Logic and linear algebra: an introduction

Daniel Murfet

July 30, 2015

Abstract

We give an introduction to logic tailored for algebraists, by explaining how to represent proofs in linear logic as linear maps between vector spaces. The interesting part of this vector space semantics is based on the cofree cocommutative coalgebra of Sweedler [74] and the recent explicit computations of liftings in [64].

A contrario, for intuitionists, *Modus Ponens* is not a legal advisor, it is the door open on a new world, it is the application of a function ($A \Rightarrow B$) to an argument (A) yielding a result (B). Proofs are no longer those sequences of symbols created by a crazy bureaucrat, they are functions, *morphisms*.

Jean-Yves Girard, *The Blind Spot*

1 Introduction

Logic is familiar to mathematicians in other fields primarily as the study of the *truth* of mathematical propositions, but there is an alternative tradition which views logic as being about the structure of the *collection of proofs*. The contrast between these points of view can be explained by an analogy (not perfect, but perhaps helpful) between propositions in logic and differential equations. In this analogy the truth of a proposition corresponds to the *existence* of a solution to a differential equation, and the set of proofs of a proposition plays the role of the space of solutions. There is often a great deal of redundancy in the space of solutions, in the sense that two solutions which are apparently different may be related by a symmetry of the underlying manifold, and thus are in essence “the same”. Moreover there are examples, such as the Seiberg-Witten equations, where the moduli space obtained by identifying solutions related by symmetries is a compact manifold, whose “finite” geometry is the key to understanding the content of the original equations.

There was an analogous phenomenon at the founding of formal symbolic logic, when a central problem was to establish the consistency of arithmetic. While working on this problem Gentzen discovered a new style of presenting proofs, called the *sequent calculus*, in which the set of proofs of any arithmetic proposition is revealed to contain a great deal of redundancy: many apparently different proofs actually have "the same" logical content. This redundancy has its origin in applications of *Modus Ponens*, which is known in sequent calculus as the *cut rule*. This rule creates indirection and implicitness in proofs, by making theorems depend on lemmas (whose proofs are somewhere else, hence the indirection). However this implicitness can be "explicated" without essentially changing the content of the proof; this deep result is known as Gentzen's *Hauptsatz*. Identifying proofs related by this explicitation (called *cut-elimination*) yields the much more tractable set of *cut-free proofs*, the analogue of the compact manifold of solutions modulo symmetry. By reducing consistency to a problem about cut-free proofs, where it is trivial, Gentzen was able to prove the consistency of arithmetic. For the full details of this story see [**?**, **?**].

The purpose of this article is to introduce the reader to this alternative tradition of logic, with its emphasis on the structure and symmetries of collections of proofs. We will do this using linear logic and its semantics in vector spaces and linear maps. Here the word "semantics" has a meaning close to what we mean by *representation* in algebra: the structure of linear logic is encoded in its connectives, deduction rules, and cut-elimination transformations, and some insight into this structure can be gained by mapping it in a structure-preserving way to linear operators on vector spaces.

The outline of the article is as follows: in Section **??** we introduce the λ-calculus, intuitionistic logic and linear logic, with an emphasis on the role of duplication. In Section 5 we assign to every proof in intuitionistic linear logic a string diagram and corresponding linear map, and give a detailed example. In Section 6 we turn to cut-elimination, examining a mildly non-trivial example in detail. In Appendix B we examine tangent maps in connection with proofs in linear logic. Most of what we have to say is well-known, with the exception of some aspects of the vector space semantics in Section 5.3. There are many aspects of logic and its connections with other subjects that we cannot cover: for a well-written account of analogies between logic, topology and physics we recommend the survey of Baez and Stay [7].

Acknowledgements. Thanks to Nils Carqueville, Jesse Burke, and Andante.

