(T(D_2) \Rightarrow id_A)) \circ op_{20}(T(D_1))$	$\backslash L(D_2, Cut(D_1, D_3))$ $T(D_3) \circ op_{21}(T(D_1)) \circ op_{22}(\beta^{-1}(T(D_2) \Rightarrow id_A))$
$Cut(D_1, \backslash L(D_2, D_3))$ $T(D_3) \circ op_{23}(\beta^{-1}(T(D_2) \Rightarrow id_A)) \circ op_{24}(T(D_1))$	$\backslash L(Cut(D_1, D_2), D_3)$ $T(D_3) \circ op_{25}(\beta^{-1}((T(D_2) \circ op_{26}(T(D_1))) \Rightarrow id_A))$
$Cut(D_1, /L(D_2, D_3))$ $T(D_3) \circ op_{27}(\gamma^{-1}(id_A \Leftarrow T(D_2))) \circ op_{28}(T(D_1))$	$/L(D_2, Cut(D_1, D_3))$ $T(D_3) \circ op_{29}(T(D_1)) \circ op_{30}(\gamma^{-1}(id_A \Leftarrow T(D_2)))$
$Cut(D_1, /L(D_2, D_3))$ $T(D_3) \circ op_{31}(\gamma^{-1}(id_A \Leftarrow T(D_2))) \circ op_{32}(T(D_1))$	$/L(Cut(D_1, D_2), D_3)$ $T(D_3) \circ op_{33}(\gamma^{-1}(id_A \Leftarrow (T(D_2) \circ op_{34}(T(D_1)))))$

Now we only need to show that the effect of the different incarnations of op is essentially the same on the left and right hand side. We only need to show the cases of $\otimes$ rules and of $\backslash$ rules as the ones for $/$ are symmetric.
So for $\otimes$ we note that

$$\begin{gathered}(T(D_2) \otimes T(D_3)) \circ op_{15}(T(D_1)) = \\ (T(D_2) \otimes T(D_3)) \circ (id_{\Gamma^\circ} \otimes op_{16}(T(D_1))) = \\ (T(D_2) \circ id_{\Gamma^\circ}) \otimes (T(D_3) \circ op_{16}(T(D_1))) = \\ T(D_2) \otimes (T(D_3) \circ op_{16}(T(D_1)))\end{gathered}$$

and that

$$(T(D_2) \otimes T(D_3)) \circ op_{17}(T(D_1)) =$$
$$(T(D_2) \otimes T(D_3)) \circ (op_{18}(T(D_1)) \otimes id_{\Delta^\circ}) =$$
$$(T(D_2) \circ op_{18}(T(D_1))) \otimes (T(D_3) \circ id_{\Delta^\circ}) =$$
$$(T(D_2) \circ op_{18}(T(D_1))) \otimes T(D_3).$$

For $\setminus$ we have somewhat more cases to consider. Firstly, we observe that

$$T(D_3) \circ op_2(T(D_2) \circ op_3(\beta^{-1}(T(D_1) \Rightarrow id_A))) =$$
$$T(D_3) \circ op_2(T(D_2)) \circ op_2(op_3(\beta^{-1}(T(D_1) \Rightarrow id_A))) =$$
$$T(D_3) \circ op_4(T(D_2)) \circ op_5(\beta^{-1}(T(D_1) \Rightarrow id_A))$$

Secondly, we simply note that by $op_12(f) = id_B \otimes op_11(f)$ we have that

$$\beta(T(D_2)) \circ op_{11}(T(D_1)) =$$
$$\beta(T(D_2) \circ (id_B \otimes op_{11}(T(D_1)))) =$$
$$\beta(T(D_2) \circ op_{12}(T(D_1))).$$

Then we continue by observing (for the last two cases) that $op_{19}(f) \circ op_{20}(g)$ can be replaced by $op_{21}(g) \circ op_{22}(f)$ as the contexts the operators act upon are separate (i.e. the different operators enforce the domains and codomains of f and g to be of a certain kind, such that the order of applying these contexts does not matter).
For the very last case we need to consider we observe (by $op_{23} = op_{25}$ and $op_{24}(f) = op_{23}(op_{26}(f) \otimes id_{B \Rightarrow A})$) that

$$T(D_3) \circ op_{23}(\beta^{-1}(T(D_2) \Rightarrow id_A)) \circ op_{24}(T(D_1)) =$$
$$T(D_3) \circ op_{23}(\beta^{-1}(T(D_2) \Rightarrow id_A)) \circ op_{23}(op_{26}(T(D_1)) \otimes id_{B \Rightarrow A}) =$$
$$T(D_3) \circ op_{23}(\beta^{-1}(T(D_2) \Rightarrow id_A) \circ (op_{26}(T(D_1)) \otimes id_{B \Rightarrow A})) =$$
$$T(D_3) \circ op_{25}(id_A \circ \beta^{-1}(T(D_2) \Rightarrow id_A) \circ (op_{26}(T(D_1)) \otimes id_{B \Rightarrow A})) =$$
$$T(D_3) \circ op_{25}(\beta^{-1}((T(D_2) \circ op_{26}(T(D_1))) \Rightarrow id_A)).$$

□

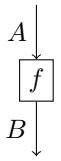
2.3.8 The Category of Proof Nets and Freeness

Now we have reached the high point of this chapter: we will show that proof nets define a bi-closed tensor category relative to a bi-closed tensor signature and show that it is a free bi-closed tensor category. Then, coherence follows by the definition of the free category. Recall the definition of a signature, and the inductive definition of proof nets. We define the category of proof nets over a signature Σ as follows:

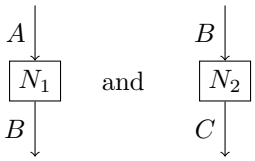
Definition 2.18. Let $\Sigma = (\Sigma_0, \Sigma_1, dom, cod)$ be a bi-closed tensor signature. The category of proof nets over Σ, denoted $\mathbf{PN}(\Sigma)$, is defined as follows:

- $Ob(\mathbf{PN}(\Sigma)) = \Sigma_0$,
- For every A in Σ_0, we define the identity morphism as

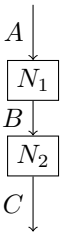
- For every f in Σ_1 with $dom(f) = A$ and $cod(f) = B$, we define the corresponding morphism as



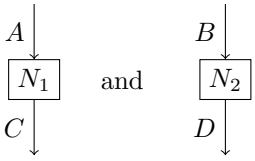
- We define composition of morphisms as vertical gluing, i.e. for morphisms



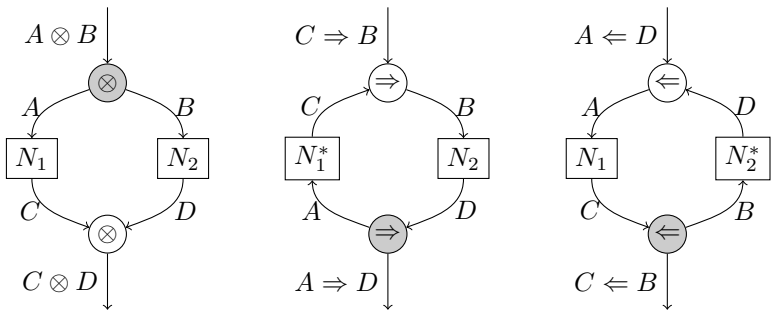
the following is their composition $N_2 \circ N_1$:



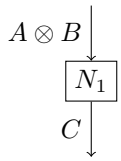
- For two morphisms



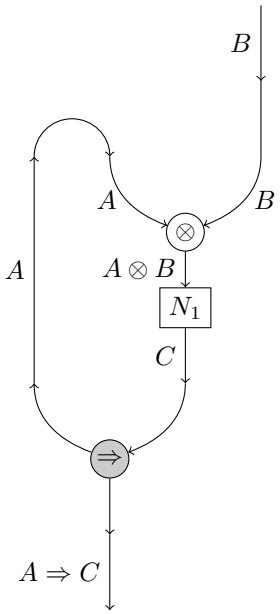
We define $N_1 \otimes N_2$, $N_1 \Rightarrow N_2$ and $N_1 \Leftarrow N_2$ as follows:



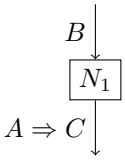
- Given a morphism



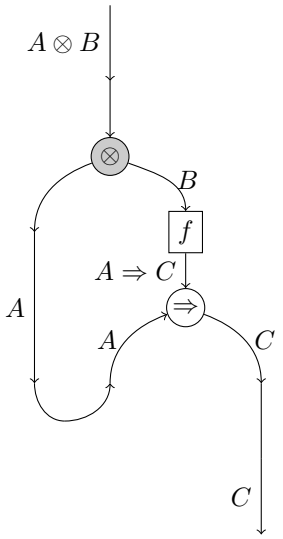
  we define $\beta(N_1)$ as



- Given a morphism



  we define $\beta^{-1}(N_1)$ as

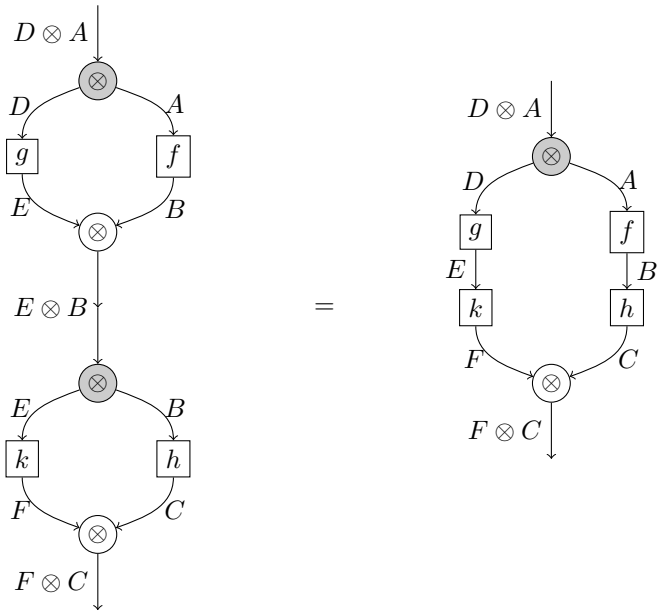


- We define γ and γ^{-1} similarly to β and β^{-1} but with the images mirrored horizontally.
- All the equations for proof nets hold.

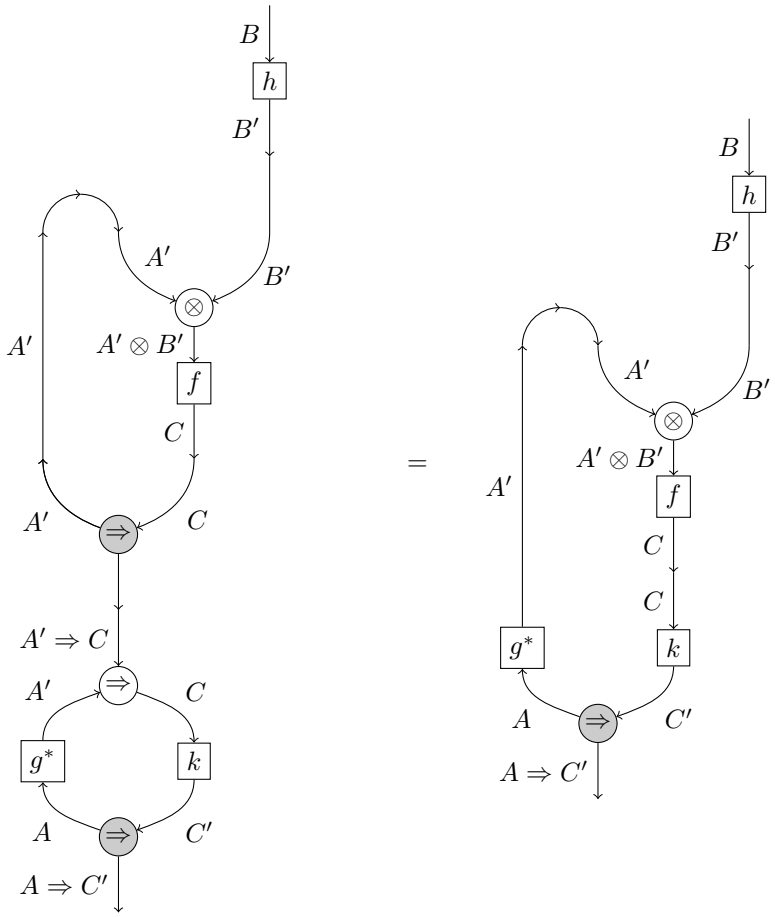
It is now a pleasant exercise to show that $\mathbf{PN}(\Sigma)$ is a bi-closed tensor category:

Proposition 2.1. *For any bi-closed tensor signature Σ, $\mathbf{PN}(\Sigma)$ is a bi-closed tensor category.*

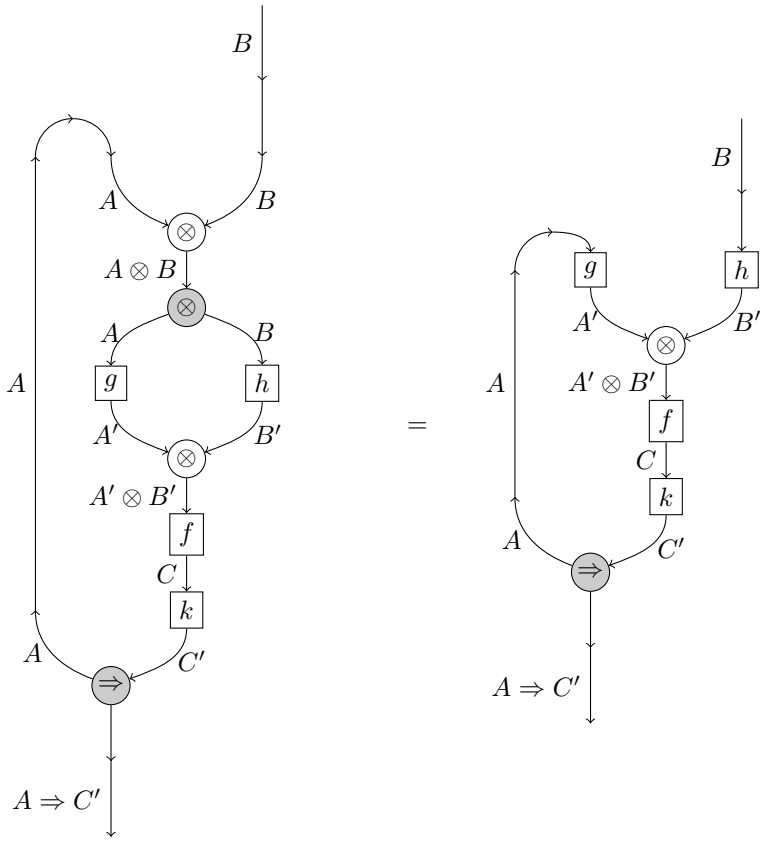
Proof. The basic categorical axioms are trivially satisfied: the associativity of gluing gives associativity of composition, and for any morphism $N_1 : A \to B$ we have that $id_B \circ N_1$ is the result of gluing an extra piece of wire on the bottom and $N_1 \circ id_A$ is the result of gluing an extra piece of wire on the top, which is just the same as the original morphism. $id_A \otimes id_B = id_{A \otimes B}$ is also trivially satisfied by the identy cut equation for $\otimes$ (similarly for $\Rightarrow$ and $\Leftarrow$. Bifunctioriality of $\otimes$, is also satisfied because $(k \otimes h) \circ (g \otimes f) = (k \circ g) \otimes (h \circ f)$ translates to



which definitely is a valid equation by virtue of the general cut equation. Bifunctoriality of $\Rightarrow$ and $\Leftarrow$ follows similarly. Isomorphicity of β and γ is easily verified using the snake equations. Finally, we need to show naturality of β and γ. We show the naturality of β as naturality for γ follows similarly. For $(g \Rightarrow k) \circ ((\beta(f)) \circ h)$ we have



whereas for $\beta(k \circ (f \circ (g \otimes h)))$ we have



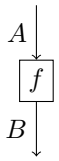
and these nets are obviously equal, given that we can bend around g to g^* and vice versa. □

Not only is $\mathbf{PN}(\Sigma)$ a bi-closed tensor category, it is moreover the free one over Σ:

Theorem 2.3. $\mathbf{PN}(\Sigma)$ *is the free bi-closed tensor category over* Σ*.*

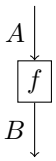
Proof. We need to give an interpretation $i : \Sigma \to \mathbf{PN}(\Sigma)$ and for any bi-closed tensor category $\mathbf{D}$ and bi-closed tensor interpretation $j : \Sigma \to \mathbf{D}$ give a unique bi-closed tensor functor $F : \mathbf{PN}(\Sigma) \to \mathbf{D}$ such that $j = F \circ i$. Take $i = \langle i_0, i_1 \rangle$ where

- i_0 is the identity on Σ_0,
- i_1 sends $f : A \to B$ in Σ_1 to

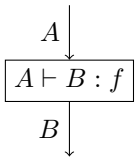


Now let $\mathbf{D}$ be any bi-closed tensor category and let $j : \Sigma \to \mathbf{D}$ be an arbitrary interpretation. We define $F : \mathbf{PN}(\Sigma) \to \mathbf{D}$ as follows:

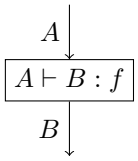
- On objects, we define $F(A) = j_0(A)$.
- On morphisms, we take $F(N : A \to B) = \hat{j_1} \circ T \circ S$ where
  - S is the sequentialization translation that sends



    to



  - T is the sequent proof translation into categorical morphisms that sends



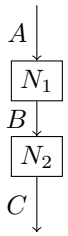
    to

$$f : A \to B$$

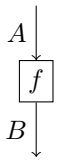
  - $\hat{j_1}$ sends all $f : A \to B$ in Σ_1 to $j_1(f)$ and acts as the identity on everything else.

We now need to show that F is a bi-closed tensor functor, that $j = F \circ i$ and that F is unique. To see that F is a functor, observe that

sequentializes to $A \vdash A : Id(A)$ which then translates into id_A (and then is preserved by $\hat{j_1}$), and next observe that



sequentializes to $A \vdash C : Cut(D_1, D_2)$ which then translates into $\hat{j_1}(g \circ f) = \hat{j_1}(g) \circ \hat{j_1}(f)$ given that D_1 is given by f and D_2 is given by g.
To see that this functor is a bi-closed tensor functor, simply take $\varphi_{A,B} = id_{A \otimes B}$, $\chi_{A,B} = id_{A \Rightarrow B}$, $\xi_{A,B} = id_{B \Leftarrow A}$ to see that F is in fact a *strict* bi-closed tensor functor! To see that $j = F \circ i$, note that on objects we have $F = j_0$ and $i = id$ so we get $j_0 = j_0 \circ id$. On morphism variables, we have $F = \hat{j_1} \circ T \circ S$, so we need to show $j_1 = \hat{j_1} \circ T \circ S \circ i_1$. So, consider any $f : A \to B$ in Σ_1. We have that i_1 sends f to



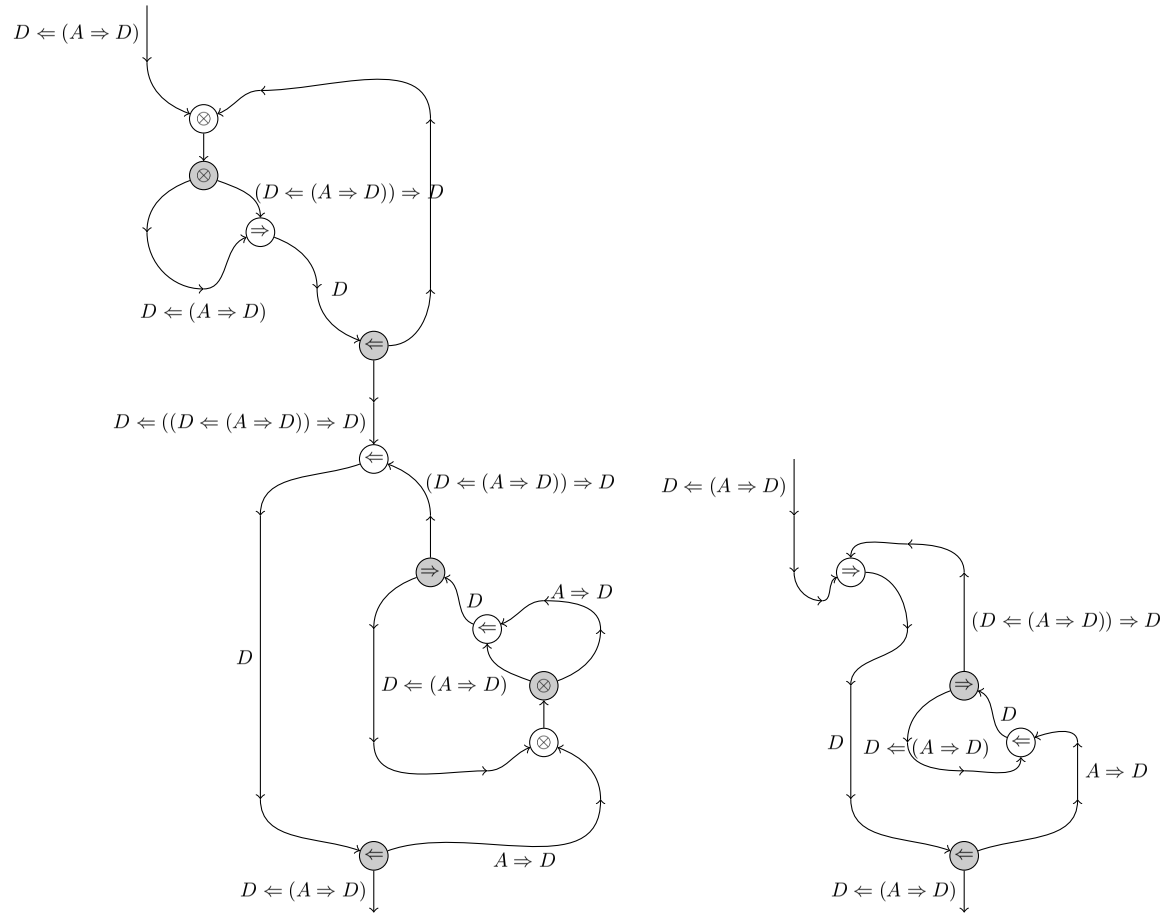
which is mapped by S to $A \vdash B : f$, which then is mapped by T simply to $f : A \to B$, where finally, $\hat{j_1}$ sends this to $j_1(f) : j_0(A) \to j_0(B)$.
Now we need to show that F is unique. So, let $G : \mathbf{PN}(\Sigma) \to \mathbf{D}$ be a bi-closed tensor functor such that $j = G \circ i$. As $i_0 = id$, we must have on objects that $G = j_0$. On morphism variables, note that $T \circ S$ is inverse to i_1 such that we get

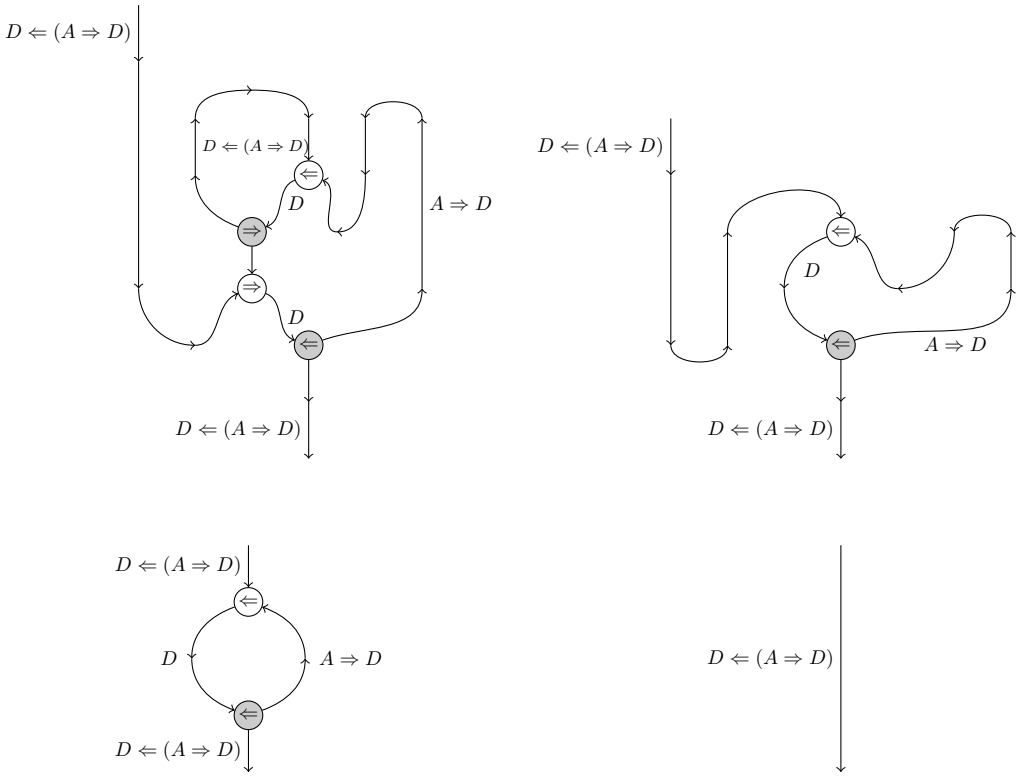
$$G = G \circ i_1 \circ T \circ S = F \circ i_1 \circ T \circ S = F$$

□

2.3.9 Illustration

We can now illustrate the use of our graphical language by stating Proposition 1.3 graphically. For instance, for the second triangle diagram, we get the graphical representation of $(id_D \Leftarrow (\beta(\gamma^{-1}(id_{D \Leftarrow (A \Rightarrow D)})))) \circ \gamma(\beta^{-1}(id_{(D \Leftarrow (A \Rightarrow D)) \Rightarrow D}))$ and its reduction to $id_{D \Leftarrow (A \Rightarrow D)}$, which is


$D \Leftarrow (A \Rightarrow D)$
$(D \Leftarrow (A \Rightarrow D)) \Rightarrow D$
D
$D \Leftarrow (A \Rightarrow D)$
$D \Leftarrow ((D \Leftarrow (A \Rightarrow D)) \Rightarrow D)$
$(D \Leftarrow (A \Rightarrow D)) \Rightarrow D$
D
$A \Rightarrow D$
D
$D \Leftarrow (A \Rightarrow D)$
$A \Rightarrow D$
$D \Leftarrow (A \Rightarrow D)$
$D \Leftarrow (A \Rightarrow D)$
$(D \Leftarrow (A \Rightarrow D)) \Rightarrow D$
D
D
$D \Leftarrow (A \Rightarrow D)$
$A \Rightarrow D$
$D \Leftarrow (A \Rightarrow D)$



2.4 Discussion

For the final section of this chapter, we would like to take up a discussion about the relation between string diagrams for non-associative and associative categories. But first we will discuss a possible concern about the developed proof net language.