2 A sketch of linear logic as a language

Linear logic was introduced by Girard in the 1980s and has been the subject of active study ever since. Its special features make it naturally connected with algebra (broadly interpreted). At root this connection is linguistic: symmetric closed monoidal categories are ubiquitous in algebra, and their formal language is a subset of linear logic. For example, writing $(-) \multimap (-)$ for the internal Hom in a symmetric closed monoidal category $\mathcal{C}$, there is for any triple of objects $a, b, c \in \mathcal{C}$ a canonical map

$$(a \multimap b) \otimes (b \multimap c) \longrightarrow a \multimap c \tag{2.1}$$

which is the internal notion of *composition*. It is derived from the structure of the category $\mathcal{C}$ in the following way: from the evaluation maps

$$e_{a,b} : a \otimes (a \multimap b) \longrightarrow b, \qquad e_{b,c} : b \otimes (b \multimap c) \longrightarrow c$$

and the adjunction between internal Hom and tensor we obtain a map

$$\begin{tikzcd}
\mathrm{Hom}_{\mathcal{C}}(c, c) \arrow[d, "\mathrm{Hom}(e_{b,c},1)"] \\
\mathrm{Hom}_{\mathcal{C}}(b \otimes (b \multimap c), c) \arrow[d, "\mathrm{Hom}(e_{a,b}\otimes 1,1)"] \\
\mathrm{Hom}_{\mathcal{C}}(a \otimes (a \multimap b) \otimes (b \multimap c), c) \arrow[d, "\cong \quad \text{adjunction}"] \\
\mathrm{Hom}_{\mathcal{C}}((a \multimap b) \otimes (b \multimap c), a \multimap c)\,.
\end{tikzcd} \tag{2.2}$$

The image of the identity on c under this map is the desired composition.

This construction is formal, in the sense that it does not depend on the nature of the particular objects a, b, c. The formality can be made precise by presenting the same construction using the internal language provided by linear logic:

$$\dfrac{\dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{\dfrac{}{B \vdash B} \qquad \dfrac{}{C \vdash C}}{B, B \multimap C \vdash C}\multimap L}{A, A \multimap B, B \multimap C \vdash C}\multimap L}{A \multimap B, B \multimap C \vdash A \multimap C}\multimap R \tag{2.3}$$

This syntactical object is called a *proof*. Here A, B, C are formal variables that we can think of as standing for unknown objects of $\mathcal{C}$ (or in fact any symmetric closed monoidal category) and $\multimap$ is a connective called linear implication. If we choose to specialise these variables to particular objects a, b, c the proof acquires a "shadow" or interpretation in $\mathcal{C}$, namely, the internal composition (2.1).

Even without formally defining what a proof is (this will be done in Section 4) one can start to see how this proof formalises the construction in (2.2). For example the deduction rule $\multimap L$ corresponds to precomposition with an evaluation map, and $\multimap R$ to the use of adjunction. This demonstrates how linear logic formalises the construction of canonical maps in a symmetric closed monoidal category. Moreover, logic also dictates the *relations* between canonical maps constructed in this way, which is the reason why logic often arises in connection with coherence questions [?].

Happily, symmetric closed monoidal categories are the *least* interesting part of linear logic. Things become more interesting when we start to talk about natural operations on morphisms in $\mathcal{C}$ that cannot be encoded by a diagram like (2.2) built up inductively from adjunction and pre- and post-composition with maps already constructed. For example, the operation which takes as input an object a, an endomorphism $f : a \longrightarrow a$, and returns the square $f \circ f$ has no such description (why?). However, this operation *is* described by a proof in linear logic, namely:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{A \vdash A}}{A, A \multimap A \vdash A}\multimap L}{A, A \multimap A, A \multimap A \vdash A}\multimap L}{A \multimap A, A \multimap A \vdash A \multimap A}\multimap R}{!(A \multimap A), A \multimap A \vdash A \multimap A}\text{der}}{!(A \multimap A), !(A \multimap A) \vdash A \multimap A}\text{der}}{!(A \multimap A) \vdash A \multimap A}\text{ctr} \tag{2.4}$$

This proof involves a new connective ! called the *exponential*, but we recognise that until the fourth line, this is the earlier proof with A substituted for B, C. Since that proof was presenting the operation of composition, it only takes a small leap of imagination to see the proof (2.4) as a gadget which takes a morphism $f : a \longrightarrow a$ and duplicates it, feeding f, f into the composition operation to obtain $f \circ f$. The aim of this note is that, by the end, the reader will be able to look at (2.4) and see all of this immediately.