The following question might be raised about the proof net language developed above: *given the presence of global graphical yanking in the language, how can one ensure that the proof nets do not reduce to the compact case?* We wish to show, however, that this is not the case. The intuition is that graphical yanking is necessary in our graphical language only to ensure coherence but has no special meaning; it is merely a syntactic construct used to make the proof nets themselves prettier and easier to interpret.

Because of the way we choose to draw our nets, specifically monotonicity nets for $\Rightarrow$ and $\Leftarrow$ we choose to let information flow upwards within a diagram. Because of this property and the way we defined sequentialization, in some cases snake patterns occur (for instance in the cases where one wants to verify that β and γ are isomorphisms). Thus, we require graphical yanking to ensure that our graphical language is sound and complete.

Because yanking in the graphical language for compact closed categories corresponds to the yanking equations of those categories, it is of course a bit strange that we require them here, while

the corresponding category is by no means a compact one. But keep in mind that we are considering *non-associative* systems here: compact closed categories are by definition monoidal and thus have associative structure, meaning that the yanking equations make sense as they require use of the associativity morphisms. But when we drop associativity, the yanking equations do not make sense anymore, and hence graphical yanking would have any special meaning to it. We illustrate this with the case of pregroups, which are instantiations of non-commutative compact closed categories, or autonomous categories. A pregroup is an ordered monoid where each element a has left and right adjoints a^l and a^r such that

$$a^l \cdot a \leq 1 \leq a \cdot a^l$$
$$a \cdot a^r \leq 1 \leq a^r \cdot a$$

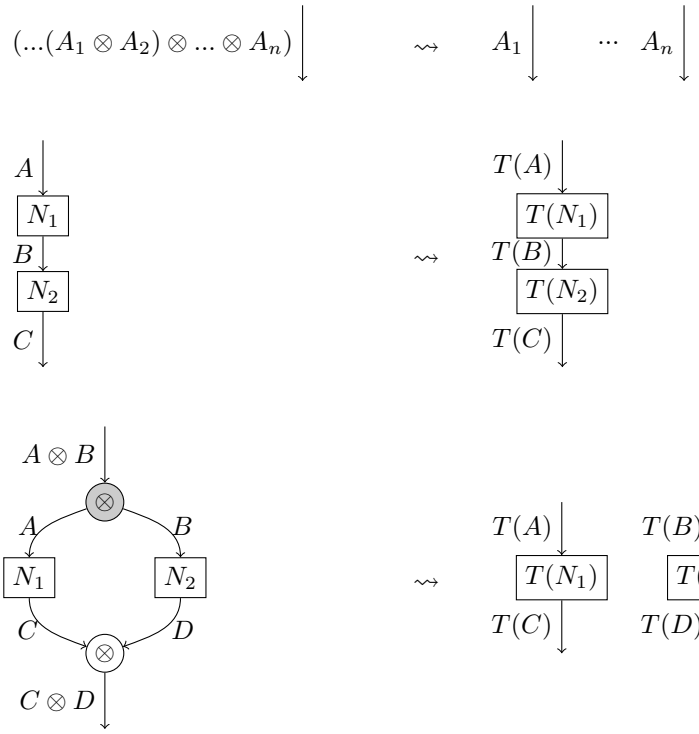
and now yanking follows because of the following chain:

$$a \leq 1 \cdot a \leq (a \cdot a^l) \cdot a = a \cdot (a^l \cdot a) \leq a \cdot 1 \leq a$$

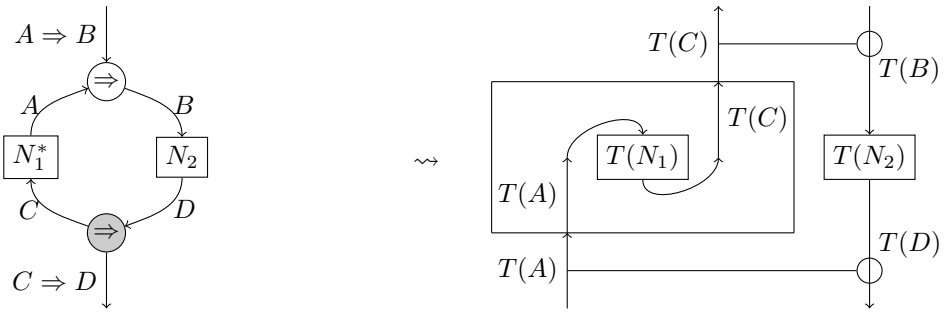
But this only follows by virtue of a pregroup having associative structure. If one were to define non-associative autonomous categories one could not admit the yanking equations and therefore graphical yanking is not the same there as it is in regular autonomous categories.

We will now turn our attention to the relation between graphical languages for the non-associative and the associative setting. Obviously, every non-associative tensor category can be embedded in an associative tensor category, whether it be closed or not. So we may ask ourselves how this would go in the setting of string diagrams. We will first consider the translation of proof nets for non-closed tensor categories into string diagrams for associative tensor categories, and then conclude with some remarks on the possibility of translating proof nets for bi-closed tensor categories into the clasp diagrams of Baez and Stay (2011).

The developed proof net calculus for bi-closed tensor categories can be reduced to a coherent one for non-associative tensor categories by getting rid of the $\Rightarrow$ and $\Leftarrow$ links and dropping the graphical yanking equations. In this case, all proof nets and their subgraphs will be drawn such that all information flows downward. More precisely, the inductive definition for these nets contains only of the identity proof net, composition, and monotonicity for $\otimes$. It then becomes clear how we would translate proof nets to string diagrams for associative tensor categories in the following translation:



When we turn our attention to bi-closed tensor categories, however, things become more complicated. Considering the Došen axiomatization, we would need to extend the previous translation with the translation of monotonicity for $\Rightarrow$ and $\Leftarrow$ and we would need to give translations for application and co-application. The translations for monotonicity would not be that complicated if one were to admit graphical yanking in the clasp diagrams (note that we only give the translation for $\Rightarrow$ as the one for $\Leftarrow$ is symmetric):



But as noted in the section on the clasp diagrams, we would be in trouble when A is of the form $E \otimes F$. This problem arises again when we consider translating co-application, as it would need to translate to co-evaluation, where we have the same problem of a clasp having to attach to either one of the wires of $A \otimes B$.

In conclusion, there is more work to be done in order to consider the clasp diagrams as a coherent graphical language for monoidal bi-closed categories. This is the reason why translating our proof nets into clasp diagrams does not work smoothly. Another aspect that deserves attention is the precise relationship between the graphical calculus we have developed in this chapter and the proof nets developed by Moot and Puite (2002).

In the next chapter we shall consider "categorifying" the different incarnations of the Lambek Calculus.

Chapter 3

Syntax

In this chapter we consider the different incarnations of the Lambek Calculus: the non-associative Lambek Calculus **NL**, the associative Lambek Calculus **L**, and the variants with units **UNL** and **UL**. Guideluines are given to turn each of the calculi into a category, after which we show that the resulting categories each take their place in the Curry-Howard-Lambek correspondence. We conclude with the definitions of categorial and categorical grammars as a means of incorporating these type logics in a compositional distributional model of meaning.

Lambek's *syntactic calculus*, which we denote by **L**, was a system introduced in (Lambek, 1958) in order to distinguish, not just in formal languages but also in natural language, *sentences* from *non-sentences*. Three years later, Lambek published an article (Lambek, 1961) in which he dropped the associativity rules, resulting in the non-associative Lambek Calculus, here denoted **NL**. We furthermore distinguish between the systems **UL** and **UNL**, where a unit object is added. All these systems are *substructural logics*, systems that are obtained by dropping some of the structural rules of classical logic, such as weakening, contraction and exchange.

We present these systems as *deductive systems*, a term coming from Lambek (1988) as a method of presenting these logics in a form particularly suitable for a categorical interpretation. We will proceed as follows: first we define several variants of the Lambek Calculus, and secondly we will show what is needed to turn each of these calculi into a category. Finally, we turn our attention to the Curry-Howard-Lambek correspondence applied to these categorical Lambek Calculi, and close the chapter with the definition of *grammars* based on Lambek Calculi, giving a system for characterizing sets of *sentences*.

3.1 Lambek Calculi, Categorically

We start out by devoting our attention to the different types of Lambek Calculi available, and the restrictions one must add to "categorify" these calculi.

3.1.1 A Landscape of Calculi

The basis of any deductive system are formulas, which we will call *types*. These are generated by a set of *basic types* and a set of *connectives*. The regular Lambek Calculus deploys a tensor connective $\otimes$ and left and right implications $\backslash$ and $/$:

Let T be a set of *basic types*. Let $C \subseteq \{\otimes, \backslash, /\}$. Then the set of free types $F(T, C)$ of T over C is the smallest set satisfying the following:

1. $T \subseteq F(T, C)$,
2. If $A, B \in F(T, C)$ then $A * B \in F(T, C)$ for $* \in C$.

From any set of types, one may define a deductive system in the style of Lambek (1988):

Definition 3.1. A deductive system over T consists of types and arrows and domain/codomain mappings from arrows to types to indicate that a given arrow f is a proof of $cod(f)$ from $dom(f)$. We simply write $f : A \to B$ for f a proof of B from A. Moreover we have the following requirements:

1. For every type $A \in T$, there is given an identity arrow $1_A : A \to A$,
2. For every two proofs $f : A \to B$ and $g : B \to C$ there is a composite proof $g \circ f : A \to C$.

This scheme is nicely depicted as the following inference system:

$$\frac{}{1_A : A \to A}\ Ax \qquad \frac{f : A \to B \quad g : B \to C}{g \circ f : A \to C}\ T$$

From now on, we represent any extension of a deductive system directly as a system of inference. We then want to encode the behaviour of the different Lambek connectives in separate deductive systems. Hence, we consider subsets $\{\otimes, \backslash\}$, $\{\otimes, /\}$, and $\{\otimes, \backslash, /\}$ when defining left, right and regular Lambek Calculi:

Definition 3.2. The (non-associative,non-unitary) left Lambek Calculus $\mathbf{NL}^l$ over T is given by its types being the elements of $F(T, \{\otimes, \backslash\})$ and the proofs generated by the following inference system:

$$\frac{}{1_A : A \to A}\ Ax \qquad \frac{f : A \to B \quad g : B \to C}{g \circ f : A \to C}\ T$$

$$\frac{f : A \otimes B \to C}{\lhd f : B \to A \backslash C}\ R2 \qquad \frac{g : B \to A \backslash C}{\lhd^{-1} g : A \otimes B \to C}\ R2'$$

$$\frac{f : A \to C \quad g : B \to D}{f \otimes g : A \otimes B \to C \otimes D}\ M_\otimes \qquad \frac{f : A \to C \quad g : B \to D}{f \backslash g : C \backslash B \to A \backslash D}\ M_\backslash$$

Definition 3.3. The (non-associative, non-unitary) right Lambek Calculus $\mathbf{NL}^r$ over T is given by its types being the elements of $F(T, \{\otimes, /\})$ and the proofs are generated by the following inference system:

$$\frac{}{1_A : A \to A}\ Ax \qquad \frac{f : A \to B \quad g : B \to C}{g \circ f : A \to C}\ T$$

$$\frac{f : A \otimes B \to C}{\rhd f : A \to C / B}\ R1 \qquad \frac{g : A \to C / B}{\rhd^{-1} g : A \otimes B \to C}\ R1'$$

$$\frac{f : A \to C \quad g : B \to D}{f \otimes g : A \otimes B \to C \otimes D}\ M_\otimes \qquad \frac{f : A \to C \quad g : B \to D}{g / f : B / C \to D / A}\ M_/$$

We now want to define the non-associative, non-unitary Lambek Calculus **NL**. One might expect it is obtained by merging $\mathbf{NL}^l$ and $\mathbf{NL}^r$. However, the monotonicity rules become *derived* rules of inference in the full system, so we have the following:

Definition 3.4. The (non-associative, non-unitary) Lambek Calculus **NL** over T is given by the types in $F(T, \{\otimes, \backslash, /\})$ and the proofs generated by the following (labelled) inference system:

$$\frac{}{1_A : A \to A}\ Ax \qquad \frac{f : A \to B \quad g : B \to C}{g \circ f : A \to C}\ T$$

$$\frac{f : A \otimes B \to C}{\rhd f : A \to C / B}\ R1 \qquad \frac{f : A \otimes B \to C}{\lhd f : B \to A \backslash C}\ R2$$

$$\frac{g : A \to C / B}{\rhd^{-1} g : A \otimes B \to C}\ R1' \qquad \frac{g : B \to A \backslash C}{\lhd^{-1} g : A \otimes B \to C}\ R2'$$

It can easily be shown that this system gives us the following derived rules of inference, the *monotonicity* rules:

$$\frac{f : A \to C \quad g : B \to D}{f \otimes g : A \otimes B \to C \otimes D}\ M_{\otimes}$$

$$\frac{f : A \to C \quad g : B \to D}{g/f : B/C \to D/A}\ M_{/}$$

$$\frac{f : A \to C \quad g : B \to D}{f\backslash g : C\backslash B \to A\backslash D}\ M_{\backslash}$$

where we have that

$$f \otimes g := \triangleright^{-1}((\triangleright \triangleleft^{-1} ((\triangleleft 1_{C\otimes D}) \circ g)) \circ f)$$
$$g/f := \triangleright(g \circ (\triangleleft^{-1}((\triangleleft \triangleright^{-1} 1_{B\backslash C}) \circ f)))$$
$$f\backslash g := \triangleleft(g \circ (\triangleright^{-1}((\triangleright \triangleleft^{-1} 1_{C\backslash B}) \circ f)))$$

We furthermore have the following proofs, usually called *application* and *co-application*:

$$\triangleleft^{-1}1_{A\backslash B} : A \otimes (A\backslash B) \to B \quad \triangleleft 1_{A\otimes B} : B \to A\backslash(A \otimes B)$$
$$\triangleright^{-1}1_{B/A} : (B/A) \otimes A \to B \quad \triangleright 1_{B\otimes A} : B \to (B \otimes A)/A$$

The nice thing about deductive systems is that one can add extra properties, simply as rules of inference. For example, to obtain the associative variants of the calculi discussed, one adds to a system of choice the following additional rules for every type A, B, C of the system under discussion:

$$\frac{}{a_{A,B,C} : (A \otimes B) \otimes C \to A \otimes (B \otimes C)}\ Ass$$

$$\frac{}{a^{-1}_{A,B,C} : A \otimes (B \otimes C) \to (A \otimes B) \otimes C}\ Ass'$$

The resulting systems are denoted $\mathbf{L}^l$, $\mathbf{L}^r$ and simply $\mathbf{L}$, respectively. To obtain the unitary variant of the associative Lambek Calculi, one simply adds a distinguished unit type I to T and the following additional axioms for every type A:

$$\frac{}{l_A : I \otimes A \to A}\ L \quad \frac{}{l^{-1}_A : A \to I \otimes A}\ L'$$

$$\frac{}{r_A : A \otimes I \to A}\ R \quad \frac{}{r^{-1}_A : A \to A \otimes I}\ R'$$

The resulting systems are denoted $\mathbf{UL}^l$, $\mathbf{UL}^r$ and $\mathbf{UL}$, respectively.

3.1.2 Going Categorical

To obtain a category for each of the Lambek Calculi noted, we need to define an appropriate equivalence relation on proofs in such calculi. We define an equivalence relation $\equiv$ on proofs. Clearly, we require $\equiv$ to be reflexive, symmetric and transitive. Furthermore, we require the following additional equivalences:

1. Substitution properties: If $f \equiv g$ then $*f \equiv *g$ where $* \in \{\triangleright, \triangleright^{-1}, \triangleleft, \triangleleft^{-1}\}$, and if $f \equiv g$ and $k \equiv h$ then $k \circ f = h \circ g$. A resulting property is that if $f \equiv g$ and $k \equiv h$ then $f * k \equiv g * h$ where $* \in \{\otimes, /, \backslash\}$.

2. Categorical axioms: this amounts to $f \circ (g \circ h) \equiv (f \circ g) \circ h$ and $f \circ 1_A \equiv f \equiv 1_B \circ f$ where $f : A \to B$.

3. Residuation is isomorphic: this amounts to $*^{-1} * f \equiv f$, $* *^{-1} g \equiv g$ for $* \in \{\triangleright, \triangleleft\}$.

4. Bifunctoriality of $\otimes, /, \backslash$. One can easily show that $1_A * 1_B \equiv 1_{A*B}$ where $* \in \{\otimes, /, \backslash\}$ but the following equivalences are defined:

$$(f \otimes g) \circ (k \otimes h) \equiv (f \circ k) \otimes (g \circ h)$$
$$(f/g) \circ (k/h) \equiv (f \circ k)/(h \circ g)$$
$$(f\backslash g) \circ (k\backslash h) \equiv (k \circ f)\backslash(g \circ h)$$

5. Naturality of residuation. This means that for $f : A' \otimes B' \to C$ and $g : A \to A'$, $h : B \to B'$, $k : C \to C'$, we have that

$$(k/h) \circ ((\triangleright f) \circ g) \equiv \triangleright(k \circ (f \circ (g \otimes h))) \text{ and}$$
$$(g\backslash k) \circ ((\triangleleft f) \circ h) \equiv \triangleleft(k \circ (f \circ (g \otimes h))).$$

6. For $\mathbf{L}^{(l/r)}$, one adds the following requirements:

   (a) Associativity is isomorphic, i.e. $a^{-1}_{A,B,C} \circ a_{A,B,C} \equiv 1_{A\otimes(B\otimes C)}$ and $a_{A,B,C} \circ a^{-1}_{A,B,C} \equiv 1_{(A\otimes B)\otimes C}$ for every A, B, C,

   (b) Associativity is natural, i.e. $(f \otimes (g \otimes h)) \circ a_{A,B,C} \equiv a_{A',B',C'} \circ ((f \otimes g) \otimes h)$ for $f : A \to A'$,$g : B \to B'$, $h : C \to C'$.

7. For $\mathbf{UL}^{(l/r)}$, one adds the following requirements:

   (a) Unit deletion is isomorphic, i.e. $l^{-1}_A \circ l_A \equiv 1_{I\otimes A}$, $l_A \circ l^{-1}_A \equiv 1_A$, $r^{-1}_A \circ r_A \equiv 1_{A\otimes I}$ and $r_A \circ r^{-1}_A \equiv 1_A$ for every A,

   (b) Unit deletion is natural, i.e. for every $f : A \to A'$ we have that $f \circ l_A \equiv l_{A'} \circ (1_I \otimes f)$ and $f \circ r_A \equiv r_{A'} \circ (f \otimes 1_I)$.

Now one can show easily that any Lambek Calculi forms a certain kind of closed category if we take its objects to be the freely generated types and its morphisms to be equivalence classes of proofs. For instance, we get the following proposition:

Proposition 3.1. *Let $C(\boldsymbol{NL})$ denote the non-associative Lambek Calculus over a set of basic types T together with the equivalence relation defined above. Then $C(\boldsymbol{NL})$ is a bi-closed tensor category.*

Similarly, $C(\mathbf{NL}^l)$ forms a left closed tensor category, $C(\mathbf{NL}^r)$ forms a right closed category, and once we add associativity and a unit, we obtain a left/right/bi-closed monoidal category. In the next section, we will consider categories having as objects these calculi, and consider the equivalences that come out.

3.2 Symmetry and Equivalence

In this section, we show several equivalence results — i.e. equivalences of categories — between Categorical Lambek Calculi.

Consider two unitary Lambek Calculi $\mathbf{UL}_T$ and $\mathbf{UL}_{T'}$. We define a *translation* of calculi as a map t that sends T to $F(T', \{\otimes, /, \backslash\})$ and sends proofs to proofs such that the following holds:

- t strictly preserves the type operations, i.e. $t(1) = 1$, $t(A \otimes B) = t(A) \otimes t(B)$, $t(A\backslash B) = t(A)\backslash t(B)$, $t(B/A) = t(B)/t(A)$,
- t preserves the structural axioms, i.e. $t(a^{(-1)}_{A,B,C}) = a^{(-1)}_{t(A),t(B),t(C)}$, $t(l^{(-1)}_A) = l^{(-1)}_{t(A)}$, $t(r^{(-1)}_A) = r^{(-1)}_{t(A)}$.
- t preserves the equivalence relation on proofs, i.e. if $f \equiv g$ in $\mathbf{UL}_T$ then $t(f) \equiv t(g)$ in $\mathbf{UL}_{T'}$.