The range of operations that can be defined using ! is significantly larger. For example any integer can be encoded as proof (in fact (2.4) is an encoding of the number 2) and addition and multiplication of integers can be also encoded. However for these operations to have an internal meaning in the same way that composition does, the category $\mathcal{C}$ needs to be equipped with additional structure. This turns out to be a comonad $! : \mathcal{C} \longrightarrow \mathcal{C}$ which interprets the exponential in the same way that the internal Hom $\multimap : \mathcal{C} \times \mathcal{C} \longrightarrow \mathcal{C}$ interprets linear implication. The category of (possibly infinite dimensional) vector spaces and linear maps is one such category, when it is equipped with the endofunctor ! sending a vector space to the cofree cocommutative coalgebra generated by it.

TODO Cite Section 2.4.2 of Benton for abelian groups

3 Computer programs and the λ-calculus

Around the same time as Turing defined his machines, Church gave a different formalisation of the idea of a computable function called the λ-calculus [25, 71]. Both concepts identify the same class of computable functions $\mathbb{N} \longrightarrow \mathbb{N}$, but the λ-calculus is more natural from the point of category theory, and it serves as the theoretical underpinning of functional programming languages like Lisp [58] and Haskell. Intuitively, while Turing machines make precise the concept of *logical state* and *state transition*, the λ-calculus captures the concepts of *variables* and *substitution*.

The λ-calculus is determined by its terms and a rewrite rule on those terms. The terms are to be thought of as programs, and the rewrite rule as the method of execution of programs. A *term* in the λ-calculus is either one of a countable number of variables $x, y, z, \ldots$ or an expression of the type

$$(M\ N) \quad \text{or} \quad (\lambda x\,.\,M) \tag{3.1}$$

where M, N are terms and x is any variable. The terms of the first type are called *function applications* while those of the second type are called *lambda abstractions*. An example of a term, or program, that will be important throughout this note is

$$T := (\lambda y\,.\,(\lambda x\,.\,(y\,(y\ x))))\,. \tag{3.2}$$

Note that the particular variables chosen are not important, but the pattern of occurrences of the *same* variable certainly is. That is to say, we deal throughout with terms up to an equivalence relation called α-conversion, under which for example T is equivalent to the term $(\lambda z\,.\,(\lambda t\,.\,(z\,(z\ t))))$. Following standard practice, we will adopt this convention.

If we are supposed to think of T as a program, we must describe what this program *does*. The dynamic content of the λ-calculus arises from a rewrite rule called *β-reduction* generated by the following basic rewrite rule

$$((\lambda x\,.\,M)\,N) \longrightarrow_\beta M[N/x] \tag{3.3}$$

where M, N are terms, x is a variable, and $M[N/x]$ denotes the term M with all free occurrences of x replaced by N.[1] We write $A \to_\beta B$ if the term B is obtained from A by rewriting a sub-term of A according to the rule (3.3). The smallest reflexive, transitive and symmetric relation containing $\to_\beta$ is called *β-equivalence*, written $M =_\beta N$ [71, §2.5].

We think of the lambda abstraction $(\lambda x\,.\,M)$ as a program with input x and body M, so that the β-reduction step in (3.3) has the interpretation of our program being fed the input term N which is subsequently bound to x throughout M. A term is *normal* if there are no sub-terms of the type on the left hand side of (3.3). In the λ-calculus computation occurs when two terms are coupled by a function application in such a way as to create a term which is not in normal form: then β-reductions are performed until a normal form (the output of the computation) is reached.[2]

In terms of the rewriting of terms generated by β-reduction, let us now examine what the program T does when fed another term. For a term M, we have

$$(T\ M) = ((\lambda y\,.\,(\lambda x\,.\,(y\,(y\ x))))\,M) \longrightarrow_\beta (\lambda x\,.\,(M\,(M\ x)))\,. \tag{3.4}$$

[1] There is a slight subtlety here since we may have to rename variables in order to avoid free variables in N being "captured" as a result of this substitution, see [71, §2.3].

[2] Not every λ-term may be reduced to a normal form by β-reduction because this process does not necessarily terminate. However if it does terminate then the resulting reduced term is canonically associated to the original term; this is the content of Church-Rosser theorem [71, §4.2].