It is not hard to determine that we can form a category, henceforth denoted by **cUL**, by taking as objects unitary Lambek Calculi and as morphisms translations between such calculi. Let $\mathbf{MBCC}_{st}$ denote the category consisting of monoidal bi-closed categories as objects and strict monoidal bi-closed functors as morphisms. As every (categorified) unitary Lambek Calculus is already a monoidal bi-closed category and as every translation is a strict monoidal bi-closed functor, we take one part of the equivalence to be the identity functor Id. We continue to define the unitary Lambek Calculus generated by a monoidal bi-closed category. So, the unitary Lambek Calculus generated by a monoidal bi-closed category $\mathbf{C}$ is precisely $\mathbf{UL}_{Ob(\mathbf{C})}$. That is,

- The set of types is precisely $Ob(\mathbf{C})$ where $\otimes$ is interpreted as $\otimes$, $\Rightarrow$ is interpreted as $\backslash$ and $\Leftarrow$ is interpreted as $/$,
- Supposing that the set of types are obtained from $Ob(\mathbf{C})$ by some map i, morphisms in $\mathbf{C}$ are mapped to proofs by a map j in the following manner:
  1. $j(\alpha_{A,B,C}) = a_{i(A),i(B),i(C)}$,
  2. $j(\lambda_A) = l_{i(A)}$,
  3. $j(\rho_A) = r_{i(A)}$,
  4. $j(\beta(f)) = \triangleleft j(f)$,
  5. $j(\gamma(g)) = \triangleright j(g)$,
  6. $j(id_A) = 1_{i(A)}$
  7. Whenever there is an additional map $f : A \to B$ in $\mathbf{C}$, there is a proof $f : i(A) \to i(B)$.
  8. $j(g \circ f) = j(g) \circ j(f)$

It is not hard to see that equality of morphisms in $\mathbf{C}$ becomes equivalence of proofs in $UL_{Ob(\mathbf{C})}$. For instance, as we have $id_A \circ f = f$ in $\mathbf{C}$, we have $j(id_A \circ f) = 1_{i(A)} \circ j(f) \equiv j(f)$.

Now, let $F : \mathbf{C} \to \mathbf{D}$ be a strict monoidal bi-closed functor between two monoidal bi-closed categories. Then we straightforwardly define the translation t between $mathbfUL_{Ob(\mathbf{C})}$ and $\mathbf{UL}_{Ob(\mathbf{D})}$ to be F. As F is strict, it preserves the type operations on the nose, as well as the structural axioms and the equivalence relation on proofs. One can apply the correspondence to $\mathbf{cNL}^{(l/r)}, \mathbf{cL}^{(l/r)}, \mathbf{cUL}^{(l/r)}$ so given that we have the following denotations for categories of left/right/bi-closed tensor/monoidal categories with corresponding strict functors

- $(\mathbf{A}/\mathbf{M})\mathbf{LCC}_{st}$ denotes left closed (associative tensor/monoidal) categories and strict left closed functors,
- $(\mathbf{A}/\mathbf{M})\mathbf{RCC}_{st}$ denotes right closed (associative tensor/monoidal) categories and strict right closed functors,
- $(\mathbf{A}/\mathbf{M})\mathbf{BCC}_{st}$ denotes bi-closed (associative tensor/monoidal) categories and strict bi-closed functors.

we get the following propositions:

Proposition 3.2. $\mathbf{cNL}^{(l/r)} \cong (\mathbf{L}/\mathbf{R}/\mathbf{B})\mathbf{CC}_{st}$.

Proposition 3.3. $\mathbf{cL}^{(l/r)} \cong \mathbf{A}(\mathbf{L}/\mathbf{R}/\mathbf{B})\mathbf{CC}_{st}$.

Proposition 3.4. $\mathbf{cUL}^{(l/r)} \cong \mathbf{M}(\mathbf{L}/\mathbf{R}/\mathbf{B})\mathbf{CC}_{st}$.

So we see that various kinds of deductive systems are closely related to categories via an appropriate notion of proof equivalence. In the next section we will employ Lambek Calculi in order to construct *typelogical grammars*.

3.3 Grammars

In this section we define *Lambek grammars* as a tool for characterizing sets of strings or — in other words — distinguishing between *sentences* and *non-sentences*. We will then use these grammars in the next chapter when we formally define compositional distributional meaning models.

3.3.1 Categorial Grammar

The intuition behind a typelogical or *categorial* grammar is that we associate types with words and use a logic, in this case a Lambek Calculus, govern the principles by which certain words may be grammatically composed to form larger expressions. The basis of such a grammar is the *dictionary* associating types with words:

Definition 3.5. Let Σ be an alphabet, i.e. a finite, non-empty set of basic words, and let L be a Lambek Calculus of choice (with or without associativity and units) having as its set of types $F(T, C)$. Then a dictionary over Σ and L is a relation $\delta \subseteq \Sigma \times F(T, C)$.

The ultimate goal of a Lambek Grammar is to distinguish between what are grammatical sequences of words, and what are not. Hence, one must have a *goal type* in order to distinguish what kind of proofs in the logic will give a grammatical sentence and what kind of proofs will not. The definition of a Lambek Grammar is thus as follows:

Definition 3.6. A Lambek Grammar over a Lambek Calculus L is a triple (Σ, δ, S) where

- Σ is an alphabet,
- δ is a dictionary over Σ and L,
- $S \in F(T, C)$ is a distinguished goal type.

Grammaticality then comes out of these grammars nearly automatically: a sequence of words is a sentence when its corresponding sequence of types can be used to derive the goal type:

Definition 3.7. Let (Σ, δ, S) be a Lambek Grammar over L. A sequence of words $w_1 \cdot ... \cdot w_n \in W^+$ is a sentence if and only if there exists a sequence $W_1 \otimes ... \otimes W_n$ (where each $W_i \in \delta(w_i)$) such that there exists a proof in L of $W_1 \otimes ... \otimes W_n \to S$.

Because we are in general interested in the kind of patterns that are recognizable by grammars, we define the *language* of a grammar and the class of languages generated by a calculus:

Definition 3.8. Let $G = (\Sigma, \delta, S)$ be a Lambek Grammar over L. The language of G, denoted $\mathfrak{L}(G)$, is defined as the set of all sentences $w \in W^+$.

Definition 3.9. Let L be a Lambek Calculus of choice. The class of languages generated by L, denoted $\mathfrak{L}(L)$, is defined as the class of languages generated by any Lambek Grammar over L.

It turns out that the *generative capacity* of any Lambek Calculus, i.e. the kind of patterns that are recognizable, coincide with the well-known class of context-free languages. For the original, associative system **L**, these results come from Pentus (1993). For the non-associative variant **NL**, the result was obtained by Kandulski (1988). The versions with units were seen to correspond to context-free grammar by Buliński (2009). So, we get the following theorem:

Theorem 3.1. *The class of languages generated by any Lambek Calculus L coincides with the context-free languages.*

It is now interesting to see how categorial grammar can be "lifted" to become categorical grammar.

3.3.2 Categorical Grammar

Given the equivalence between categorical Lambek Calculi and closed categories, it is easy to define categorical grammar: just take the definitions of categorial grammar and replace the calculus L by its categorified version $C(L)$. The test for sentencehood then boils down to checking whether $Hom(W_1 \otimes ... \otimes W_n, S) \neq \emptyset$.

Obviously, the generative capacity for these categorical grammars will be the same as for categorial grammar because we have only identified proofs in the process of categorification, hence the accepted strings for a Lambek Grammar and its categorical variant will be the same.

However, these categorical versions of Lambek Grammars will be of importance once we define compositional distributional models of meaning, in the next chapter.

Chapter 4

Semantics

In this chapter, we define *basic compositional distributional models of meaning*. We will first devote attention to a simplified, non-intentional version of Montague semantics and show how these are subsumed by vector space models of meaning. We continue to show that finite-dimensional vector spaces are instantiations of compact closed categories, in turn a special case of monoidal bi-closed categories. We then define *semantic interpretations* and compositional distributional models of meaning based on the different Lambek Calculi considered so far. We then give directions on how one can *induce* such models from large data.

Categorial grammar automatically comes equipped, via the Curry-Howard isomorphism, with a syntax-semantics interface that maps proofs in a Lambek Grammar to programs in the simply typed lambda calculus. The idea is that the dictionary of a Lambek Grammar is extended to include for every word a semantic function that adheres to the type of the word under the semantic interpretation of types. The given mapping ensures *compositionality* of the semantics in question, but requires a predefined semantic lexicon. Here we will lift compositionality to the categorical level by relying on a functorial passage from the category of syntax to the category of semantics instead.

4.1 From Montague Semantics to Vector Space Semantics

4.1.1 Montague-style models

We start out with the definition of semantic models as a means to interpret Lambek Grammars:

Definition 4.1. A semantic model is a structure $M = \langle D_e, D_t \rangle$ where D_e is a finite set, a *domain of entities*, and where D_t is a finite set of *truth-values*.

From these very basic data, one can construct properties and relations over entities (e.g. the love relation, the property of being a man) or even relations between entities and truth-values. All these properties and relations are represented by *characteristic functions*, i.e. functions f_A such that $f_A(x) = 1$ iff $x \in A$. We denote the domain of functions by $D_{ab} := D_b^{D_a}$, inserting brackets to disambiguate. For instance, a relation $R \subseteq D_e \times D_e$ (imagine the relation 'x loves y' here) for a domain $D_e = \{j, m, b\}$ where we have that $R = \{(j, m), (j, b), (m, b), (b, m)\}$ can be represented by the function f_R in D_{eet}, where $f_R(x, y) = 1$ iff $(x, y) \in R$ (poor John!).

As a concrete example, we give a toy grammar for a few sentences in english and a corresponding semantic model. So, consider the following toy grammar:

$$\begin{array}{lcl} \text{john} & : & np \\ \text{mary} & : & np \\ \text{loves} & : & (np\backslash s)/np \end{array}$$

where we can show that sentences like *john loves mary*, *john loves john* etc. are grammatical. Now consider the semantic model $M = \langle \{j, m\}, \{0, 1\} \rangle$. We can now consider the following (typed) homomorphism from syntactic to semantic types and from words to semantic functions obeying the assigned types:

$$\begin{array}{lcl} h(np) & = & D_e \\ h(s) & = & D_t \\ h(a\backslash b) & = & D_{h(a)h(b)} \\ h(b/a) & = & D_{h(a)h(b)} \end{array}$$

and

$$\begin{array}{lcl} h(john) & = & j \\ h(mary) & = & m \\ h(loves) & = & f \end{array}$$

where f is the characteristic function of $\{(j, m), (j, j)\}$.

We can now see that the semantics for *john loves mary* is $f(m)(j) = 1$, indicating truth of the sentence, whereas *mary loves john* results in $f(j)(m) = 0$ (poor John!).

One can immediately observe that using these semantic models for a semantics leads to a fundamental problem: one needs to predefine the meaning of every word, something that is really not very desirable when it comes to practical applications. We will see that it is more desirable to switch to *vector space semantics*, giving a means of extracting word meaning out of a corpus and a means to measure the *similarity* of linguistic expressions.

4.1.2 Vector Space Semantics

Vector space semantics associates to words a *vector* instead of a semantic function. The nice thing about this kind of semantics is that the vectors in the dictionary can be obtained from a corpus in the form of *co-occurrence counts*, stating how often a word co-occurs with any other word in the corpus. In order to build compositional distributional models of meaning, we must first show that finite-dimensional vector spaces form an interpretable category for Lambek Grammars:

Proposition 4.1. *The category* **FVect** *of finite-dimensional vector spaces over* $\mathbb{R}$ *together with the tensor product* $\otimes$*, is a compact closed category.*

Proof. For every two vector spaces A, B there exists a tensor space $A \otimes B$, and for every two linear maps $f : A \to C, g : B \to D$, we construct their tensor product by $f \otimes g := \overrightarrow{a} \otimes \overrightarrow{b} \mapsto f(\overrightarrow{a}) \otimes g(\overrightarrow{b})$. We have bifunctoriality by

$$\begin{aligned}
&(id_A \otimes id_B)(\overrightarrow{a} \otimes \overrightarrow{b}) \\
&= id_A(\overrightarrow{a}) \otimes id_B(\overrightarrow{b}) \\
&= \overrightarrow{a} \otimes \overrightarrow{b} \\
&= id_{A\otimes B}(\overrightarrow{a} \otimes \overrightarrow{b})
\end{aligned}
\qquad\qquad
\begin{aligned}
&((k \otimes h) \circ (g \otimes f))(\overrightarrow{a} \otimes \overrightarrow{b}) \\
&= (k \otimes h)((g \otimes f)(\overrightarrow{a} \otimes \overrightarrow{b})) \\
&= (k \otimes h)(g(\overrightarrow{a}) \otimes f(\overrightarrow{b})) \\
&= k(g(\overrightarrow{a})) \otimes h(f(\overrightarrow{b})) \\
&= (k \circ g)(\overrightarrow{a}) \otimes (h \circ f)(\overrightarrow{b}) \\
&= ((k \circ g) \otimes (h \circ f))(\overrightarrow{a} \otimes \overrightarrow{b})
\end{aligned}$$

We define the unit to be $\mathbb{R}$ and go on to define the following (natural) isomorphisms:

$$\begin{aligned}
\alpha_{A,B,C} &:= (\overrightarrow{a} \otimes \overrightarrow{b}) \otimes \overrightarrow{c} \mapsto \overrightarrow{a} \otimes (\overrightarrow{b} \otimes \overrightarrow{c}) \\
\lambda_A &:= r \otimes \overrightarrow{a} \mapsto r\overrightarrow{a} \\
\rho_A &:= \overrightarrow{a} \otimes r \mapsto r\overrightarrow{a} \\
c_{A,B} &:= \overrightarrow{a} \otimes \overrightarrow{b} \mapsto \overrightarrow{b} \otimes \overrightarrow{a}
\end{aligned}$$

And coherence for a symmetric monoidal category follows easily.

Now define the dual of an object to be the dual vector space, i.e. $A^l = A^r = A^*$. Since we consider finite-dimensional vector spaces, we have that $A \cong A^*$ by sending the basis of A to the dual basis of A^*. Lastly, we can define an inner product on finite-dimensional vector spaces over $\mathbb{R}$. So we define the following closure maps:

$$\eta^* := 1 \mapsto \sum_i \overrightarrow{v}_i \otimes \overrightarrow{v}_i \qquad\qquad \epsilon^* := \overrightarrow{v}_i \otimes \overrightarrow{v}_j \mapsto \langle \overrightarrow{v}_i \mid \overrightarrow{v}_j \rangle$$

where n is the dimension of V and $\{v_i\}_{i=1}^n$ is a basis for V.
Coherence follows easily. □

Corollary 4.1. *The category* **FVect** *of finite-dimensional vector spaces over* $\mathbb{R}$ *together with the tensor product* $\otimes$*, is a monoidal bi-closed category.*

We will explicitly state the definitions of β and γ in this category. We have that both β and γ are defined in terms of the co-evaluation map given by the map η^*, so we get

$$\beta(f) := \overrightarrow{b} \mapsto \sum_i \overrightarrow{a_i} \otimes f(\overrightarrow{a_i} \otimes \overrightarrow{b})$$
$$\gamma(f) := \overrightarrow{a} \mapsto \sum_i f(\overrightarrow{a} \otimes \overrightarrow{b_i}) \otimes \overrightarrow{b_i}$$

We have that β^{-1} and γ^{-1} are defined in terms of the evaluation map given by ϵ^*, so we get

$$\beta^{-1}(g) := \overrightarrow{a} \otimes \overrightarrow{b} \mapsto \sum_{ij} w_{ij} \langle \overrightarrow{a} | \overrightarrow{a_i'} \rangle \overrightarrow{c_j} \text{ where } g(\overrightarrow{b}) = \textstyle\sum_{ij} w_{ij} \overrightarrow{a_i'} \otimes \overrightarrow{c_j}$$
$$\gamma^{-1}(g) := \overrightarrow{a} \otimes \overrightarrow{b} \mapsto \sum_{ij} \overrightarrow{c_i} \langle \overrightarrow{b_j'} | \overrightarrow{b} \rangle w_{ij} \text{ where } g(\overrightarrow{a}) = \textstyle\sum_{ij} w_{ij} \overrightarrow{c_i} \otimes \overrightarrow{b_j'}$$

In fact, finite-dimensional vector spaces are stronger than semantic models in the sense that every semantic model can be modeled using vector spaces: the following definition gives the *vector representation* of a semantic model, indicating that every semantic model can be represented within vector space semantics with preservation of truth and falsity.

Definition 4.2. Let $M = \langle D_e, D_t \rangle$ be a semantic model. The *vector space representation* of M is as follows:

- The vector space V_{D_e} is spanned by the basis vectors $\overrightarrow{x}$ for $x \in D_e$,
- The vector space V_{D_t} is spanned by one basis vector $\overrightarrow{1}$,
- Any function space D_{AB} is represented by $V_{D_A} \Rightarrow V_{D_B}$.

More specifically, any semantic function $f \in D_{AB}$ on a semantic model M can now be interpreted in the vector representation of M as the following linear map $V(f) \in V_{D_A} \Rightarrow V_{D_B}$: for any basis vector $\overrightarrow{x} \in V_{D_A}$, we have that $V(f)(\overrightarrow{x}) = \overrightarrow{y}$ where $\overrightarrow{y}$ is the vector representation of $y \in D_B$. A special case is when $D_B = D_t$, in which case 0 is interpreted as $\overrightarrow{0}$ and 1 is interpreted as the unique basic vector $\overrightarrow{1} \in V_{D_t}$.

4.2 Interpreting Lambek Calculi

To interpret a Lambek grammar into a suitable semantics, we must first define what a suitable semantic model is. In principle, we want this to be realized by any monoidal bi-closed category, but we need something to map the words in the grammar in question to. Hence, we define a semantic signature over a bi-closed category:

Definition 4.3. A semantic signature over a bi-closed category $\mathbf{C}$ is a triple $M = (C, \tau, S)$ where:

- C is a finite set of *constants*,
- $\tau : C \to Ob(\mathbf{C})$ is a map that assigns an object in $\mathbf{C}$ to every constant,
- S is a distinguished goal type in $Ob(\mathbf{C})$.

Now that we have defined semantic signatures, we can define a semantic interpretation over Lambek grammars:

Definition 4.4. Let $G = (\Sigma, \delta, S)$ be a (categorical) Lambek Grammar over $C(NL_T)$ and let $M = (C, \tau, S')$ be a semantic signature over a bi-closed category $\mathbf{C}$. A *semantic interpretation* is a bi-closed functor $F : C(NL_T) \to \mathbf{C}$ together with a map $I : \Sigma \to C$ such that

$$F(S) = S'$$
$$F(\delta(w)) = \tau(I(w)) \text{ for all } w \in \Sigma$$

i.e. the word interpretation map I respects types.

Note that, in order to define the meaning of a sentence, we require the constants of a semantic signature to be actual inhabitants of the objects of the category the signature is defined over. So, given this assumption, we can define the meaning of a sentence as follows:

Definition 4.5. Given $G = (\Sigma, \delta, S)$ a Lambek Grammar over $C(NL_T)$, $M = (C, \tau, S')$ a semantic signature over $\mathbf{C}$ such that the elements of C are actual inhabitants of the objects of $\mathbf{C}$, and a semantic interpretation $\langle F, I \rangle$, the meaning of a string $w = w_1...w_n \in \Sigma^+$ is defined iff w is a sentence of G. If it is defined, the meaning of w is $F(p)(I(w_1) \otimes ... \otimes I(w_n))$ where p is the grammatical proof of $\delta(w_1) \otimes ... \otimes \delta(w_n) \to S$.

We are now ready to define the basic compositional distributional model of meaning as it is employed in (Coecke et al., 2013) [1]

Definition 4.6. A categorical compositional distributional model of meaning is a triple $(G, M, \langle F, I \rangle)$ where G is a Lambek grammar, M is a semantic signature and $\langle F, I \rangle$ is a semantic interpretation over G and M.

4.3 An Example CCDMM

To illustrate our definitions, we give a simple truth-theoretic instantiation of a categorical compositional distributional model in terms of vector space semantics.

Consider the following Lambek grammar G:

john	:	np
mary	:	np
loves	:	$(np \backslash s)/np$
the	:	np/n
unicorn	:	n

[1] Actually, the model of Coecke et al. is based on Lambek monoids, which enjoy antisymmetry.

A witness for the sentencehood of *John loves the unicorn* is $\lhd^{-1} \lhd^{-1} \rhd^{-1}((\lhd \rhd^{-1} (1_{(np\backslash s)/np}))/ 1_n)$.

Now we define the following semantic signature: **C** is the category of finite dimensional vector spaces freely generated over the vector spaces N and S, where N is spanned by $\overrightarrow{j}, \overrightarrow{m}, \overrightarrow{u_1}, \overrightarrow{u_2}$ and S is spanned by just one vector $\overrightarrow{1}$. We define $C := \{\overrightarrow{j}, \overrightarrow{m}, \overrightarrow{u_1}, \overrightarrow{u_2}, \overrightarrow{1}, f, g, h\}$, where f is the vector representation of the characteristic function of $\{(j, u_1), (j, u_2)\}$, that is,

$$f \cong \overrightarrow{j} \otimes \overrightarrow{1} \otimes \overrightarrow{u_1} + \overrightarrow{j} \otimes \overrightarrow{1} \otimes \overrightarrow{u_2} + \sum_{\overrightarrow{x}, \overrightarrow{y} \in \{n_i\}_i} \overrightarrow{x} \otimes \overrightarrow{0} \otimes \overrightarrow{y} \text{ where } (x, y) \notin \{(j, u_1), (j, u_2)\}$$

and g is the map from $N \otimes S$ to N that returns $\overrightarrow{x}$ only if the right value in the vector in $N \otimes S$ is exactly $\overrightarrow{1}$ and corresponds to $\overrightarrow{x}$ and returns $\overrightarrow{0}$ otherwise, that is,

$$\begin{gathered} g \cong \overrightarrow{u_1} \otimes (\overrightarrow{u_1} \otimes \overrightarrow{1} + \overrightarrow{u_2} \otimes \overrightarrow{0} + \overrightarrow{j} \otimes \overrightarrow{0} + \overrightarrow{m} \otimes \overrightarrow{0}) \\ + \overrightarrow{u_2} \otimes (\overrightarrow{u_1} \otimes \overrightarrow{0} + \overrightarrow{u_2} \otimes \overrightarrow{1} + \overrightarrow{j} \otimes \overrightarrow{0} + \overrightarrow{m} \otimes \overrightarrow{0}) \\ + \overrightarrow{j} \otimes (\overrightarrow{u_1} \otimes \overrightarrow{0} + \overrightarrow{u_2} \otimes \overrightarrow{0} + \overrightarrow{j} \otimes \overrightarrow{1} + \overrightarrow{m} \otimes \overrightarrow{0}) \\ + \overrightarrow{m} \otimes (\overrightarrow{u_1} \otimes \overrightarrow{0} + \overrightarrow{u_2} \otimes \overrightarrow{0} + \overrightarrow{j} \otimes \overrightarrow{0} + \overrightarrow{m} \otimes \overrightarrow{1}) \\ + \overrightarrow{0} \otimes \text{ other combinations in } N \otimes S. \end{gathered}$$

and finally, h is the vector $\overrightarrow{j} \otimes \overrightarrow{0} + \overrightarrow{m} \otimes \overrightarrow{0} + \overrightarrow{u_1} \otimes \overrightarrow{1} + \overrightarrow{u_2} \otimes \overrightarrow{1}$ in $N \otimes S$.