Thus $(T\,M)$ behaves like the square of M, in the sense that it is a program which takes a single input x and returns $(M\,(M\;x))$. If we modify T by nesting more than two function applications, we can define for each integer $n \geq 0$ a term $\underline{n}$ called the nth *Church numeral* [71, §3.2]. This program has the property that $(\underline{n}\,M)$ behaves like the nth power of M. The Church numberal $\underline{2}$, which is our old friend T, will be our object of study throughout this note.[3]

We have now defined the λ-calculus and introduced our basic example of a program, the Church numeral $\underline{2}$. The emphasis in this article is on the structure of sets of proofs, and this is turns out to be closely related to the structure of sets of *programs*, that is, λ-terms. Let D denote the set of all β-equivalence classes of λ-terms. This is a fabulously complicated set: consider that for example T defines a function

$$\begin{aligned} T : D &\longrightarrow D, \\ [M] &\mapsto [(T\,M)]\,. \end{aligned}$$

and the same is true of any term. The discovery of the first mathematical model of this structure by Dana Scott [68] was an important milestone in the development of computer science, and it marked the beginning of the field called *denotational semantics*. For our purposes, however, the full λ-calculus is *too* complicated. There is a much weaker version called the simply-typed λ-calculus which is more closely related to linear logic.

The reader familiar with programming will know that functions in many programming languages are *typed* by their input and output specification. For instance, a function computing the nth Fibonacci number in the language C takes an input of type "int" – the integer n – and returns an output also of type "int". In the *simply-typed* λ-calculus the Church numeral $\underline{2}$ is the same as before, but with type annotations

$$T_{typed} := (\lambda y^{A\to A}\,.\,(\lambda x^{A}\,.\,(y\,(y\;x))))\,.$$

These annotations indicate that the first input y is restricted to be of the type of a function from A to A (that is, of type $A \to A$) while x is restricted to be of type A. Overall T_{typed} takes a term of type $A \to A$ and returns another term of the same type, so it is a term of type $(A \to A) \to (A \to A)$. For a more complete discussion see [71, §6].

[3]To put this into a broader context: once the integers have been translated into terms in the λ-calculus it makes sense to say that a program F *computes* a function $f : \mathbb{N} \longrightarrow \mathbb{N}$ if for each $n \geq 0$ we have $(F\;\underline{n}) =_\beta \underline{f(n)}$. A function f is called *computable* if there is such a term F.

4 Linear logic

Linear Logic is based on the idea of resources, an idea violently negated by the contraction rule. The contraction rule states precisely that a resource is potentially infinite, which is often a sensible hypothesis, but not always. The symbol ! can be used precisely to distinguish those resources for which there are no limitations. From a computational point of view $!A$ means that the datum A is stored in the memory and may be referenced an unlimited number of times. In some sense, $!A$ means forever.

Jean-Yves Girard, Andre Scedrov, Philip J. Scott, *Bounded linear logic*

There are two main styles of formal mathematical proofs: Hilbert style systems, which most mathematicians will be exposed to as undergraduates, and natural deduction or sequent calculus systems, which are taught to students of computer science. What these two styles share is that they are about propositions (or sequents) and their proofs, which are constructed from axioms via deduction rules. The differences lie in the way that proofs are formatted and manipulated as objects on the page. In this note we will consider only the sequent calculus style of logic, since it is more naturally connected to category theory. The proofs (2.3) and (2.4) from the introduction are examples of such proofs.

In *linear logic*[4] there are propositional variables $x, y, z, \ldots$, two binary connectives $\multimap$ (linear implication), $\otimes$ (tensor) and a single unary connective ! (the exponential). There is a single constant 1. The set of *formulas* is defined recursively as follows: any propositional variable or constant is a formula, and if A, B are formulas then so are

$$A \multimap B, \qquad A \otimes B, \qquad !A\,.$$

An important example of a formula is $\mathbf{int}_A$ defined for any formula A as

$$\mathbf{int}_A = \,!(A \multimap A) \multimap (A \multimap A)\,. \tag{4.1}$$

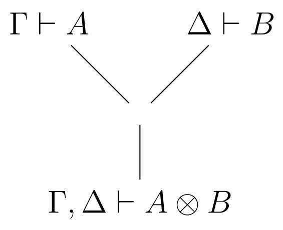
For reasons that will become clear, $\mathbf{int}_A$ is referred to the type of *integers on* A (throughout *type* is used as a synonym for formula). A *sequent* is an expression of the form

$$A_1, \ldots, A_n \vdash B$$

with formulas $A_1, \ldots, A_n, B$ of the logic connected by a *turnstile* $\vdash$. The intuitive reading of this sequent is the proposition that B may be deduced from the hypotheses $A_1, \ldots, A_n$. The letters Γ, Δ are used to stand for arbitrary sequences of formulas, possibly empty. A *proof* of a sequent is a series of deductions, beginning from tautologous *axioms* of the form $A \vdash A$, which terminates with the given sequent. At each step of the proof the deduction must follow a list of *deduction rules*.

[4]We consider propositional intuitionistic linear logic without additives, with the exception of Section **??** where we add quantifiers.

More precisely, let us define a *pre-proof* to be a rooted tree whose edges are labelled with sequents. In order to follow the logical ordering, the tree is presented with its root vertex at the bottom of the page, and we orient edges towards the root (so downwards). The labels on incoming edges at a vertex are called *hypotheses* and on the outgoing edge the *conclusion*. For example consider the following tree, and its equivalent presentation in sequent calculus notation:



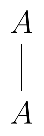
$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \tag{4.2}$$

Leaves are presented in the sequent calculus notation with an empty numerator.

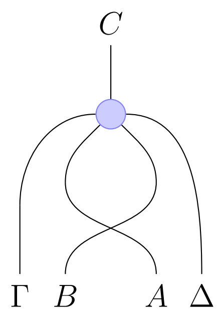
Definition 4.1. A *proof* is a pre-proof together with a compatible labelling of vertices by deduction rules. The list of deduction rules is given in the first column of (4.3) – (4.15). A labelling is *compatible* if at each vertex, the sequents labelling the incident edges match the format displayed in the deduction rule.

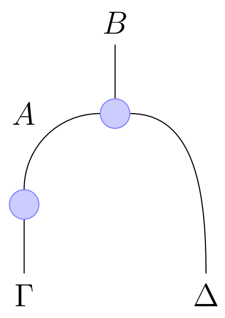
In all deduction rules, the sets Γ and Δ may be empty and, in particular, the promotion rule may be used with an empty premise. In the promotion rule, $!\Gamma$ stands for a list of formulas each of which is preceeded by an exponential modality, for example $!A_1, \ldots, !A_n$. The diagrams on the right are string diagrams and should be ignored until Section 5. In particular they are *not* the trees associated to proofs.

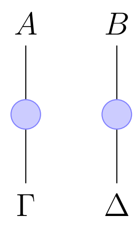
$$\text{(Axiom):} \quad \frac{}{A \vdash A} \tag{4.3}$$

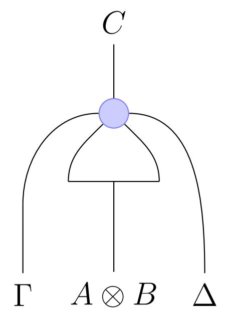


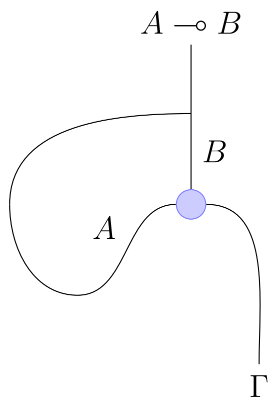
$$\text{(Exchange):} \quad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C} \tag{4.4}$$



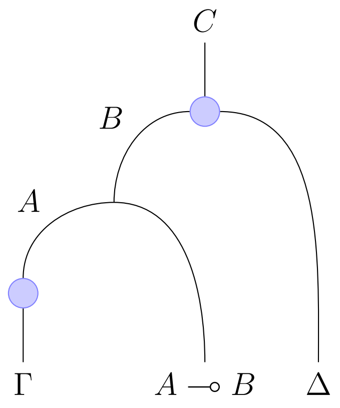
(Cut): $\dfrac{\Gamma \vdash A \qquad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}\,\text{cut}$  (4.5)

(Right $\otimes$): $\dfrac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\,\otimes\text{-}R$  (4.6)

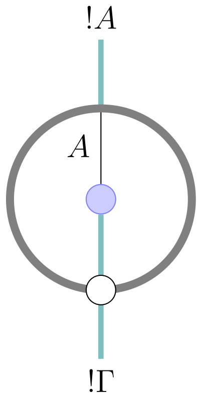
(Left $\otimes$): $\dfrac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\,\otimes\text{-}L$  (4.7)

(Right $\multimap$): $\dfrac{A, \Gamma \vdash B}{\Gamma \vdash A \multimap B}\,\multimap R$  (4.8)