Finally, we define the following semantic interpretation to complete our model: F is the straightforward strict bi-closed functor

$$\begin{array}{lcl} F(np) & = & N \\ F(n) & = & N \otimes S \\ F(s) & = & S \\ F(A \otimes B) & = & F(A) \otimes F(B) \\ F(A\backslash B) & = & F(A) \Rightarrow F(B) \\ F(B/A) & = & F(B) \Leftarrow F(A) \end{array}$$

and I, the interpretation map, is defined as follows:

$$\begin{array}{lcl} I(john) & = & \overrightarrow{j} \\ I(mary) & = & \overrightarrow{m} \\ I(loves) & = & f \\ I(the) & = & g \\ I(unicorn) & = & h \end{array}$$

We now have a complete compositional distributional model of meaning. The meaning of *John loves the unicorn*, for instance, becomes the application of $F([\lhd^{-1} \lhd^{-1} \rhd^{-1}((\lhd \rhd^{-1} (1_{(np\backslash s)/np}))/ 1_n)])$ to $\overrightarrow{j} \otimes f \otimes g \otimes h$. This is

$$\begin{gathered} \beta^{-1}\beta^{-1}\gamma^{-1}((\beta\gamma^{-1}(id_{(N\otimes S)\otimes N})) \Leftarrow id_{N\otimes S})(\overrightarrow{j} \otimes f \otimes g \otimes h) = \\ \sum_{ik} \langle f \mid (\overrightarrow{n_i} \otimes \overrightarrow{1}) \otimes \overrightarrow{n_k} \rangle \langle g_r \mid h \rangle \langle \overrightarrow{n_k} \mid g_l \rangle \langle \overrightarrow{j} \mid \overrightarrow{n_i} \rangle \overrightarrow{1} \end{gathered}$$

As $\langle \overrightarrow{j} \mid \overrightarrow{n_i} \rangle$ is 1 when $\overrightarrow{n_i} = \overrightarrow{j}$ and 0 otherwise, we can infer that in the expression, we only need to cover the cases where $\overrightarrow{n_i} = \overrightarrow{j}$. But then we can infer from the definition of f that $\overrightarrow{n_k}$ can be instantiated by $\overrightarrow{u_1}$ and $\overrightarrow{u_2}$. This leads us to infer that g_l can only be instantiated with $\overrightarrow{u_1}$ or $\overrightarrow{u_2}$, leading to consider only those right hand parts for g when taking the inner product with h. But as these never agree with h, we still end up with an inner product of 0. Hence, the whole result will be $\overrightarrow{0}$, representing the value *false*. This, however, is as predicted since *the* can only refer to a specific unicorn when there is in fact exactly one unicorn.

Of course, in practice one will want to extract the meaning of basic words from a sufficiently large corpus. We will not concentrate on actually doing this, but rather give directions on how one would go about this in the next section.

4.4 Obtaining CCDMMs

How are compositional distributional models of meaning set up? Obviously, one want to avoid the problem of having to predefine the semantic lexicon. But for the same reason, we do not want to have to predefine grammars as well. So, obtaining a compositional distributional model becomes the task of (a) inducing a grammar from a corpus and (b) inducing a semantics from a corpus.

Addressing part (b) first, one of the motivations of choosing vector space semantics over montague semantics is exactly the idea that vector spaces lend themselves perfectly to distributional models, in which co-occurrence vectors/matrices are extracted from big data to obtain a meaning for individual words. So, the semantics can be obtained from an unannotated corpus.

The question remains how grammars are obtained from big data; this is the problem of *grammar induction*. This can be done either from semantic data, as is done by Delpeuch (2014), but inducing grammars from semantic data is already NP-complete for systems based on compact closed categories. Another way of grammar induction is to obtain grammars from a set of structured sentences, as is done by Bonato and Retoré (2001); Kanazawa (1996).

Part II

Extended Compositional Distributional Models of Meaning

Chapter 5

Categories Revisited

In this chapter, we continue our categorical exploration and delve into new concepts, largely obtained by duality. We consider the dual notions of the various kinds of closed categories defined in Chapter 1, giving rise to co-closed or *open* categories. We continue to explore the implications of merging closed and open categories and adding *linear distributivities* to categories with a tensor and cotensor. We will observe the new forms of symmetry obtained from the consideration of open categories.

5.1 Open Categories

We have already encountered the notion of *duality* in Chapter 1 where we defined the opposite category and turned monads into comonads. Here we will apply duality to the various kinds of closed categories considered in the first chapter. We saw that for $\mathbf{C}$ to be a left closed tensor category, the functor $A \otimes \_$ determined by left tensoring by a fixed object A must have a right adjoint $A \Rightarrow \_$ determined by applying the internal hom with a fixed object A. This gives a natural isomorphism between $Hom_{\mathbf{C}}(A \otimes B, C)$ and $Hom_{\mathbf{C}}(B, A \Rightarrow C)$ natural in A, B and C. Dualizing this notion, we get that a category $\mathbf{C}$ is right co-closed when it has a bifunctor $\oplus : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ together with a bifunctor $\ll:: \mathbf{C} \times \mathbf{C}^{\mathrm{op}} \to \mathbf{C}$ such that the functor $\_ \oplus A$ has as a *left adjoint* the functor $\_ \ll A$, inducing an isomorphism between $Hom(C, B \oplus A)$ and $Hom_{\mathbf{C}}(C \ll A, B)$ natural in A, B and C. So, in summary, we get the following notions for free:

Definition 5.1. A right open tensor category is a tensor category $(\mathbf{C}, \oplus)$ equipped with a bifunctor $\ll: \mathbf{C} \to \mathbf{C}^{\mathrm{op}} \to \mathbf{C}$ together with a natural isomorphism specified by $\sigma_{A,B,C} : Hom^{\mathbf{C}}(C, B \oplus A) \to Hom^{\mathbf{C}}(C \ll A, B)$.

Definition 5.2. A left open tensor category is a tensor category $(\mathbf{C}, \oplus)$ equipped with a bifunctor $\gg: \mathbf{C}^{\mathrm{op}} \times \mathbf{C} \to \mathbf{C}$ together with a natural isomorphism specified by $\omega_{A,B,C} : Hom^{\mathbf{C}}(C, B \oplus A) \to Hom^{\mathbf{C}}(B \gg C, A)$.

Definition 5.3. A bi-open tensor category is a tensor category $(\mathbf{C}, \oplus)$ that is both left and right open.

Following the same pattern as in Chapter 1, we can analogously define associative and unitary open tensor categories and monoidal open categories, to get another hierarchy of categories with additional structure. Finally, we consider *merging* closed and open categories to get what we like to call *clopen* categories:

Definition 5.4. A bi-clopen (bi-)tensor category is a double tensor category $(\mathbf{C}, \otimes, \oplus)$ that is bi-closed with respect to $\otimes$ and bi-open with respect to $\oplus$.

We may now ask ourselves what happens when we dualize the notion of an autonomous category. The following proposition states that nothing happens!

Proposition 5.1. *Autonomous categories are self-dual, i.e. every co-autonomous category is an autonomous category.*

Proof. Dualizing the notion of an autonomous category gives us the following maps:

$$\begin{array}{ll} co\text{-}\eta^l : A \otimes A^l \to I & co\text{-}\epsilon^l : I \to A^l \otimes A \\ co\text{-}\eta^r : A^r \otimes A \to I & co\text{-}\epsilon^r : I \to A \otimes A^r \end{array}$$

together with the four yanking diagrams but with the arrows reversed. Now set $A^l := A^r$ and $A^r := A^l$ and

$$\begin{array}{l} \eta^l := co\text{-}\epsilon^r \\ \epsilon^l := co\text{-}\eta^l \\ \eta^r := co\text{-}\epsilon^l \\ \epsilon^r := co\text{-}\eta^l \end{array}$$

It is immediate that the dual diagrams now form exactly the defining diagrams of an autonomous category. □

As autonomous categories are self-dual, so are their symmetric variants compact closed categories. We can thus consider autonomous and compact closed categories as instantiations of bi-clopen categories where the two tensors are identified:

Corollary 5.1. *Every autonomous category is a bi-clopen category.*

We will now consider another symmetry that is obtained by considering the relation between left closed and right open (and similarly, right closed and left open) categories.

5.2 Another Symmetry

As we have obtained the definitions of left and right open categories from duality, it is obvious there is a covariant isomorphism of categories between any left open and right open category sharing the same set of basic objects. However, we note here that there is also a *contravariant* isomorphism of categories between left closed and right open categories on the one hand, and between right closed and left open categories on the other hand:

Let $(\mathbf{C}, \otimes, \Rightarrow, \beta)$ be a left closed tensor category and let $(\mathbf{D}, \oplus, \ll, \sigma)$ be a right open category. Define the following (contravariant) functor $F : \mathbf{C} \to \mathbf{D}$:

$$\begin{array}{lcl} F(A) & = & A \\ F(A \otimes B) & = & F(B) \oplus F(A) \\ F(A \Rightarrow B) & = & F(B) \ll F(A) \\ F(id_A) & = & id_{F(A)} \\ F(g \circ f) & = & F(f) \circ F(g) \\ F(f \otimes g) & = & F(g) \oplus F(f) \\ F(f \Rightarrow g) & = & F(g) \ll F(f) \\ F(\beta(f)) & = & \sigma(F(f)) \end{array}$$

Next, define the following functor $G : \mathbf{D} \to \mathbf{C}$:

$$\begin{array}{lcl} G(A) & = & A \\ G(B \oplus A) & = & G(A) \otimes G(B) \\ G(B \ll A) & = & G(A) \Rightarrow G(B) \\ G(id_A) & = & id_{G(A)} \\ G(g \circ f) & = & G(f) \circ G(g) \\ G(g \oplus f) & = & G(f) \otimes F(g) \\ G(g \ll f) & = & G(f) \Rightarrow G(g) \\ G(\sigma(f)) & = & \beta(G(f)) \end{array}$$

Proposition 5.2. *The functors F, G establish an isomorphism of categories between $\mathbf{C}$ and $\mathbf{D}$.*

In a similar fashion, one can construct an isomorphism between right closed and left open tensor categories. Obviously, these results extend to the monoidal case. Moreover, one can merge these symmetries to obtain a contravariant automorphism on bi-clopen categories, to get another automorphism providing a symmetry on bi-clopen categories. As there is in general no relation

between the two tensors of a bi-clopen category, we will consider adding distributivities that relate the two "families". Anticipating on the next chapter, we require that these distributivities respect resources, and thus we consider *linear distributivities*.

5.3 Linearly Distributive Categories

In this section we consider *linear distributive categories* by enriching bi-clopen categories with interaction transformations. As the logics we consider are designed for linguistic analysis, we wish to add distributivities that *respect resources*, i.e. they should not copy or delete any information, but rather restructure the given information in a way that does not induce associativity or commutativity. We borrow the term *linear distributivities* from the work of Cockett et al. (Cockett and Seely, 1997b; Blute et al., 1996; Cockett and Seely, 1997a) but we do intend to refer to a slightly different notion here: indeed, linear distributivities in the sense of Cockett and Seely refers to natural transformations relating two tensors in categories providing a characterization for linear logic. They consider transformations of the form $\delta^L_L : A \otimes (B \oplus C) \to (A \otimes B) \oplus C$ and $\delta^R_R : (B \oplus C) \otimes A \to B \oplus (C \otimes A)$ which induce, in the case of a symmetric tensor, the permuting distributivities $\delta^L_R : A \otimes (B \oplus C) \to B \oplus (A \otimes C)$ and $\delta^R_L : (B \oplus C) \otimes A \to (B \otimes A) \oplus C$. We deviate from their definitions for the purpose of characterizating the *Lambek-Grishin Calculus* with type IV interactions (to be defined in Chapter 7); the distributivities used there have shown to have linguistic application by the modelling of adjunction in Tree Adjoining Grammar (see (Moot, 2007)). Rather than directly relating the two tensors $\otimes$ and $\oplus$, we rely on interaction between the primary tensor $\otimes$ and the two opening bifunctors $\ll$ and $\gg$ (we shall see later on that this induces also a relation between the secondary tensor $\oplus$ and the primary closing bifunctors $\Rightarrow$ and $\Leftarrow$). For the case of a bi-clopen tensor category, these natural transformations need not satisfy any coherence axioms, hence the definition becomes as follows:

Definition 5.5. A linearly distributive tensor category is a bi-closed, bi-open tensor category $(\mathbf{C}, \otimes, \oplus, \Rightarrow, \Leftarrow, \ll, \gg, \beta, \gamma, \sigma, \omega)$ together with natural transformations specified by

$$\begin{aligned}
\delta^L &: (A \gg B) \otimes C \to A \gg (B \otimes C) \\
\delta^R &: C \otimes (B \ll A) \to (C \otimes B) \ll A \\
\kappa^L &: C \otimes (A \gg B) \to A \gg (C \otimes B) \\
\kappa^R &: (B \ll A) \otimes C \to (B \otimes C) \ll A
\end{aligned}$$

Things change when the base category is monoidal, and additional coherence axioms must be stated:

Definition 5.6. A linearly distributive monoidal category is a bi-clopen monoidal category $(\mathbf{C}, \otimes, \oplus, I, J, \alpha_\otimes, \lambda_\otimes, \rho_\otimes, \alpha_\oplus, \\ , \Leftarrow, \beta, \gamma, \ll, \gg, \sigma, \omega)$ together with natural transformations specified by

$$\begin{aligned}
\delta^L &: (A \gg B) \otimes C \to A \gg (B \otimes C) \\
\delta^R &: C \otimes (B \ll A) \to (C \otimes B) \ll A \\
\kappa^L &: C \otimes (A \gg B) \to A \gg (C \otimes B) \\
\kappa^R &: (B \ll A) \otimes C \to (B \otimes C) \ll A
\end{aligned}$$

satisfying additionally the following equations:

- Unit and distribution:

$$\begin{array}{lcl}\rho_{\otimes} & = & (id \gg \rho_{\otimes}) \circ \delta^L \\ \lambda_{\otimes} & = & (\lambda_{\otimes} \ll id) \circ \delta^R \\ \lambda_{\otimes} & = & (id \gg \lambda_{\otimes}) \circ \kappa^L \\ \rho_{\otimes} & = & (\rho_{\otimes} \ll id) \circ \kappa^R \end{array}$$

- Associativity and distribution:

$$\begin{array}{lcl}\delta^L \circ \alpha_{\otimes} & = & (id \gg \alpha_{\otimes}) \circ \delta^L \circ (\delta^L \otimes id) \\ \delta^R \circ \alpha_{\otimes}^{-1} & = & (\alpha_{\otimes}^{-1} \ll id) \circ \delta^R \circ (id \otimes \delta^R) \\ (id \gg \alpha_{\otimes}) \circ \kappa^L & = & \kappa^L \circ (id \otimes \kappa^L) \circ \alpha_{\otimes} \\ (\alpha_{\otimes}^{-1} \ll id) \circ \kappa^R & = & \kappa^R \circ (\kappa^R \otimes id) \circ \alpha_{\otimes}^{-1} \end{array}$$

- Distribution and distribution:

$$\begin{array}{lcl}\kappa^L \circ (id \otimes \delta^L) \circ \alpha_{\otimes} & = & (id \gg \alpha_{\otimes}) \circ \delta^L \circ (\kappa^L \otimes id) \\ \kappa^R \circ (\delta^R \otimes id) \circ \alpha_{\otimes}^{-1} & = & (\alpha_{\otimes}^{-1} \ll id) \circ \delta^R \circ (id \otimes \kappa^R) \end{array}$$

The obvious question is whether the symmetries on a bi-clopen tensor category are preserved in the presence of linear distributivities. The answer is confirmative, as we will show next.

5.3.1 Symmetry Preserved

We already noted that there are two symmetries in a bi-clopen tensor category: one in the form of a covariant functor establishing a symmetry between left and right, and one in the form of a contravariant functor establishing a symmetry between left/right and closed/open. What happens when we try to apply these symmetries in the context of linear distributivities? We will show that by a suitable choice of mapping the distributivities, these symmetries are preserved.

Consider the left/right symmetry which maps objects as follows:

$$\begin{array}{lcl}S(A) & = & A \\ S(A \otimes B) & = & S(B) \otimes S(A) \\ S(A \Rightarrow B) & = & S(B) \Leftarrow S(A) \\ S(B \Leftarrow A) & = & S(A) \Rightarrow S(B) \\ S(B \oplus A) & = & S(A) \oplus S(B) \\ S(A \gg B) & = & S(B) \ll S(A) \\ S(B \ll A) & = & S(A) \gg S(B) \end{array}$$

The easiest way to extend this symmetry to the linear distributive case is to simply map δ^L to δ^R and κ^L to κ^R and back. We then get the following proposition:

Proposition 5.3. *The endofunctor S is involutive.*

Now consider the left/right and closed/open symmetry, which maps objects as follows:

$$\begin{array}{lcl}T(A) & = & A \\ T(A \otimes B) & = & T(B) \oplus T(A) \\ T(A \Rightarrow B) & = & T(B) \ll T(A) \\ T(B \Leftarrow A) & = & T(A) \gg T(B) \\ T(B \oplus A) & = & T(A) \otimes T(B) \\ T(A \gg B) & = & T(B) \Leftarrow T(A) \\ T(B \ll A) & = & T(A) \Rightarrow T(B) \end{array}$$

We now face the issue that, for instance, $T(\delta^L)$ should have as input the object $A \Rightarrow (B \oplus C)$ and as output $(A \Rightarrow B) \oplus C$. To see that this poses no problem at all, we define T on the linear distributivities as follows:

$$\begin{array}{lcl} T(\delta^L) & = & \omega^{-1}(\gamma(\omega(\gamma^{-1}(id_{(C\oplus B)\Leftarrow A})) \circ \delta^L)) \\ T(\delta^R) & = & \sigma^{-1}(\beta(\sigma(\beta^{-1}(id_{A\Rightarrow(B\oplus C)})) \circ \delta^R)) \\ T(\kappa^L) & = & \sigma^{-1}(\gamma(\sigma(\gamma^{-1}(id_{(B\oplus C)\Leftarrow A})) \circ \kappa^R)) \\ T(\kappa^R) & = & \omega^{-1}(\beta(\omega(\beta^{-1}(id_{A\Rightarrow(C\oplus B)})) \circ \kappa^L)) \end{array}$$

To ensure that T is involutive, we need to show that $T(T(f)) = f$ for $f \in \{\delta^L, \delta^R, \kappa^L, \kappa^R\}$. Since the cases for δ^R and κ^R are similar to their left variants by the first symmetry, we only show the cases for δ^L and κ^L. The first case is proven by naturality and isomorphicity of γ, ω and δ^L:

$$\begin{array}{c} T(T(\delta^L)) = \\ T(\omega^{-1}(\gamma(\omega(\gamma^{-1}(id_{(C\oplus B)\Leftarrow A})) \circ \delta^L))) = \\ \gamma^{-1}(\omega(\omega^{-1}(\gamma(\omega(\gamma^{-1}(id_{(C\oplus B)\Leftarrow A})) \circ \delta^L)) \circ \gamma(\omega^{-1}(id_{A\gg(B\otimes C)})))) = \\ \gamma^{-1}(\omega(\omega^{-1}(\gamma(\omega(\gamma^{-1}(id_{(C\oplus B)\Leftarrow A})) \circ \delta^L) \circ id_C \gg \gamma(\omega^{-1}(id_{A\gg(B\otimes C)}))))) = \\ \gamma^{-1}(\gamma(\omega(\gamma^{-1}(id_{(C\oplus B)\Leftarrow A})) \circ \delta^L) \circ id_C \gg \gamma(\omega^{-1}(id_{A\gg(B\otimes C)}))) = \\ \gamma^{-1}(\gamma(\omega(\gamma^{-1}(id_{(C\oplus B)\Leftarrow A})) \circ \delta^L \circ (id_C \gg \gamma(\omega^{-1}(id_{A\gg(B\otimes C)}))) \otimes id_A)) = \\ \omega(\gamma^{-1}(id_{(C\oplus B)\Leftarrow A})) \circ \delta^L \circ ((id_C \gg (\gamma(\omega^{-1}(id_{A\gg(B\otimes C)})))) \otimes id_A) = \\ \omega(\gamma^{-1}(id_{(C\oplus B)\Leftarrow A})) \circ (id_C \gg (\gamma(\omega^{-1}(id_{A\gg(B\otimes C)})) \otimes id_A)) \circ \delta^L = \\ \omega(\gamma^{1}(id_{(C\oplus B)\Leftarrow A}) \circ (\gamma(\omega^{-1}(id_{A\gg(B\otimes C)})) \otimes id_A)) \circ \delta^L = \\ \omega(\gamma^{-1}(\gamma(\omega(id_{A\gg(B\otimes C)})))) \circ \delta^L = \\ \delta^L. \end{array}$$

The proof of the second case involves naturality and isomorphicity of β and ω and naturality of κ^L:

$$\begin{array}{c} T(T(\kappa^L)) = \\ T(\sigma^{-1}(\gamma(\sigma(\gamma^{-1}(id_{(B\oplus C)\Leftarrow A})) \circ \kappa^R))) = \\ \beta^{-1}(\omega(\omega^{-1}(\beta(\omega(\beta^{-1}(id_{A\Rightarrow(C\oplus B)})) \circ \kappa^L)) \circ \beta(\omega^{-1}(id_{A\gg(C\otimes B)})))) = \\ \beta^{-1}(\omega(\omega^{-1}(\beta(\omega(\beta^{-1}(id_{A\Rightarrow(C\oplus B)})) \circ \kappa^L) \circ (id_C \gg \beta(\omega^{-1}(id_{A\gg(C\otimes B)})))))) = \\ \beta^{-1}(\beta(\omega(\beta^{-1}(id_{A\Rightarrow(C\oplus B)})) \circ \kappa^L) \circ (id_C \gg \beta(\omega^{-1}(id_{A\gg(C\otimes B)})))) = \\ \beta^{-1}(\beta(\omega(\beta^{-1}(id_{A\Rightarrow(C\oplus B)})) \circ \kappa^L \circ (id_A \otimes (id_C \gg \beta(\omega^{-1}(id_{A\gg(C\otimes B)})))))) = \\ \omega(\beta^{-1}(id_{A\Rightarrow(C\oplus B)})) \circ \kappa^L \circ (id_A \otimes (id_C \gg \beta(\omega^{-1}(id_{A\gg(C\otimes B)})))) = \\ \omega(\beta^{-1}(id_{A\Rightarrow(C\oplus B)})) \circ (id_C \gg (id_A \otimes \beta(\omega^{-1}(id_{A\gg(C\otimes B)})))) \circ \kappa^L = \\ \omega(\beta^{-1}(id_{A\Rightarrow(C\oplus B)}) \circ (id_A \otimes \beta(\omega^{-1}(id_{A\gg(C\otimes B)})))) \circ \kappa^L = \\ \omega(\beta^{-1}(\beta(\omega^{-1}(id_{A\gg(C\oplus B)})))) \circ \kappa^L = \\ \kappa^L. \end{array}$$

By the symmetry established by the functor S, the remaining cases are covered, so we get the following result:

Proposition 5.4. *The endofunctor T is involutive.*

Now that we have established the categorical definitions needed for characterizing the intended extensions of the Lambek Calculus, we will consider dualizing the graphical language we have developed in Chapter 2.

Chapter 6

Graphical Languages, Again

In this chapter, we review graphical languages for open tensor categories, and linearly distributive categories.

6.1 Graphical Languages Dualized?

As we have reviewed in the previous chapter, there is a certain symmetry between closed and open categories. We will want to exploit this duality at the level of graphical languages, in order to automatically obtain graphical languages for open tensor categories. This graphical duality also means that we get coherence for free. After exploring the novel graphical language, which we will refer to as *dual proof nets*, we consider merging the graphical languages and investigate coherence for this new language. A final step is to include linear distributivities to the graphical language and assert its coherence.

6.2 Graphical Languages for Open Tensor Categories

We wish to develop a graphical language for bi-open tensor categories, but we wish to avoid the tedious work of Chapter 2. Our way out then resides in duality: given that bi-open tensor categories are dual to bi-closed tensor categories, we may exploit these symmetries at the graphical level. We will start out by dualizing the sequent calculus for **NL** to obtain a sequent calculus called **NG** (we will see in the next chapter why we chose a **G** abbreviating Grishin). We then develop a method of dualizing proof nets into dual proof nets, and go on to define these and their equations. Finally, we provide a dual sequentialization and give a method of obtaining morphisms in a bi-open tensor category from the dual sequent calculus.

6.2.1 A Dual Sequent Calculus

As we want to employ the dual of the sequent calculus defined in Chapter 2, we simply transfer contexts to the right of the turnstile and exchange premisses. We then get the following dual definition of formulae:

Definition 6.1 (Formulae)**.** Given a set of atomic formulae At, the set of dual formulae is defined as follows:

$$A, B := p \mid B \oplus A \mid B \oslash A \mid A \otimes B \text{ for } p \in At.$$

Given that we can express contexts now with the same binary merger as in the calculus for **NL**, we can skip immediately to the definition of a dual sequent calculus:

Definition 6.2 (Dual Sequent Calculus)**.** The sequent calculus presentation of **NG** is as follows:

$$\frac{}{A \vdash A}\ Id \qquad \frac{A \vdash \Gamma[B] \quad B \vdash \Delta}{A \vdash \Gamma[\Delta]}\ Cut$$

$$\frac{C \vdash \Gamma[B \bullet A]}{C \vdash \Gamma[B \oplus A]} \oplus R \qquad \frac{B \vdash \Delta \quad A \vdash \Gamma}{B \oplus A \vdash \Delta \bullet \Gamma} \oplus L$$

$$\frac{C \vdash \Gamma[A] \quad B \vdash \Delta}{C \vdash \Gamma[A \oslash B \bullet \Delta]} \oslash R \qquad \frac{A \vdash \Gamma \bullet B}{A \oslash B \vdash \Gamma} \oslash L$$

$$\frac{C \vdash \Gamma[A] \quad B \vdash \Delta}{C \vdash \Gamma[\Delta \bullet B \otimes A]} \otimes R \qquad \frac{A \vdash B \bullet \Gamma}{B \otimes A \vdash \Gamma} \otimes L$$

We can then dualize the equivalence relation on sequent proofs by adopting a similar notation for proofs:

Definition 6.3. We define the following equivalence relation on proofs in **NG**:

- Identity unfolding, meaning

$$\oplus R(\oplus L(Id(B), Id(A))) \equiv Id(B \oplus A)$$
$$\oslash L(\oslash R(Id(B), Id(A))) \equiv Id(B \oslash A)$$
$$\otimes L(\otimes R(Id(B), Id(A))) \equiv Id(A \otimes B)$$

- Cut-elimination base case, meaning

$$Cut(Id(A), D_1) \equiv D_1$$
$$Cut(D_1, Id(B)) \equiv D_1$$

- Principal cut-elimination, meaning

$$Cut(\oplus R(D_3), \oplus L(D_2, D_1)) \equiv Cut(Cut(D_3, D_1), D_2)$$
$$Cut(\oplus R(D_3), \oplus L(D_2, D_1)) \equiv Cut(Cut(D_3, D_2), D_1)$$
$$Cut(\oslash R(D_3, D_2), \oslash L(D_1)) \equiv Cut(D_3, Cut(D_1, D_2))$$
$$Cut(\oslash R(D_3, D_2), \oslash L(D_1)) \equiv Cut(Cut(D_3, D_1), D_2)$$
$$Cut(\otimes R(D_3, D_2), \otimes L(D_1)) \equiv Cut(D_3, Cut(D_1, D_2))$$
$$Cut(\otimes R(D_3, D_2), \otimes L(D_1)) \equiv Cut(Cut(D_3, D_1), D_2)$$

- Permutative cut-elimination, meaning

$$Cut(D_2, \oplus R(D_1)) \equiv \oplus R(Cut(D_2, D_1))$$

$$Cut(D_3, \oslash R(D_2, D_1)) \equiv \oslash R(Cut(D_3, D_2), D_1)$$

$$Cut(\oplus R(D_2), D_1) \equiv \oplus R(Cut(D_2, D_1))$$

$$Cut(\oslash L(D_2), D_1) \equiv \oslash L(Cut(D_2, D_1))$$

$$Cut(\otimes L(D_2), D_1) \equiv \otimes L(Cut(D_2, D_1))$$

$$Cut(\oplus L(D_3, D_2), D_1) \equiv \oplus L(Cut(D_3, D_1), D_2)$$
when $LHS(D_1) = C$ and $RHS(D_3) = \Gamma'[C]$

$$Cut(\oplus L(D_3, D_2), D_1) \equiv \oplus L(D_3, Cut(D_2, D_1))$$
when $LHS(D_1) = C$ and $RHS(D_2) = \Gamma[C]$

$$Cut(\oslash R(D_3, D_2), D_1) \equiv \oslash R(Cut(D_3, D_1), D_2)$$
when $LHS(D_1) = C$ and $RHS(D_3) = \Gamma[C][A]$

$$Cut(\oslash R(D_3, D_2), D_1) \equiv \oslash R(D_3, Cut(D_2, D_1))$$

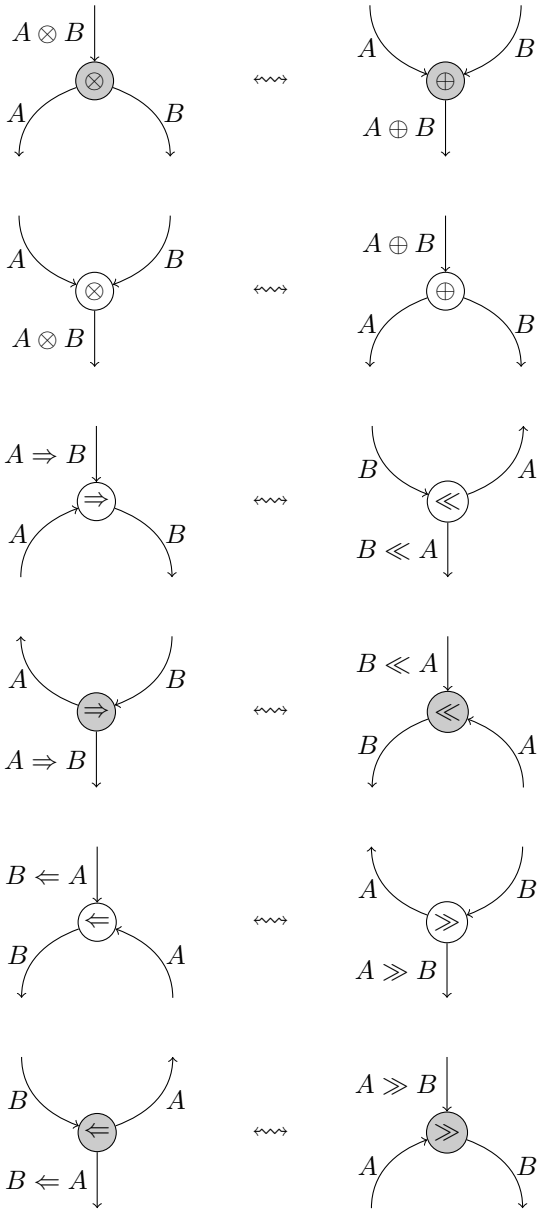
$$\text{when } LHS(D_1) = C \text{ and } RHS(D_2) = \Gamma'[C]$$

$$Cut(\oslash R(D_3, D_2), D_1) \equiv \oslash R(Cut(D_3, D_1), D_2)$$
$$\text{when } LHS(D_1) = C \text{ and } RHS(D_3) = \Gamma[C][A]$$

$$Cut(\oslash R(D_3, D_2), D_1) \equiv \oslash R(D_3, Cut(D_2, D_1))$$
$$\text{when } LHS(D_1) = C \text{ and } RHS(D_2) = \Gamma'[C]$$

6.2.2 Dual Proof Nets

According to the observed symmetry between closed and open categories, the $\otimes$ connective is linked to the $\oplus$ connective, while the $\Rightarrow$ is turned around and linked to the $\ll$ connective, and symmetrically we have that $\Leftarrow$ is linked to $\gg$. We say that this is the *symmetric image* of the connectives. A proof net for a bi-closed tensor category is turned into a proof net for a bi-open tensor category in four steps (which may be applied interchangeably): 1) flip the net vertically, 2) flip the net horizontally, 3) reverse the direction of the arrows and finally 4) replace each connective by its symmetric image. We then obtain new links for proof nets for a bi-open tensor category by applying this symmetry:

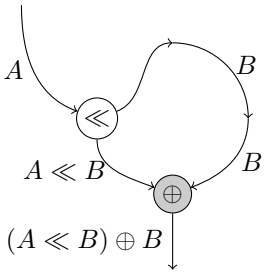


Given that we may apply this graphical symmetry also to the inductive definition of proof nets, we can immediately state the following inductive definition of dual proof nets (based on the simplified došen axiomatization):

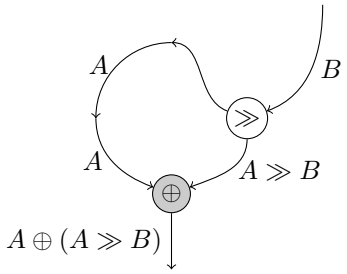
Definition 6.4. The class of dual proof nets is defined inductively as follows:

- **Identity.** The identity dual proof net for arbitrary A is given by

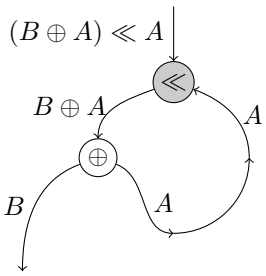
- **Dual Left Application.** The following is a dual proof net:



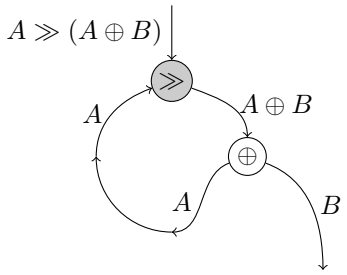
- **Dual Right Application** The following is a dual proof net:



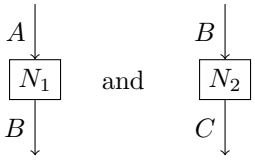
- **Dual Left Co-Application.** The following is a dual proof net:



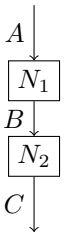
- **Dual Right Co-Application.** The following is a dual proof net:



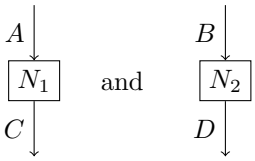
- **Composition.** Given two dual proof nets



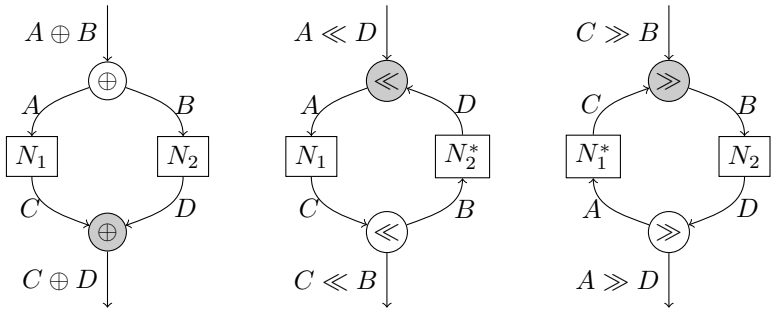
the following is a dual proof net:



- **Monotonicity.** Given two dual proof nets



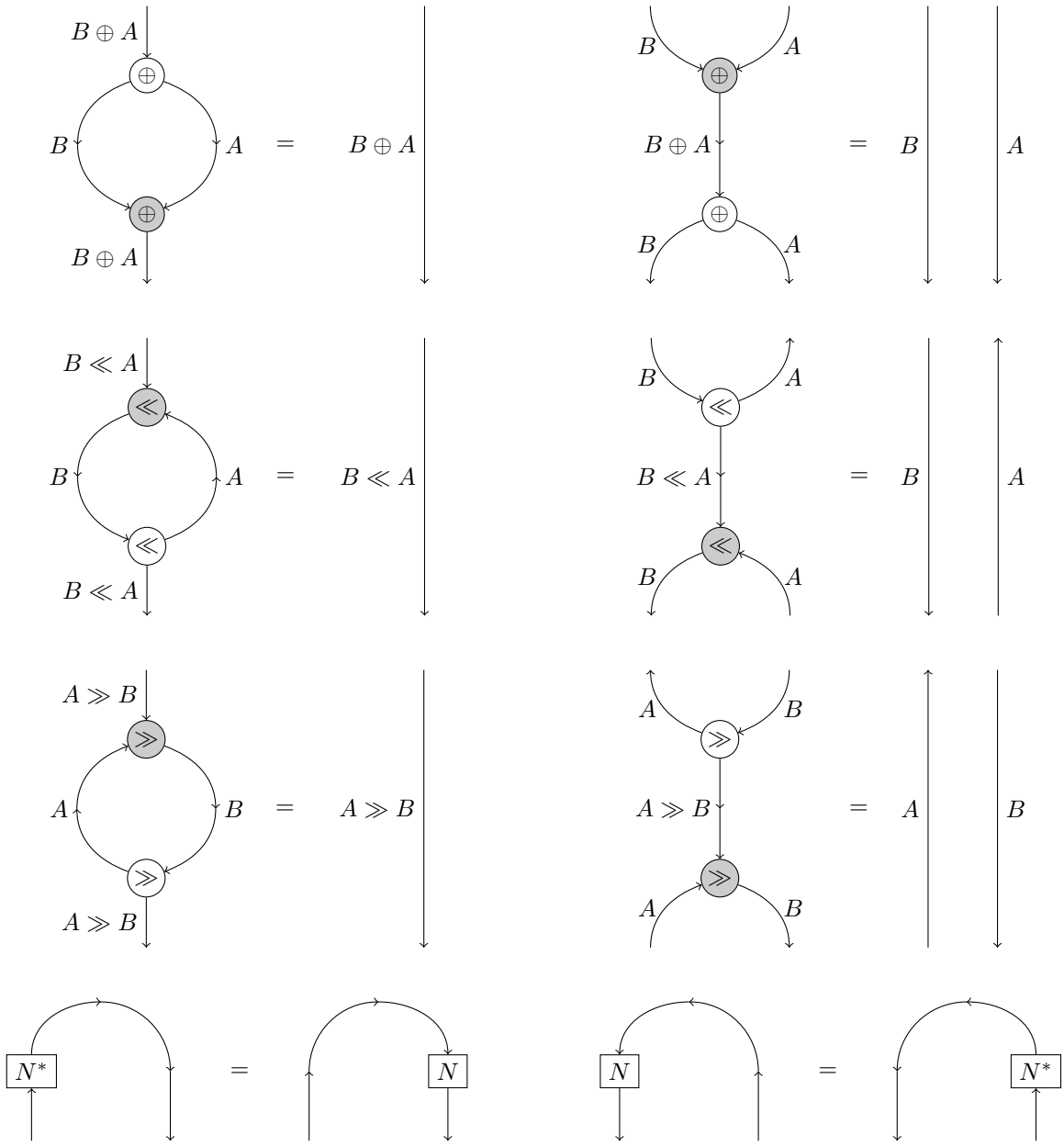
the following are dual proof nets:

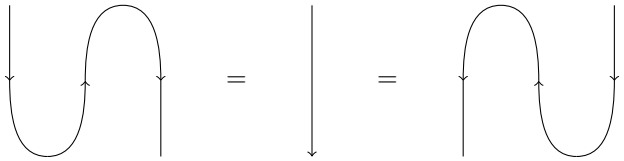


- Nothing else is a dual proof net.

6.2.3 Dual Equations

To obtain the dual equations for our dual proof nets, we simply apply the bi-closed to bi-open tensor category symmetry to our normal equations from Chapter 2. We then get the following equations on dual proof nets (next to reflexivity, transitivity and symmetry):


$B \oplus A$
B
A
$B \oplus A$
$=$
$B \oplus A$
B
A
$B \oplus A$
B
A
$=$
B
A
$B \ll A$
B
A
$B \ll A$
$=$
$B \ll A$
B
A
$B \ll A$
B
A
$=$
B
A
$A \gg B$
A
B
$A \gg B$
$=$
$A \gg B$
A
B
$A \gg B$
A
B
$=$
A
B
N^*
$=$
N
N
$=$
N^*



6.2.4 Sequentialization

The process of sequentialization for dual proof nets is obviously similar to that for proof nets. The sequentialization process is indicated in Figure 6.1. We get the following lemmas for free:

Lemma 6.1. *Every dual proof net sequentializes.*

Lemma 6.2. *Any two equal dual proof nets have equivalent sequentializations.*

6.2.5 Obtaining Categorical Morphisms

Translating from sequent proofs to categorical morphisms in a bi-open tensor category now becomes almost trivial in the light of duality. The formula interpretation of sequents is already defined, and we just note the following lemma:

Lemma 6.3. *For every context* $\Gamma[]$ *and morphism* $f : A \to \Delta^\circ$ *there is a morphism* $op(f) : \Gamma[A]^\circ \to \Gamma[\Delta]^\circ$.

In Figure 6.2 the translation is given. From it, we immediately get the following lemma:

Lemma 6.4. *Every equivalence between dual sequent proofs* $D_1 \equiv D_2$ *becomes equality of morphisms* $T(D_1) = T(D_2)$.

6.2.6 The Category of Dual Proof Nets

Like in chapter 2, we are now ready to define and state a freeness theorem about the category of dual proof nets.

Definition 6.5. Let $\Sigma = (\Sigma_0, \Sigma_1, dom, cod)$ be a bi-open tensor signature. We define the category of dual proof nets over Σ, denoted $\mathbf{DPN}(\Sigma)$, as follows:

- $Ob(\mathbf{PN}(\Sigma)) = \Sigma_0$,
- For every A in Σ_0, we define the identity morphism as

- For every f in Σ_1 with $dom(f) = A$ and $cod(f) = B$, we define the corresponding morphism as

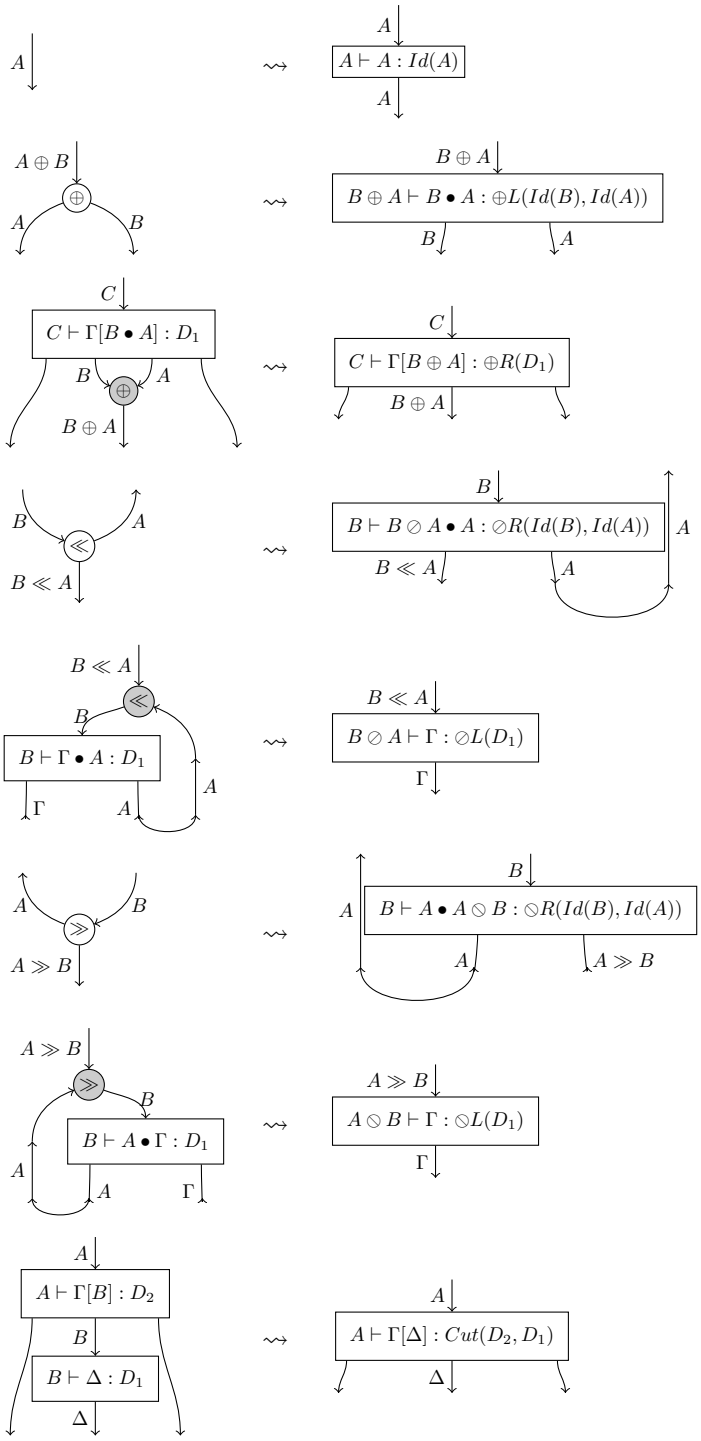


Figure 6.1: Sequentialization for dual proof nets

$$\frac{}{A \vdash A}\ Id \qquad \rightsquigarrow \qquad id_A : A \to A$$

$$\frac{A \vdash \Gamma[B] \quad B \vdash \Delta}{A \vdash \Gamma[\Delta]}\ Cut \qquad \rightsquigarrow \qquad \frac{f : A \to \Gamma[B]^\circ \quad g : B \to \Delta^\circ}{op(g) \circ f : A \to \Gamma[\Delta]^\circ}$$

$$\frac{C \vdash \Gamma[B \bullet A]}{C \vdash \Gamma[B \oplus A]}\ \oplus R \qquad \rightsquigarrow \qquad \frac{f : C \to \Gamma[B \bullet A]^\circ}{f : C \to \Gamma[B \bullet A]^\circ}$$

$$\frac{B \vdash \Delta \quad A \vdash \Gamma}{B \oplus A \vdash \Delta \bullet \Gamma}\ \oplus L \qquad \rightsquigarrow \qquad \frac{f : B \to \Delta \quad g : A \to \Gamma}{f \oplus g : B \oplus A \to (\Delta \bullet \Gamma)^\circ}$$