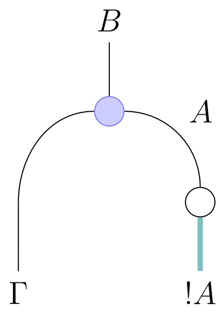
(Left $\multimap$): $\dfrac{\Gamma \vdash A \qquad B, \Delta \vdash C}{\Gamma, A \multimap B, \Delta \vdash C} \multimap L$ (4.9)



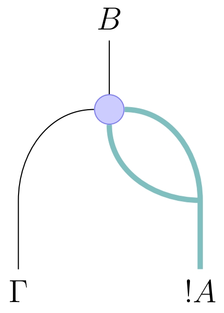
(Promotion): $\dfrac{!\Gamma \vdash A}{!\Gamma \vdash !A}$ prom (4.10)



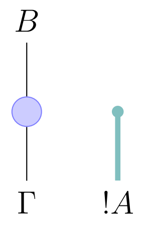
(Dereliction): $\dfrac{\Gamma, A \vdash B}{\Gamma, !A \vdash B}$ der (4.11)



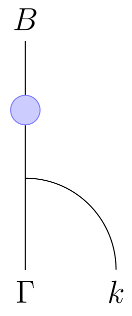
(Contraction): $\dfrac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B}$ ctr (4.12)



$$\text{(Weakening):}\quad \dfrac{\Gamma \vdash B}{\Gamma, !A \vdash B}\,\text{weak} \tag{4.13}$$



$$\text{(Left 1):}\quad \dfrac{\Gamma \vdash A}{\Gamma, 1 \vdash A}\,1\text{-}L \tag{4.14}$$



$$\text{(Right 1):}\quad \dfrac{}{\vdash 1}\,1\text{-}R \tag{4.15}$$

What about permutations? I mean the deduction rules need things in certain places.

Example 4.2. For any formula A let $\underline{2}_A$ denote the proof (2.4) from Section 2, which we repeat here for the reader's convenience:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{A \vdash A}}{A, A \multimap A \vdash A}\multimap L}{A, A \multimap A, A \multimap A \vdash A}\multimap L}{A \multimap A, A \multimap A \vdash A \multimap A}\multimap R}{!(A \multimap A), A \multimap A \vdash A \multimap A}\,\text{der}}{!(A \multimap A), !(A \multimap A) \vdash A \multimap}\,\text{der}}{\dfrac{!(A \multimap A) \vdash A \multimap A}{\vdash \mathbf{int}_A}\multimap R}\,\text{ctr} \tag{4.16}$$

For convenience in what follows we will also write $\underline{2}_A$ for the proof of $!(A \multimap A) \vdash A \multimap A$ obtained by reading the above proof up to the penultimate line. For each integer $n \geq 0$ there is a proof $\underline{n}_A$ of $\mathbf{int}_A$ constructed along similar lines, see [37, §5.3.2] and [26, §3.1].

In what sense is this proof an avatar of the numeral 2? The same question made sense in the context of the λ-calculus. There, we only appreciated the relationship between the term T in (**??**) and the numeral 2 when we saw how T *interacted* with other terms M by forming the function application $(T\,M)$ and then finding a normal form with respect to β-equivalence. The analogue of function application in linear logic is the cut rule. The analogue of β-equivalence is an equivalence relation on the set of proofs of any sequent, which we write as $\rho =_{cut} \delta$ and communicate by saying that ρ and δ are *equivalent under cut-elimination*. The details of this relation will be explained later, but for now let us examine how it gives a "dynamic" meaning to the proof $\underline{2}_A$.

Example 4.3. Let π be any proof of $A \vdash A$ and consider the proof

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\pi\\ \vdots\\ A \vdash A\end{matrix}}{\vdash A \multimap A}\,{\multimap R}}{\vdash !(A \multimap A)}\,\text{prom} \qquad \begin{matrix}\underline{2}_A\\ \vdots\\ !(A \multimap A) \vdash A \multimap A\end{matrix}}{\vdash A \multimap A}\,\text{cut} \tag{4.17}$$

We write $\delta \,|\, \rho$ for the cut rule applied with left branch ρ and right branch δ, and $\text{prom}(\pi)$ for the left hand branch of (4.17), so that the whole proof is denoted $\underline{2}_A \,|\, \text{prom}(\pi)$.