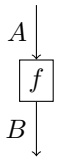
$$\frac{C \vdash \Gamma[A] \quad B \vdash \Delta}{C \vdash \Gamma[A \oslash B \bullet \Delta]}\ \oslash R \rightsquigarrow \qquad \frac{f : C \to \Gamma[A]^\circ \quad g : B \to \Delta^\circ}{op(\sigma^{-1}(id_A \ll g)) \circ f : C \to \Gamma[A \ll B \bullet \Delta]^\circ}$$

$$\frac{A \vdash \Gamma \bullet B}{A \oslash B \vdash \Gamma}\ \oslash L \qquad \rightsquigarrow \qquad \frac{f : A \to (\Gamma \bullet B)^\circ)}{\sigma(f) : A \ll B \to \Gamma^\circ}$$

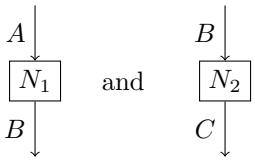
$$\frac{C \vdash \Gamma[A] \quad B \vdash \Delta}{C \vdash \Gamma[\Delta \bullet B \oslash A]}\ \oslash R \rightsquigarrow \qquad \frac{f : C \to \Gamma[A]^\circ \quad g : B \to \Delta^\circ}{op(\omega^{-1}(g \gg id_A)) \circ f : C \to \Gamma[\Delta \bullet B \gg A]^\circ}$$

$$\frac{A \vdash B \bullet \Gamma}{B \oslash A \vdash \Gamma}\ \oslash L \qquad \rightsquigarrow \qquad \frac{f : A \to (B \bullet \Gamma)^\circ}{\omega(f) : B \gg A \to \Gamma}$$

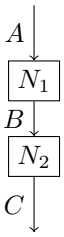
Figure 6.2: From Dual Sequent Proofs to Categorical Morphisms



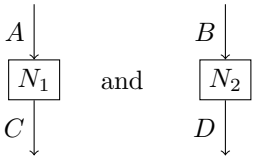
- We define composition of morphisms as vertical gluing, i.e. for morphisms



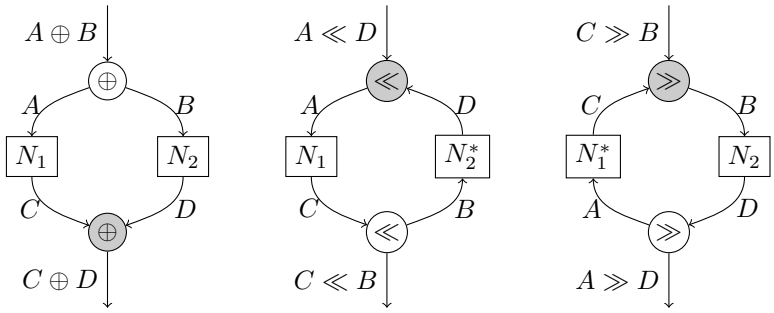
the following is their composition $N_2 \circ N_1$:



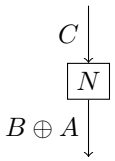
- For two morphisms



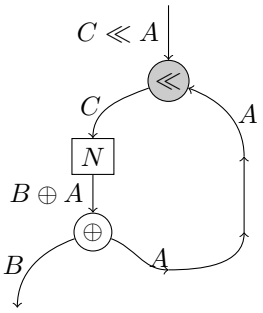
we define $N_1 \oplus N_2$, $N_1 \ll N_2$ and $N_1 \gg N_2$ as follows:



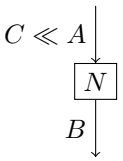
- Given a morphism



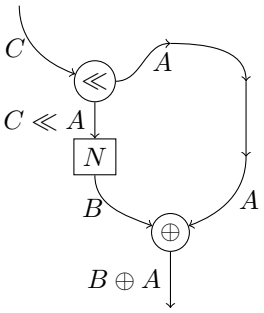
we define $\sigma(N)$ as



- Given a morphism



we define $\sigma^{-1}(N)$ as follows:



- We define ω and ω^{-1} similarly to σ and σ^{-1} but with the images mirrored horizontally,
- All the equations for dual proof nets hold.

We leave it to the reader to verify the following proposition:

Proposition 6.1. *For any bi-open tensor signature Σ, $\mathbf{DPN}(\Sigma)$ is a bi-open tensor category.*

We moreover get a freeness theorem for free:

Theorem 6.1. *For any bi-open tensor signature Σ, $\mathbf{DPN}(\Sigma)$ is the free bi-open tensor category over Σ.*

6.3 Proof Nets and Dual Proof Nets Combined

We now want to explore the possibily of proof nets for bi-clopen tensor categories. But, as there is no interaction between the $\otimes, \Rightarrow, \Leftarrow$ functors and the $\oplus, \ll, \gg$ functors, we can simply merge the definitions of proof nets and dual proof nets and get the following correction criteria:

Definition 6.6 (Planarity)**.** A proof structure is planar when it contains no crossing links.

Definition 6.7 (External Face Requirement)**.** A planar proof structure satisfies the external face requirement if its unique input and output wires are on the external face of the graph.

Definition 6.8 (Operator Balance)**.** A proof structure satisfies operator balance if and only if every cycle contains an equal number of $\otimes, \Rightarrow, \Leftarrow$ tensor and cotensor nodes and an equal number of $\oplus, \ll, \gg$ tensor and cotensor nodes.

Definition 6.9 (Return Cycle Requirement)**.** A proof structure satisfies the return cycle requirement precisely when all of the following requirements are met:

1. For every $\Rightarrow$ cotensor node, there is a directed path from the node through its left output, returning at the node,
2. For every $\Leftarrow$ cotensor node, there is a directed path from the node through its right output, returning at the node,
3. For every $\otimes$ cotensor node, there is no directed path from the node through one of its outputs returning at the node,
4. For every $\ll$ cotensor node, there is a directed path from the node through its left output, returning at the node,
5. For every $\gg$ cotensor node, there is a directed path from the node through its right output, returning at the node,
6. For every $\oplus$ cotensor node, there is no directed path from the node through its only output, returning at the node.

Definition 6.10 (Bi-Proof Net)**.** A proof structure is a bi-proof net iff it satisfies planarity, operator balance and the return cycle requirement.

We can now simply state that there is an inductive definition of bi-proof nets, it is namely the merging of Definitions 2.16 and 6.4.

A sequent calculus for bi-clopen tensor categories would be the sequent calculus given in chapter 2 combined with the sequent calculus given in this chapter. We then get sequentialization and the freeness theorem for the category of bi-proof nets for free. Things become somewhat more opaque, however, when we start thinking about incorporating linear distributive laws, as the next section shows.

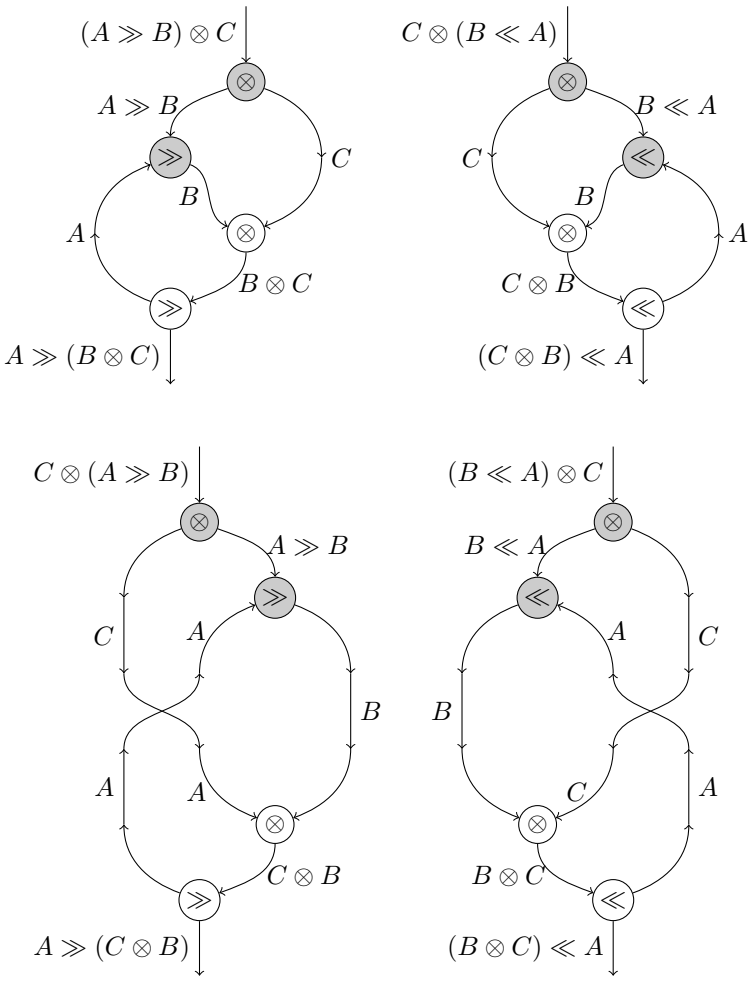
6.4 Graphical Languages for Linearly Distributive Categories

Up to this point, we have not really seen any new ideas in the landscape of graphical languages. But it's time to consider linear distributive laws and how these should affect our notion of bi-proof nets.

We have decided to keep the proof net calculus as modular as possible. Hence, we introduce a new proof net for every linear distributive law. Strictly speaking, these nets are not proof nets because they violate operator balance and additionally may violate planarity. Therefore, we should give them a special treatment when considering sequentialization.

Definition 6.11. The class of linear distributive proof nets is generated inductively as:

- All bi-proof nets are linear distributive proof nets,
- The following are linear distributive proof nets:

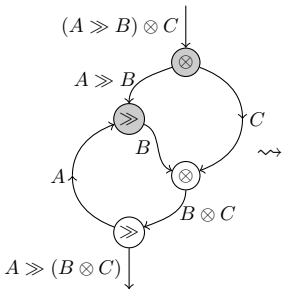


In order to be able to interpret these new nets, we add the following axioms to the sequent calculus for bi-proof nets:

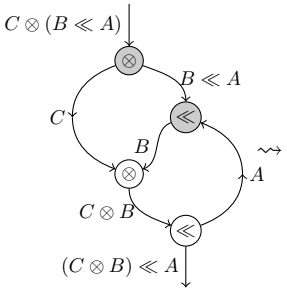
$$\frac{}{(A \oslash B) \otimes C \vdash A \oslash (B \otimes C)}\ D1 \qquad \frac{}{C \otimes (B \oslash A) \vdash (C \otimes B) \oslash A}\ D2$$

$$\frac{}{C \otimes (A \oslash B) \vdash A \oslash (C \otimes B)}\ D3 \qquad \frac{}{(B \oslash A) \otimes C \vdash (B \otimes C) \oslash A}\ D4$$

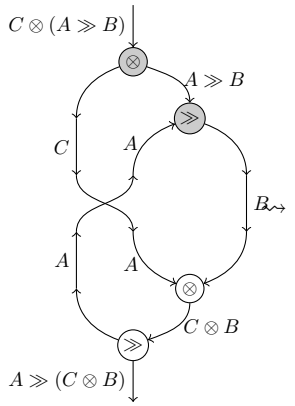
We simply extend our sequentialization by the following rules:



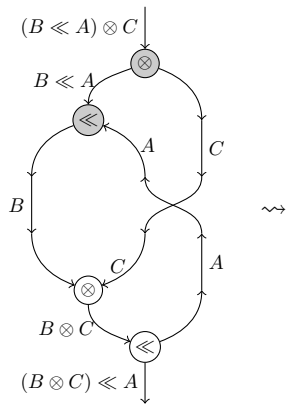
$\rightsquigarrow$ $\dfrac{}{(A \oslash B) \otimes C \vdash A \oslash (B \otimes C)}\ D1$



$\rightsquigarrow$ $\dfrac{}{C \otimes (B \oslash A) \vdash (C \otimes B) \oslash A}\ D2$



$$\rightsquigarrow \qquad \frac{}{C \otimes (A \otimes B) \vdash A \otimes (C \otimes B)}\ D3$$



$$\rightsquigarrow \qquad \frac{}{(B \oslash A) \otimes C \vdash (B \otimes C) \oslash A}\ D4$$

We then simply send the four distributive sequent axioms to the four distributive laws in the linear distributive laws, via the following map:

$$\frac{}{(A \otimes B) \otimes C \vdash A \otimes (B \otimes C)}\ D1 \qquad \rightsquigarrow \qquad \delta^L : (A \gg B) \otimes C \to A \gg (B \otimes C)$$

$$\frac{}{C \otimes (B \oslash A) \vdash (C \otimes B) \oslash A}\ D2 \qquad \rightsquigarrow \qquad \delta^R : C \otimes (B \ll A) \to (C \otimes B) \ll A$$

$$\frac{}{C \otimes (A \otimes B) \vdash A \otimes (C \otimes B)}\ D3 \qquad \rightsquigarrow \qquad \kappa^L : C \otimes (A \gg B) \to A \gg (C \otimes B)$$

$$\frac{}{(B \oslash A) \otimes C \vdash (B \otimes C) \oslash A}\ D4 \qquad \rightsquigarrow \qquad \kappa^R : (B \ll A) \otimes C \to (B \otimes C) \ll A)$$

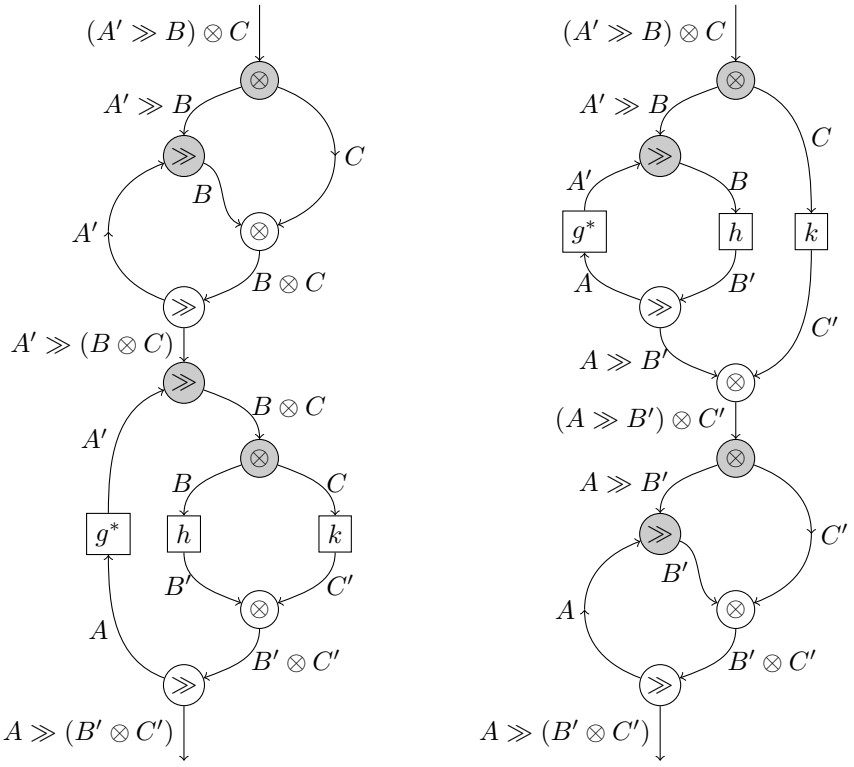
Given a bi-clopen tensor signature Σ we claim that $\mathbf{LDPN}(\Sigma)$, the system obtained by merging $\mathbf{PN}(\Sigma)$ and $\mathbf{DPN}(\Sigma)$ and adding the four linear distributive nets, is in fact a linear distributive category:

Proposition 6.2. *For any bi-clopen tensor signature Σ, $\mathbf{LDPN}(\Sigma)$ is a linear distributive category.*

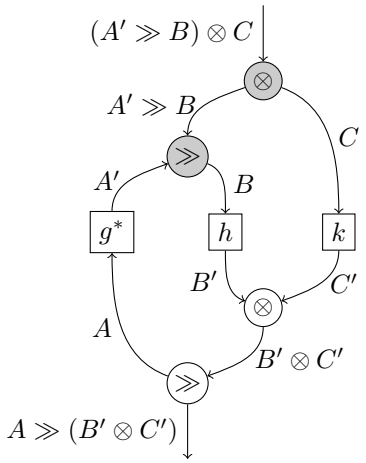
Proof. We only need to check naturality of the distributivity nets, i.e. that the following equations hold graphically:

$$(g \gg (h \otimes k)) \circ \delta^L = \delta^L \circ ((g \gg h) \otimes k)$$
$$(g \gg (k \otimes h)) \circ \kappa^L = \kappa^L \circ (k \otimes (g \gg h))$$
$$((k \otimes h) \ll g) \circ \delta^R = \delta^R \circ (k \otimes (h \ll g))$$
$$((h \otimes k) \ll g) \circ \kappa^R = \kappa^R \circ ((h \ll g) \otimes k)$$
for $g : A \to A'$, $h : B \to B'$ and $k : C \to C'$.

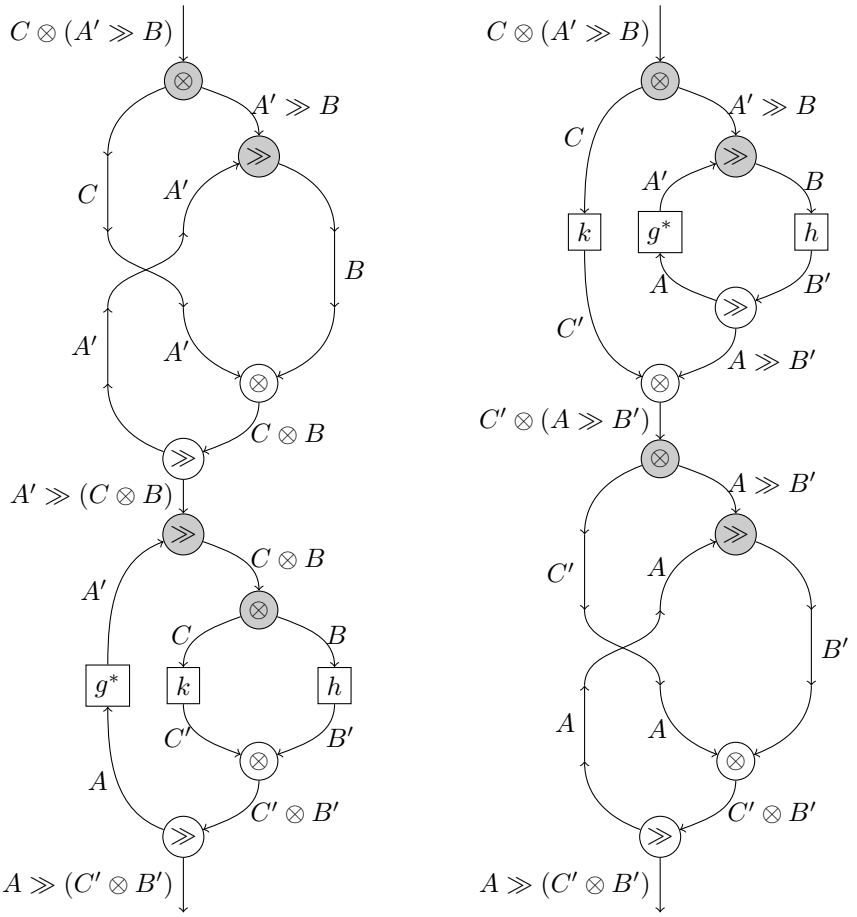
Since the latter two equations are symmetric to the first two, we will only illustrate the first two equations. For the first equation, we note that the graphical representations of $(g \gg (h \otimes k)) \circ \delta^L$ and $\delta^L \circ ((g \gg h) \otimes k)$ are as follows:



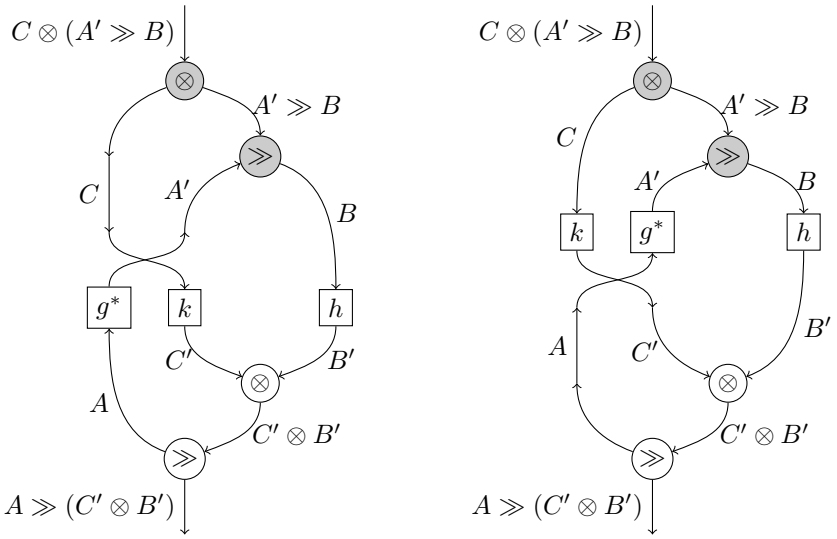
And it is not very hard to see that both nets are equal to the following net:



For the second equation, we draw the graphical representations of $(g \gg (k \otimes h)) \circ \kappa^L$ and $\kappa^L \circ (k \otimes (g \gg h))$:



These nets reduce respectively to the following:



By sliding the g^* and k over the crossing, we get that these nets are equal. □

We then get the following freeness theorem:

Theorem 6.2. *For any linear distributive signature* Σ*,* $\mathbf{LDPN}(\Sigma)$ *is the free linear distributive category over* Σ*.*

Chapter 7

A New Syntax

In this chapter, we consider dual variants of the Lambek Calculi, which we will call Grishin Calculi. Combining the two variants gives the Lambek-Grishin Calculus, earlier explored in (Moortgat, 2009). We will again depict these logics as deductive systems in order to allow a natural categorification of the systems considered. We will then see that this categorical interpretation gives rise to new equivalences of categories.

7.1 Grishin and Lambek-Grishin Calculi, Categorically

Grishin Calculi are obtained by dualizing the patterns in Lambek Calculi. Instead of residuation laws that, for instance, turn a proof $f : A \otimes B \to C$ into a proof $\triangleleft f : B \to A \backslash C$, one considers *dual residuation laws* that would turn a proof of $f : C \to B \otimes A$ into a proof $\blacktriangleright f : C \oslash A \to B$. Hence, we consider the sets of free types generated by the *dual connectives* $\oplus$, $\oslash$ and $\obslash$, so in the following definitions, T is a subset of $\{\oplus, \oslash, \obslash\}$.

Definition 7.1. The (non-associative, non-unitary) right Grishin Calculus $\mathbf{NG}^r$ over T is given by the types in $F(T, \{\oplus, \oslash\})$ and the proofs generated by the following deductive system:

$$\frac{}{1_A : A \to A}\ Ax \qquad \frac{f : A \to B \quad g : B \to C}{g \circ f : A \to C}\ T$$

$$\frac{f : C \to B \oplus A}{\blacktriangleright f : C \oslash A \to B}\ CR2 \qquad \frac{g : C \oslash A \to B}{\blacktriangleright^{-1} g : C \to B \oplus A}\ CR2'$$

$$\frac{f : A \to C \quad g : B \to D}{f \oplus g : A \oplus B \to C \oplus D}\ M_\oplus \qquad \frac{f : A \to C \quad g : B \to D}{f \oslash g : A \oslash D \to C \oslash B}\ M_\oslash$$

Definition 7.2. The (non-associative, non-unitary) left Grishin Calculus $\mathbf{NG}^l$ is given by the types in $F(T, \{\oplus, \obslash\})$ and the proofs generated by the following deductive system:

$$\frac{}{1_A : A \to A}\ Ax \qquad \frac{f : A \to B \quad g : B \to C}{g \circ f : A \to C}\ T$$

$$\frac{f : C \to B \oplus A}{\blacktriangleleft f : B \oslash C \to A}\ CR1 \qquad \frac{g : B \oslash C \to A}{\blacktriangleleft^{-1} g : C \to B \oplus A}\ CR1'$$

$$\frac{f : A \to C \quad g : B \to D}{f \oplus g : A \oplus B \to C \oplus D}\ M_\oplus \qquad \frac{f : A \to C \quad g : B \to D}{g \oslash f : D \oslash A \to B \oslash C}\ M_\oslash$$

Again, we can derive monotonicity when merging the $\oslash$ and $\oslash$ connectives:

Definition 7.3. The (non-associative, non-unitary) Grishin Calculus **NG** over is given by the types in $F(T, \{\oplus, \oslash, \oslash\})$ and the proofs generated by the following deductive system:

$$\frac{}{1_A : A \to A}\ Ax \qquad \frac{f : A \to B \quad g : B \to C}{g \circ f : A \to C}\ T$$

$$\frac{f : C \to B \oplus A}{\blacktriangleleft f : B \oslash C \to A}\ CR1 \qquad \frac{f : C \to B \oplus A}{\blacktriangleright f : C \oslash A \to B}\ CR2$$

$$\frac{g : B \oslash C \to A}{\blacktriangleleft^{-1} g : C \to B \oplus A}\ CR1' \qquad \frac{g : C \oslash A \to B}{\blacktriangleright^{-1} g : C \to B \oplus A}\ CR2'$$

Now we define

$$\begin{aligned} f \oplus g :=& \blacktriangleleft^{-1} (g \circ (\blacktriangleleft\blacktriangleright^{-1} (f \circ (\blacktriangleright 1_{A \oplus B})))) \\ f \oslash g :=& \blacktriangleright ((\blacktriangleleft^{-1} (g \circ (\blacktriangleleft\blacktriangleright^{-1} 1_{C \oslash B}))) \circ f) \\ g \oslash f :=& \blacktriangleleft ((\blacktriangleright^{-1} (g \circ (\blacktriangleright\blacktriangleleft^{-1} 1_{B \oslash C}))) \circ f) \end{aligned}$$

And we obtain monotonicity rules as derived rules of inference:

$$\frac{f : A \to C \quad g : B \to D}{f \oplus g : A \otimes B \to C \otimes D}\ M_\oplus$$

$$\frac{f : A \to C \quad g : B \to D}{f \oslash g : A \oslash D \to C \oslash B}\ M_\oslash$$

$$\frac{f : A \to C \quad g : B \to D}{g \oslash f : D \oslash A \to B \oslash C}\ M_\oslash$$

Again, we might add associativity and a unit object. This amounts to adding the following rules:

$$\frac{}{{}_\oplus a_{A,B,C} : (A \oplus B) \oplus C \to A \oplus (B \oplus C)}\ Ass_\oplus$$

$$\frac{}{{}_\oplus a^{-1}_{A,B,C} : A \oplus (B \oplus C) \to (A \oplus B) \oplus C}\ Ass'_\oplus$$

and/or adding a unit object J to the set of basic types T and consequently adding the following inference rules:

$$\frac{}{{}_{\oplus}l_A : J \oplus A \to A}\ L_{\oplus} \qquad \frac{}{{}_{\oplus}l_A^{-1} : A \to J \oplus A}\ L'_{\oplus}$$

$$\frac{}{{}_{\oplus}r_A : A \oplus J \to A}\ R_{\oplus} \qquad \frac{}{{}_{\oplus}r_A^{-1} : A \to A \oplus J}\ R'_{\oplus}$$

We now want to explore the possibility of merging the Lambek Calculus and the Grishin Calculus. This is called the Lambek-Grishin Calculus and was already explored in (Moortgat, 2009).

Definition 7.4. The (non-associative, non-unitary) Lambek-Grishin Calculus $\mathbf{LG}_\emptyset$ is given by the types in $F(T, \{\otimes, \backslash, /, \oslash, \obslash\})$ and the proofs generated by the following deductive system:

$$\frac{}{1_A : A \to A}\ Ax \qquad \frac{f : A \to B \quad g : B \to C}{g \circ f : A \to C}\ T$$

$$\frac{f : A \otimes B \to C}{\triangleright f : A \to C/B}\ R1 \qquad \frac{f : A \otimes B \to C}{\triangleleft f : B \to A\backslash C}\ R2$$

$$\frac{g : A \to C/B}{\triangleright^{-1} g : A \otimes B \to C}\ R1' \qquad \frac{g : B \to A\backslash C}{\triangleleft^{-1} g : A \otimes B \to C}\ R2'$$

$$\frac{f : C \to B \oplus A}{\blacktriangleleft f : B \oslash C \to A}\ CR1 \qquad \frac{f : C \to B \oplus A}{\blacktriangleright f : C \obslash A \to B}\ CR2$$

$$\frac{g : B \oslash C \to A}{\blacktriangleleft^{-1} g : C \to B \oplus A}\ CR1' \qquad \frac{g : C \obslash A \to B}{\blacktriangleright^{-1} g : C \to B \oplus A}\ CR2'$$

Again, the monotonicity rules are derived using exactly the same recipes as before. Also, one could add associativity for either of the $\otimes, \oplus$ connectives, or add a unit object together with the appropriate deduction rules.

We will now consider extending $\mathbf{LG}_\emptyset$ with so-called *weak distributivity* laws. However, a note of caution is in its place here: for linguistics purposes, one might not want to admit associativity of either the $\otimes$ of the $\oplus$ connective. Furthermore, commutativity is definitely an undesirable property. We will consider the following laws, henceforth referred to as *type IV interactions*:

$$\begin{array}{ll} \delta^{\oslash} : (A \oslash B) \otimes C \to A \oslash (B \otimes C) & \delta^{\obslash} : C \otimes (B \obslash A) \to (C \otimes B) \obslash A \\ \kappa^{\oslash} : C \otimes (A \oslash B) \to A \oslash (C \otimes B) & \kappa^{\obslash} : (B \obslash A) \otimes C \to (B \otimes C) \obslash A \end{array}$$

as well as their converses, i.e.

$$\begin{array}{ll} \delta'^{\oslash} : A \oslash (B \otimes C) \to (A \oslash B) \otimes C & \delta'^{\obslash} : (C \otimes B) \obslash A \to C \otimes (B \obslash A) \\ \kappa'^{\oslash} : A \oslash (C \otimes B) \to C \otimes (A \oslash B) & \kappa'^{\obslash} : (B \otimes C) \obslash A \to (B \obslash A) \otimes C \end{array}$$

However, as we will show, adding all eight laws leads to undesirable properties: near-associativity and near-commutativity are present for the individual $\otimes$ and $\oplus$ connectives when all laws are present. A few cases for $\oplus$ are shown in Figure 7.1.

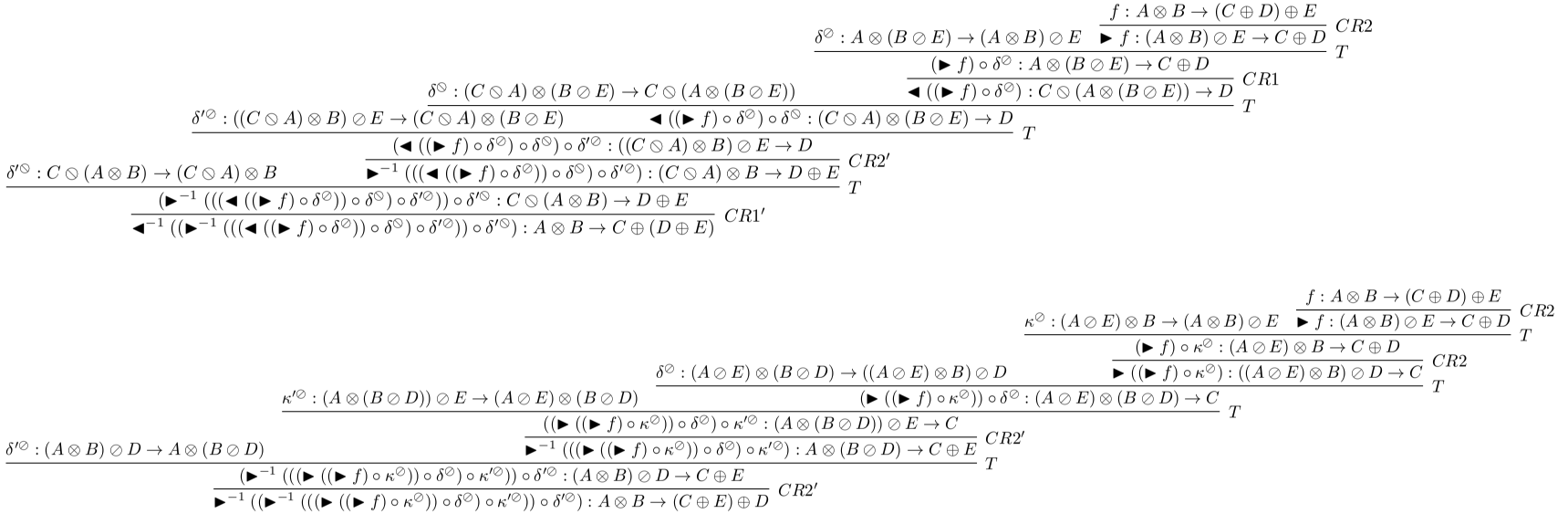


Figure 7.1: Near-associativity and near-commutativity for $\oplus$

All in all, we get the following derivable two-way inference rules (see also (Bastenhof, 2013)):

$$\frac{(A \otimes B) \otimes C \to D \oplus E}{A \otimes (B \otimes C) \to D \oplus E} \quad \frac{(A \otimes B) \otimes C \to D \oplus E}{(A \otimes C) \otimes B \to D \oplus E} \quad \frac{A \otimes (B \otimes C) \to D \oplus E}{B \otimes (A \otimes C) \to D \oplus E}$$

$$\frac{A \otimes B \to (C \oplus D) \oplus E}{A \otimes B \to C \oplus (D \oplus E)} \quad \frac{A \otimes B \to (C \oplus D) \oplus E}{A \otimes B \to (C \oplus E) \oplus D} \quad \frac{A \otimes B \to C \oplus (D \oplus E)}{A \otimes B \to D \oplus (C \oplus E)}$$

7.1.1 Categorification

The process of categorification for Grishin Calculi is completely analogous to that of Lambek Calculi. Thus, we construct an equivalence relation using the following equivalences:

1. Substitution properties: If $f \equiv g$ then $*f \equiv *g$ where $* \in \{\blacktriangleleft, \blacktriangleleft^{-1}, \blacktriangleright, \blacktriangleright^{-1}\}$, and if $f \equiv g$ and $k \equiv h$ then $k \circ f = h \circ g$. A resulting property is that if $f \equiv g$ and $k \equiv h$ then $f * k \equiv g * h$ where $* \in \{\oplus, \oslash, \oslash\}$.

2. Categorical axioms: this amounts to $f \circ (g \circ h) \equiv (f \circ g) \circ h$ and $f \circ 1_A \equiv f \equiv 1_B \circ f$ where $f : A \to B$.

3. Coresiduation is isomorphic: this amounts to $*^{-1} * f \equiv f$, $* *^{-1} g \equiv g$ for $* \in \{\blacktriangleleft, \blacktriangleright\}$.

4. Bifunctoriality of $\oplus, \oslash, \oslash$. One can easily show that $1_A * 1_B \equiv 1_{A*B}$ where $* \in \{\oplus, \oslash, \oslash\}$ but the following equivalences are defined:

$$(f \oplus g) \circ (k \oplus h) \equiv (f \circ k) \oplus (g \circ h)$$
$$(h \oslash k) \circ (g \oslash f) \equiv (g \circ h) \oslash (k \circ f)$$
$$(h \oslash k) \circ (g \oslash f) \equiv (h \circ g) \oslash (f \circ k)$$

5. Naturality of coresiduation. This means that for $f : C \to B' \oplus A'$ and $g : A' \to A$, $h : B' \to B$, $k : C' \to C$, we have that

$$(g \circ (\blacktriangleleft f)) \circ (h \oslash k) \equiv \blacktriangleleft (((h \oplus g) \circ f) \circ k) \text{ and}$$
$$(h \circ (\blacktriangleright f)) \circ (k \oslash g) \equiv \blacktriangleright (((h \oplus g) \circ f) \circ k).$$

6. For $\mathbf{G}^{(l/r)}$, one adds the following requirements:

   (a) Associativity is isomorphic, i.e. ${}_{\oplus}a^{-1}_{A,B,C} \circ {}_{\oplus}a_{A,B,C} \equiv 1_{A\oplus(B\oplus C)}$ and ${}_{\oplus}a_{A,B,C} \circ {}_{\oplus}a^{-1}_{A,B,C} \equiv 1_{(A\oplus B)\oplus C}$ for every A, B, C,

   (b) Associativity is natural, i.e. $(f \oplus (g \oplus h)) \circ {}_{\oplus}a_{A,B,C} \equiv {}_{\oplus}a_{A',B',C'} \circ ((f \oplus g) \oplus h)$ for $f : A \to A'$,$g : B \to B'$, $h : C \to C'$.

7. For $\mathbf{UG}^{(l/r)}$, one adds the following requirements:

   (a) Unit deletion is isomorphic, i.e. ${}_{\oplus}l^{-1}_{A} \circ {}_{\oplus}l_A \equiv 1_{J\oplus A}$, ${}_{\oplus}l_A \circ {}_{\oplus}l^{-1}_{A} \equiv 1_A$, ${}_{\oplus}r^{-1}_{A} \circ {}_{\oplus}r_A \equiv 1_{A\oplus J}$ and ${}_{\oplus}r_A \circ {}_{\oplus}r^{-1}_{A} \equiv 1_A$ for every A,

   (b) Unit deletion is natural, i.e. for every $f : A \to A'$ we have that $f \circ {}_{\oplus}l_A \equiv {}_{\oplus}l_{A'} \circ (1_J \oplus f)$ and $f \circ {}_{\oplus}r_A \equiv {}_{\oplus}r_{A'} \circ (f \oplus 1_J)$.

To obtain the categorification of $\mathbf{LG}_\emptyset$, one simply merges the equivalence relations defined for **NL** and **NG**. When one adds weak distributive laws, one must ensure that these are natural, hence for type IV distributivities one adds

$$(g \oslash (h \otimes k)) \circ \delta^{\oslash} \equiv \delta^{\oslash} \circ ((g \oslash h) \otimes k)$$
$$(g \oslash (k \otimes h)) \circ \kappa^{\oslash} \equiv \kappa^{\oslash} \circ (k \otimes (g \oslash h))$$
$$((k \otimes h) \oslash g) \circ \delta^{\oslash} \equiv \delta^{\oslash} \circ (k \otimes (h \oslash g))$$
$$((h \otimes k) \oslash g) \circ \kappa^{\oslash} \equiv \kappa^{\oslash} \circ ((h \oslash g) \otimes k)$$

The resulting category, definitely a linear distributive category, is denoted $\mathbf{cLG}_{IV}$.

7.2 Another Equivalence

It is not hard to define translations of Grishin Calculi, whether they are associative or not, and whether they are unitary or not. By duality, we thus get the following propositions:

Proposition 7.1. $\mathbf{cNG}^{(l/r)} \cong (\mathbf{L/R/B})\mathbf{OC}_{st}$.

Proposition 7.2. $\mathbf{cG}^{(l/r)} \cong \mathbf{A}(\mathbf{L/R/B})\mathbf{OC}_{st}$.

Proposition 7.3. $\mathbf{cUG}^{(l/r)} \cong \mathbf{M}(\mathbf{L/R/B})\mathbf{OC}_{st}$.

We are now in a position to show another equivalence result, namely between the category of Lambek-Grishin Calculi with type IV interactions and the category of linear distributive categories. To this end, we define translations of Lambek-Grishin Calculi as follows:

Given two Lambek-Grishin Calculi $\mathbf{LG}^{IV}_T$ and $\mathbf{LG}^{IV}_{T'}$, a translation is a map t that sends T to $F(T', \{\otimes, /, \backslash, \oplus, \oslash, \oslash\})$ and sends proofs to proofs such that the following holds:

- t strictly preserves the type operations, i.e. $t(A \otimes B) = t(A) \otimes t(B)$, $t(B/A) = t(B)/t(A)$, $t(A\backslash B) = t(A)\backslash t(B)$, $t(B \oplus A) = t(B) \oplus t(A)$, $t(A \oslash B) = t(A) \oslash t(B)$, $t(B \oslash A) = t(B) \oslash t(A)$,
- t preserves the structural axioms, i.e. $t(\delta^{\oslash}) = \delta^{\oslash}$, $t(\delta^{\oslash}) = \delta^{\oslash}$, $t(\kappa^{\oslash}) = \kappa^{\oslash}$, $t(\kappa^{\oslash}) = \kappa^{\oslash}$,
- t preserves the equivalence relation on proofs, i.e. if $f \equiv g$ in $\mathbf{LG}^{IV}_T$ then $t(f) \equiv t(g)$ in $\mathbf{LG}^{IV}_{T'}$.

Like in Chapter 3, we can form a category by considering all Lambek-Grishin Calculi with type IV distributivities as objects and translations between them as morphisms. We will denote this category by $\mathbf{cLG}^{IV}$. As the objects of this category are all linear distributive tensor categories and the translations are (strict) linear distributive functors, we already have one direction of the *Curry-Howard-Lambek* correspondence for Lambek-Grishin systems and linear distributive tensor categories. Given such a linear distributive tensor category $(\mathbf{C}, \otimes, \oplus, \Rightarrow, \Leftarrow, \ll, \gg, \beta, \gamma, \sigma, \omega, \delta^L, \delta^R, \kappa^L, \kappa^R)$ we can easily generate a Lambek-Grishin system as the system $\mathbf{LG}^{IV}_{Ob(\mathbf{C})}$. That is,

- The set of types T is precisely $Ob(\mathbf{C})$ under the following object-to-type interpretation:

$$
\begin{array}{lll}
i(A) & = & A \\
i(A \otimes B) & = i(A) \otimes i(B) & \\
i(A \Rightarrow B) & = & i(A) \backslash i(B) \\
i(B \Leftarrow A) & = & i(B) / i(A) \\
i(B \oplus A) & = & i(B) \oplus i(A) \\
i(B \ll A) & = & i(B) \oslash i(A) \\
i(A \gg B) & = & i(A) \osl i(B)
\end{array}
$$

- Morphisms of $\mathbf{C}$ are mapped to proofs by the following interpretation:

$$
\begin{array}{lcl}
j(id_A) & = & 1_{i(A)} \\
j(g \circ f) & = & j(g) \circ j(f) \\
j(f \otimes g) & = & j(f) \otimes j(g) \\
j(f \Rightarrow g) & = & j(f) \backslash j(g) \\
j(g \Leftarrow f) & = & j(g) / j(f) \\
j(g \oplus f) & = & j(g) \oplus j(f) \\
j(g \ll f) & = & j(g) \oslash j(f) \\
j(f \gg g) & = & j(f) \osl j(g) \\
j(\beta(f)) & = & \lhd(j(f)) \\
j(\gamma(f)) & = & \rhd(j(f)) \\
j(\sigma(g)) & = & \blacktriangleright (j(g)) \\
j(\omega(g)) & = & \blacktriangleleft (j(g)) \\
j(\delta^L) & = & \delta^{\osl} \\
j(\delta^R) & = & \delta^{\oslash} \\
j(\kappa^L) & = & \kappa^{\osl} \\
j(\kappa^R) & = & \kappa^{\oslash}
\end{array}
$$

  and for any additional map $f : A \to B$ in $\mathbf{C}$, there is a proof $j(f) : j(A) \to j(B)$.

Let $\mathbf{LDTC}_{st}$ denote the category of linear distributive tensor categories with strict linear distributive tensor functors as morphisms. Now, given that any translation between two Lambek-Grishin systems is a strict functor, and for any strict functor $F : \mathbf{C} \to \mathbf{D}$ where $\mathbf{C}$ and $\mathbf{D}$ are linear distributive tensor categories, there is the obvious translation F' between the associated Lambek-Grishin systems $\mathbf{LG}^{IV}_{\mathbf{C}}$ and $\mathbf{LG}^{IV}_{\mathbf{D}}$, which acts precisely like F but under the interpretation of $\mathbf{C}$ and $\mathbf{D}$ into deductive systems, we get the following proposition:

Proposition 7.4. $\mathbf{cLG}^{IV} \cong \mathbf{LDTC}_{st}$.

7.3 Grammars

In this section, we define Grishin and Lambek-Grishin grammars and note some results on generative capacity of such systems.

Definition 7.5. Given an alphabet Σ and a Grishin calculus G of choice (with or without associativity and/or units) where $F(T, C)$ is the set of types of G, a dictionary is a relation $\delta \subseteq \Sigma \times F(T, C)$.

Definition 7.6 (Grishin Grammar)**.** A Grishin grammar over a Grishin calculus G is a triple (Σ, δ, S) where

- Σ is an alphabet,
- δ is a dictionary over Σ and G,
- $S \in F(T, C)$ is a distinguished start symbol.

Definition 7.7. Given a Grishin grammar (Σ, δ, S) over G, a sequence of words $w_1 \cdot ... \cdot w_n \in \Sigma^+$ is a sentence if and only if there exists a sequence $W_1 \oplus ... \oplus W_n$ (where each $W_i \in \delta(w_i)$) such that there is a proof in the system G of $S \to W_1 \oplus ... \oplus W_n$.

Definition 7.8. Given a Grishin grammar $G = (\Sigma, \delta, S)$ over G, the language of G is defined as

$$(G) = \{w \in \Sigma^+ \mid w \text{ is a sentence of } G\}.$$

Definition 7.9. The class of languages generated by a Grishin Calculus G is defined as the class containing all languages generated by some Grishin Grammar over G.

We then get the following theorem by symmetry:

Theorem 7.1. *The class of languages generated by any Grishin grammar G coincides with the context-free languages.*

Sketch. Given that the class of languages generated by any Lambek grammar L coincides with the context-free languages, consider such a Lambek grammar $L = (\Sigma, \delta, S)$. Applying the arrow-preserving left/right symmetry and then the arrow-reversing dualizing symmetry to the types in δ and to S, we get that exactly the same language will be generated by the resulting Grishin grammar. □

Only the definition of sentencehood will be different when considering Lambek-Grishin systems. The other definitions can be obtained by replacing the Grishin calculi G by the Lambek-Grishin calculi $LG^{(IV)}$:

Definition 7.10. Given a Lambek-Grishin grammar (Σ, δ, S) over $LG^{(IV)}$, a sequence of words $w_1 \cdot ... \cdot w_n \in \Sigma^+$ is a sentence if and only if there exists a sequence $W_1 \otimes ... \otimes W_n$ (where each $W_i \in \delta(w_i)$) such that there is a proof in the system $LG^{(IV)}$ of $W_1 \otimes ... \otimes W_n \to S$.

Note that we could have also required the type sequence to be of the form $W_1 \oplus ... \oplus W_n$ and then requiring that there be a proof of $S \to W_1 \oplus ... \oplus W_n$. We chose the former option, however.

As to generative capacity, it turns out that the Lambek-Grishin base logic (that is, the system without interactions) has the same recognizable class of languages as the Lambek Calculus or Grishin Calculus. This might be not so obvious, but becomes clear when one considers that there is no way to relate the Lambek connectives with the Grishin connectives, and hence a base Lambek-Grishin Calculus may at most recognize the union of two context-free languages, but the latter class is closed under union. So, we have the following theorem due to Bastenhof (2010):

Theorem 7.2. *The class of languages generated by any Lambek-Grishin calculus LG coincides with the context-free languages.*

Considering the Lambek-Grishin Calculus enriched with interaction postulates, we get that the recognizing capacity of the corresponding grammars exceeds that of the context-free languages. It was already shown by Moot (2007) that Tree Adjoining Grammars, which recognize mildly context-sensitive languages, can be modelled by the interaction maps. However, the generative capacity of the Lambek-Grishin system was two years later shown to even exceed the class of Tree Adjoining Languages by Melissen (2011):

Theorem 7.3. *The class of languages generated by any Lambek-Grishin calculus with distributivities LG^{IV} properly includes the class of Tree Adjoining Languages.*

Now that we have explored the symmetric extension of the Lambek Calculus and have seen how it is categorically characterized, we will show in the next chapter how to embed these type logics in *extended compositional distributional models of meaning*.

Chapter 8

Semantics

In this chapter, we elaborate on the possibility of connecting Lambek-Grishin grammars to finite-dimensional vector spaces. As we have seen in the previous chapter, a Lambek-Grishin calculus with type IV interactions forms a linearly distributive tensor category. To interpret such a calculus, thus requires a semantic category that is at least a linearly distributive tensor category, possibly with some additional structure. We will show that finite-dimensional vector spaces exactly have these desired properties.

8.1 Vector Spaces as a Linearly Distributive Category

We have already seen how vector space semantics can be used to interpret derivation in a Lambek grammar, as finite-dimensional vector spaces with the tensor product form a bi-closed monoidal category (since they are compact closed). However, because of the self-duality of compact closed categories, it follows that **FVect** is automatically a bi-clopen tensor category where the two tensors coincide. Because the tensor is symmetric, the linear distributivities now can be defined in terms of the associativity and symmetry maps. Specifically, we have that the β and γ maps (and their inverses) of **FVect** are given by

$$\beta(f) := \overrightarrow{b} \mapsto \sum_i \overrightarrow{a_i} \otimes f(\overrightarrow{a_i} \otimes \overrightarrow{b})$$

$$\gamma(f) := \overrightarrow{a} \mapsto \sum_i f(\overrightarrow{a} \otimes \overrightarrow{b_i}) \otimes \overrightarrow{b_i}$$

$$\beta^{-1}(g) := \overrightarrow{a} \otimes \overrightarrow{b} \mapsto \sum_{ij} w_{ij} \langle \overrightarrow{a} | \overrightarrow{a_i'} \rangle \overrightarrow{c_j} \text{ where } g(\overrightarrow{b}) = \textstyle\sum_{ij} w_{ij} \overrightarrow{a_i'} \otimes \overrightarrow{c_j}$$

$$\gamma^{-1}(g) := \overrightarrow{a} \otimes \overrightarrow{b} \mapsto \sum_{ij} \overrightarrow{c_i} \langle \overrightarrow{b_j'} | \overrightarrow{b} \rangle w_{ij} \text{ where } g(\overrightarrow{a}) = \textstyle\sum_{ij} w_{ij} \overrightarrow{c_i} \otimes \overrightarrow{b_j'}$$

whereas the σ and ω maps and their inverses are given by

$$\sigma(f) := \overrightarrow{c} \otimes \overrightarrow{a} \mapsto \sum_{ij} \overrightarrow{b_i} \langle \overrightarrow{a_j'} | \overrightarrow{a} \rangle w_{ij} \text{ where } f(\overrightarrow{c}) = \textstyle\sum_{ij} w_{ij} \overrightarrow{b_i} \otimes \overrightarrow{a_j'}$$

$$\omega(f) := \overrightarrow{b} \otimes \overrightarrow{c} \mapsto \sum_{ij} w_{ij} \langle \overrightarrow{b} | \overrightarrow{b_i'} \rangle \overrightarrow{a_j} \text{ where } f(\overrightarrow{c}) = \textstyle\sum_{ij} w_{ij} \overrightarrow{b_i'} \otimes \overrightarrow{a_j}$$

$$\sigma^{-1}(g) := \overrightarrow{c} \mapsto \sum_i g(\overrightarrow{c} \otimes \overrightarrow{a_i}) \otimes \overrightarrow{a_i}$$

$$\omega^{-1}(g) := \overrightarrow{c} \mapsto \sum_i \overrightarrow{b_i} \otimes g(\overrightarrow{b_i} \otimes \overrightarrow{c})$$

and the linear distributivities are defined as follows:

$$\begin{array}{lcl} \delta^L & := & \alpha \\ \delta^R & := & \alpha^{-1} \\ \kappa^L & := & \alpha \circ (c_{C,A} \otimes id_B) \circ \alpha^{-1} \\ \kappa^R & := & \alpha^{-1} \circ (id_B \otimes c_{A,C}) \circ \alpha \end{array}$$

We then get the following proposition:

Proposition 8.1. *The category* **FVect** *of finite-dimensional vector spaces together with the tensor product $\otimes$, forms a linearly distributive monoidal category.*

Proof. We have already seen that **FVect** is a bi-clopen monoidal category by virtue of compact closure. We then only need to check that the additional coherence maps for the linear distributivities are satisfied, but these are routine checks of which we will show a few:

- The first coherence map for unit and distribution states that $\rho_{A\otimes B} = (id_A \otimes \rho_B) \circ \alpha_{A,B,I}$. But given a vector $\overrightarrow{a} \otimes \overrightarrow{b} \otimes r$ we have that applying $\rho_{A\otimes B}$ results in $r(\overrightarrow{a} \otimes \overrightarrow{b})$ whereas applying $(id_A \otimes \rho_B) \circ \alpha_{A,B,I}$ gives $\overrightarrow{a} \otimes r\,\overrightarrow{b}$ but these are the same.

- The first coherence map for associativity and distribution states that $\alpha_{A,B,(C\otimes D)} \circ \alpha_{(A\otimes B),C,D} = (id_A \otimes \alpha_{B,C,D}) \circ \alpha_{A,(B\otimes C),D} \circ (\alpha_{A,B,C} \otimes id_D)$ but the effect of applying the left map to $((\overrightarrow{a} \otimes \overrightarrow{b}) \overrightarrow{c}) \overrightarrow{d}$ is $\overrightarrow{a} \otimes (\overrightarrow{b} \otimes (\overrightarrow{c} \otimes \overrightarrow{d}))$ which is equal to the effect of the right map.
- The first coherence map for distribution and distribution states that $\alpha_{A,C,(B\otimes D)} \circ (c_{C,A} \otimes id_{B\otimes D}) \circ \alpha^{-1}_{C,A,(B\otimes D)} \circ (id_C \otimes \alpha_{A,B,D}) \circ \alpha_{C,(A\otimes B),D} = (id_A \otimes \alpha_{C,B,D}) \circ \alpha_{A,(C\otimes B),D} \circ ((\alpha_{A,C,B} \circ (c_{C,A} \otimes id_B) \circ \alpha^{-1}_{C,A,B}) \otimes id_D)$ but the effect of applying the left map to $(\overrightarrow{c} \otimes (\overrightarrow{a} \otimes \overrightarrow{b})) \otimes \overrightarrow{d}$ is $\overrightarrow{a} \otimes (\overrightarrow{c} \otimes (\overrightarrow{b} \otimes \overrightarrow{d}))$ which is also obtained when applying the right map.

□

Now that we have established that **FVect** is a linear distributive monoidal category, we have ensured that grammars based on the Lambek-Grishin Calculus with type IV interactions are interpretable in finite-dimensional vector spaces and thus we are ready to define extended compositional distributional models of meaning, which we will do in the next section.

8.2 Interpreting the Lambek-Grishin Calculus

To begin with, we define a semantic signature and interpretation for a Lambek-Grishin grammar completely analogous to the case of Lambek grammars.

Definition 8.1. A semantic signature over a linearly distributive category $\mathbf{C}$ is a triple $M = (C, \tau, S)$ where:

- C is a finite set of constants,
- $\tau : C \to Ob(\mathbf{C})$ is a map that assigns an object in $\mathbf{C}$ to every constant,
- S is a distinguished goal type in $Ob(\mathbf{C})$.

Definition 8.2. Let $G = (\Sigma, \delta, S)$ be a (categorical) Lambek-Grishin Grammar over $C(LG^{IV}_T)$ and let $M = (C, \tau, S')$ be a semantic signature over a linearly distributive category $\mathbf{C}$. A *semantic interpretation* is a linearly distributive functor $F : C(LG^{IV}_T) \to \mathbf{C}$ together with a map $I : \Sigma \to C$ such that

$$F(S) = S'$$
$$F(\delta(w)) = \tau(I(w)) \text{ for all } w \in \Sigma$$

Definition 8.3. An extended categorical compositional distributional model of meaning is a categorical Lambek-Grishin grammar G over $C(LG^{IV}_T)$ together with a semantic signature M over $\mathbf{C}$ and a semantic interpretation $\langle F, I \rangle$.

It should be clear that here we opt for a direct translation of Lambek-Grishin derivations into proofs in a linear distributive category. Again, we demand our semantic signatures to be defined over categories that have sets as objects, so that one can actually compute a meaning:

Definition 8.4. Given $G = (\Sigma, \delta, S)$ a Lambek-Grishin Grammar over $C(LG^{IV}_T)$, $M = (C, \tau, S')$ a semantic signature over $\mathbf{C}$ such that the elements of C are actual inhabitants of the objects of $\mathbf{C}$, and a semantic interpretation $\langle F, I \rangle$, the meaning of a string $w = w_1...w_n \in \Sigma^+$ is defined iff w is a sentence of G. If it is defined, the meaning of w is $F(p)(I(w_1) \otimes ... \otimes I(w_n))$ where p is the grammatical proof of $\delta(w_1) \otimes ... \otimes \delta(w_n) \to S$.

The definition of an extended compositional distributional model of meaning is then analogous to the case of the Lambek systems:

Definition 8.5. An extended categorical compositional distributional model of meaning is a triple $(G, M, \langle F, I\rangle)$ where G is a Lambek-Grishin grammar, M is a semantic signature and $\langle F, I\rangle$ is a semantic interpretation over G and M.

8.3 A Collapsed Semantics

One might now wonder what the effect is of using the Lambek-Grishin Calculus as a syntactic backbone when we use the same vector spaces as a semantics; the collapse of tensor and co-tensor and their four implications make that the structural morphisms of the semantics are identical to those in the base models. Why should we get better results?

Well, one should consider that when using the models, the syntactic derivations are used in order to compute meaning. Since the extended models block global associativity but do allow for linear distributivities, the way meaning is computed is, although still by simple application, much more fine grained. That is, even though global commutativity and associativity are present in finite dimensional vector spaces, they are only used when the syntax specifies it.

Conclusion & Future Directions

In this final chapter, we will summarize our findings and elaborate on possible directions for future research.

Contributions

This thesis is an attempt at providing a framework for compositional distributional semantics based on the different variants of the Lambek Calculus (part I) and the extension of the non-associative Lambek Calculus **NL** to the Lambek-Grishin Calculus with type IV interactions (part II). Each part has a similar structure. Each first chapter has introduced the relevant categorical concepts one needs in order to situate the relevant type logics with respect to the Curry-Howard-Lambek correspondence. Each second chapter then develops the method of reasoning about morphism structure and equality in terms of *string diagrams* which are based on proof nets but deviate from the latter in that they require another way of defining equality between nets. Each third and fourth chapter then closes the parts by introducing the type logics and their categorical characterization and showing how each logic can interpreted in the category of finite-dimensional vector spaces, thus forming the basis of defining categorical compositional distributional models of meaning.

Future Research

There are obvious limitations to this study: first of all, the proposed framework ought to be *empirically evaluated* to see whether the adoption of Lambek-Grishin grammars indeed gives a more accurate semantics. Secondly, the Lambek-Grishin Calculus is not the only type logic around; one could choose from a wide range of extensions of the Lambek Calculus and see whether these give rise to structured categories interpretable in vector space semantics.

Evaluation

For the purpose of practically applying compositional distributional models of meaning, these models should be empirically evaluated against a large corpus. As we already noted in chapter 4, this also raises the issue of integrating *grammar induction* in the process of empirical evaluation of the models. Taking a truly empirical stand, evaluation should consist of both inducing grammars and extracting co-occurrence vectors from a realistic corpus. Although grammar extraction for Dutch and French has been done for certain type logical grammars Moot (2010a,b), the task of grammar

induction has not been performed for Lambek-Grishin grammars. However, one could take an existing, fixed grammar and use that to perform evaluation tasks such as sense disambiguation (in a similar fashion to what is done by Coecke et al. (2013)) or sentence similarity judgements.

Different Type-Logics

As we saw in the Introduction, the Lambek-Grishin Calculus is but one of the type logics designed to overcome the limitations of the Lambek Calculus. We have chosen to investigate the categorical aspects of this calculus as it was expected to have an intuitive interpretation in terms of symmetry and linear distributivities. Some notable other candidates for discussion are Combinatory Categorial Grammar, Multimodal Categorial Grammar, and the Displacement Calculus.

Multimodal Categorial Grammar enriches the base Lambek system with unary modalities that allow *structural control*: type-assignments with these modalities license or block the applicability of certain structural rules in the derivation process. The modalities of Multimodal Categorial Grammar come in residuated pairs, which then give rise to adjoint functors in the corresponding categorical languages. It would certainly be an interesting line of research to see how these multimodal systems can be interpreted in finite-dimensional vector spaces.

The Displacement Calculus extends the Lambek Calculus with wrapping operations for handling discontinuous dependencies. Whereas the Lambek calculus can be seen as the logic of strings composed by concatenation, Displacement Calculus adds facilities for dealing with *split strings*: expressions consisting of detached parts. On the type level, in addition to the Lambek operations, one has an extra residuated triple for the wrapping operations and a special constant, the separator, serving as a placeholder to mark the target sites for the intercalation of expressions. Further research would be needed to properly understand the categorical properties of this system, but an idea would be to introduce *displacement categories*, where a special object acts as a placeholder for other objects.

Bibliography

Steve Awodey. *Category theory*, volume 49 of *Oxford Logic Guides*. Oxford University Press, 2006.

J. Baez and M. Stay. Physics, topology, logic and computation: A Rosetta stone. In Bob Coecke, editor, *New Structures for Physics*, volume 813 of *Lecture Notes in Physics*, pages 95–172. Springer Berlin Heidelberg, 2011.

Arno Bastenhof. Tableaux for the Lambek-Grishin calculus. *CoRR*, abs/1009.3238, 2010. URL `http://arxiv.org/abs/1009.3238`.

Arno Bastenhof. *Categorial Symmetry*. PhD thesis, Utrecht University, 2013. URL `http://dspace.library.uu.nl/bitstream/1874/273870/1/bastenhof.pdf`.

RF Blute, JRB Cockett, RAG Seely, and TH Trimble. Natural deduction and coherence for weakly distributive categories. *Journal of Pure and Applied Algebra*, 113(3):229–296, 1996.

Richard Blute and Philip Scott. Category theory for linear logicians. *Linear Logic in Computer Science*, 316:3–65, 2004.

Roberto Bonato and Christian Retoré. Learning rigid Lambek grammars and minimalist grammars from structured sentences. In Luboš Popelínský, editor, *Third workshop on Learning Language in Logic, Strasbourg*, pages 23–34, 2001.

Maria Bulińska. On the complexity of nonassociative Lambek calculus with unit. *Studia Logica*, 93 (1):1–14, 2009.

Wojciech Buszkowski. Completeness results for Lambek syntactic calculus. *Mathematical Logic Quarterly*, 32(1-5):13–28, 1986.

J Robin B Cockett and Robert AG Seely. Proof theory for full intuitionistic linear logic, bilinear logic, and mix categories. *Theory and Applications of categories*, 3(5):85–131, 1997a.

JRB Cockett and RAG Seely. Weakly distributive categories. *Journal of Pure and Applied Algebra*, 114(2):133–173, 1997b.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. Mathematical foundations for a compositional distributional model of meaning. *arXiv preprint arXiv:1003.4394*, 2010.

Bob Coecke, Edward Grefenstette, and Mehrnoosh Sadrzadeh. Lambek vs. Lambek: Functorial vector space semantics and string diagrams for Lambek calculus. *Annals of Pure and Applied Logic*, 164(11):1079–1100, 2013.

Antonin Delpeuch. Complexity of grammar induction for quantum types. *arXiv preprint arXiv:1404.3925*, 2014.

Kosta Došen. Sequent-systems and groupoid models. I. *Studia Logica*, 47(4):353–385, 1988.

J. R. Firth. A synopsis of linguistic theory 1930-55. 1952-59:1–32, 1957.

Nikolaos Galatos, Peter Jipsen, Tomasz Kowalski, and Hiroakira Ono. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics: An Algebraic Glimpse at Substructural Logics*, volume 151. Elsevier, 2007.

Jean-Yves Girard. Linear logic. *Theoretical computer science*, 50(1):1–101, 1987.

V.N. Grishin. On a generalization of the Ajdukiewicz-Lambek system. In *Studies in nonclassical logics and formal systems*, pages 315–334. Moscow, 1983. Nauka (English translation in Abrusci & Casadio (Eds.), New perspectives in logic and formal linguistics. Bulzoni, Rome, 2002).

Herman Hendriks. Compositionality and model-theoretic interpretation. *Journal of Logic, Language and Information*, 10(1):29–48, 2001.

Riny Huybregts. The weak inadequacy of context-free phrase structure grammars. In G.J. de Haan, M. Trommelen, and W. Zonneveld, editors, *Van periferie naar kern*, pages 81–99. 1984.

Aravind K Joshi, K Vijay Shanker, and David Weir. The convergence of mildly context-sensitive grammar formalisms. 1990.

Aravind Krishna Joshi. *Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions?* University of Pennsylvania, Moore School of Electrical Engineering, Department of Computer and Information Science, 1985.

André Joyal and Ross Street. The geometry of tensor calculus, I. *Advances in Mathematics*, 88(1): 55–112, 1991a.

André Joyal and Ross Street. The geometry of tensor calculus II. 585, 1991b. URL `http://www.math.mq.edu.au/~street/GTCII.pdf`.

Makoto Kanazawa. Identification in the limit of categorial grammars. *Journal of Logic, Language and Information*, 5(2):115–155, 1996.

Maciej Kandulski. The equivalence of nonassociative Lambek categorial grammars and context-free grammars. *Mathematical Logic Quarterly*, 34(1):41–52, 1988.

J. Lambek. Categorial and categorical grammars. In Richard T. Oehrle, Emmon Bach, and Deirdre Wheeler, editors, *Categorial Grammars and Natural Language Structures*, volume 32 of *Studies in Linguistics and Philosophy*, pages 297–317. Springer Netherlands, 1988. URL `http://dx.doi.org/10.1007/978-94-015-6878-4_11`.

Joachim Lambek. The mathematics of sentence structure. *American mathematical monthly*, 65: 154–170, 1958.

Joachim Lambek. On the calculus of syntactic types. In Roman Jakobson, editor, *Structure of language and its mathematical aspects*, volume 12, pages 166–178. Providence, RI: American Mathematical Society, 1961.

Joachim Lambek. Deductive systems and categories. *Theory of Computing Systems*, 2(4):287–318, 1968.

Joachim Lambek and Philip J Scott. *Introduction to higher-order categorical logic*, volume 7 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1988.

Matthijs Melissen. The generative capacity of the Lambek–Grishin calculus: A new lower bound. In Philippe de Groote, Markus Egg, and Laura Kallmeyer, editors, *Proceedings of the 14th Conference on Formal Grammar 2009 (FG 2009)*, volume 5591 of *Lecture Notes in Computer Science*, pages 118–132, Bordeaux, France, 2011. Springer. URL `http://www.matthijsmelissen.nl/site/publications/fg09.pdf`.

Richard Montague. *English as a formal language*. Ed. di Comunità, 1970a.

Richard Montague. Universal grammar. *Theoria*, 36(3):373–398, 1970b.

Michael Moortgat. Multimodal linguistic inference. *Journal of Logic, Language and Information*, 5(3-4):349–385, 1996.

Michael Moortgat. Symmetries in natural language syntax and semantics: the Lambek-Grishin calculus. In *Logic, Language, Information and Computation*, pages 264–284. Springer, 2007.

Michael Moortgat. Symmetric categorial grammar. *Journal of Philosophical Logic*, 38(6):681–710, 2009.

Michael Moortgat. Categorial type logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language (Second Edition)*, pages 95–179. Elsevier, London, 2011.

Michael Moortgat and Richard Moot. Proof nets and the categorial flow of information. In Alexandru Baltag, Davide Grossi, Alexandru Marcoci, Ben Rodenhäuser, and Sonja Smets, editors, *Logic and Interactive RAtionality. Yearbook 2011*, pages 270–302. 2012. URL `http://www.illc.uva.nl/dg/?page_id=78`.

Richard Moot. *Proof nets for linguistic analysis*. PhD thesis, Utrecht University, 2002. URL `http://dspace.library.uu.nl/handle/1874/613`.

Richard Moot. Proof nets for display logic. *arXiv preprint arXiv:0711.2444*, 2007.

Richard Moot. Extraction of type-logical supertags from the Spoken Dutch Corpus. *Supertagging: Using Complex Lexical Descriptions in Natural Language Processing*, 2010a.

Richard Moot. Semi-automated extraction of a wide-coverage type-logical grammar for french. In *TALN 2010*, 2010b.

Richard Moot and Quintijn Puite. Proof nets for the multimodal Lambek calculus. *Studia Logica*, 71(3):415–442, 2002.

Glyn Morrill, Oriol Valentín, and Mario Fadda. The displacement calculus. *Journal of Logic, Language and Information*, 20(1):1–48, 2011.

Mati Pentus. Lambek grammars are context free. In *Logic in Computer Science, 1993. LICS'93., Proceedings of Eighth Annual IEEE Symposium on*, pages 429–433. IEEE, 1993.

Dirk Roorda. *Resource Logics: proof-theoretical investigations*. PhD thesis, University of Amsterdam, 1991.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229, 1991.

Peter Selinger. A survey of graphical languages for monoidal categories. In Bob Coecke, editor, *New structures for physics*, pages 289–355. Springer, 2011.

Stuart M Shieber. Evidence against the context-freeness of natural language. In Walter J Savitch, Emmon Bach, William Marsh, and Gila Safran-Naveh, editors, *The Formal complexity of natural language*, pages 320–334. Springer, 1987.

Morten Heine Sørensen and Paweł Urzyczyin. *Lectures on the Curry-Howard Isomorphism*, volume 149 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 2006.

Edward P Stabler. Computational perspectives on minimalism. In Cedric Boeckx, editor, *Oxford Handbook of Linguistic Minimalism*, pages 617–643. 2011.

Mark Steedman. *The syntactic process*. MIT Press, 2000.

Lutz Straßburger and François Lamarche. On proof nets for multiplicative linear logic with units. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic*, volume 3210 of *Lecture Notes in Computer Science*, pages 145–159. Springer Berlin Heidelberg, 2004. URL `http://dx.doi.org/10.1007/978-3-540-30124-0_14`.

Heinrich Wansing. Formulas-as-types for a hierarchy of sublogics of intuitionistic propositional logic. In *Nonclassical Logics and Information Processing*, pages 125–145. Springer, 1992.