Intuitively, the promotion rule "makes perennial" the data of the proof π and prepares it to be used more than once (that is, in a *nonlinear* way). The cut rule is logically the incarnation of Modus Ponens, and from the point of view of categorical semantics is the linguistic antecedent of composition. In the context of (4.17) it plays the role of "feeding" the perennial version of π as input to $\underline{2}_A$. After our experience with the λ-term T it is not a surprise that the above proof is equivalent under cut-elimination to

$$\dfrac{\begin{matrix}\pi\\ \vdots\\ A \vdash A\end{matrix} \qquad \begin{matrix}\pi\\ \vdots\\ A \vdash A\end{matrix}}{A \vdash A}\,\text{cut} \tag{4.18}$$

which we write as $\pi \,|\, \pi$.

With π standing for the term M in Section **??**, we see the close analogy between the program T of λ-calculus and the proof $\underline{2}_A$ of linear logic, with the cut (4.17) playing the role of $(T\,M)$ and cut-elimination the role of β-reduction. This is one aspect of the *Curry-Howard correspondence* which relates programs in the simply-typed λ-calculus (and its extensions) to proofs in intuitionistic logic (and its extensions) with cut-elimination playing the role of β-reduction; see [**?**].

5 Semantics of linear logic

As we have already mentioned, on the set of proofs of any sequent $\Gamma \vdash A$ there is an equivalence relation called *cut-elimination*, see Section **??**.

One way to elaborate on the reason *why* (4.17) is equivalent to (4.18) would be to list the rewrite rules that make up the equivalence relation. However we take an alternative approach: in the next section we develop the semantics of linear logic and explain how to reframe the discussion of $\underline{2}_A$ in terms of an associated linear map of vector spaces.

Return to the structure of collections of proofs

Writing $[A \vdash A]$ for the set of equivalence classes of proofs of $A \vdash A$ notice how from the proof $\underline{2}_A$ we have constructed a *function*, namely

$$\begin{gathered}[A \vdash A] \longrightarrow [A \vdash A]\,, \\ [\pi] \mapsto [\underline{2}_A \,|\, \mathrm{prom}(\pi)] = [\pi \,|\, \pi]\,.\end{gathered}$$

A *categorical semantics* of linear logic [61, 18] assigns to each type A an object $[\![A]\!]$ of some category and to each proof of $\Gamma \vdash A$ a morphism $[\![\Gamma]\!] \longrightarrow [\![A]\!]$ in such a way that two proofs equivalent under cut-elimination are assigned the same morphism; these objects and morphisms are called *denotations*. The connectives of linear logic become structure on the category of denotations, and compatibility with cut-elimination imposes identities relating these structures to one another.

The upshot is that to define a categorical semantics the category of denotations must be a closed symmetric monoidal category equipped with a comonad, which is used to model the exponential modality [61, §7]. This is a refinement of the equivalence between simply-typed λ-calculus and cartesian closed categories due to Lambek and Scott [56]. The first semantics of linear logic were the coherence spaces of Girard [37, §3] which are a refined form of Scott's model of the λ-calculus. Models of full linear logic with negation involve the $\star$-autonomous categories of Barr [9, 10, 11] and the extension to include quantifiers involves indexed monoidal categories [70].

5.1 Denotations of formulas

In this paper denotations all take place in the category $\mathcal{V}$ of k-vector spaces. Throughout k is an algebraically closed field of characteristic zero. We explain the denotations of types now, and leave the denotation of proofs to the next section. To this end, for a vector space V let $!V$ denote the cofree cocommutative coalgebra generated by V. We will discuss the explicit form of this coalgebra in the next section; for the moment it is enough to know that it exists and is determined up to unique isomorphism.

Definition 5.1. The *denotation* $[\![A]\!]$ of a type A is defined inductively as follows:

- The propositional variables $x, y, z, \ldots$ are assigned chosen finite-dimensional vector spaces $[\![x]\!], [\![y]\!], [\![z]\!], \ldots$;
- $[\![1]\!] = k$;
- $[\![A \otimes B]\!] = [\![A]\!] \otimes [\![B]\!]$;
- $[\![A \multimap B]\!] = [\![A]\!] \multimap [\![B]\!]$ which is notation for $\mathrm{Hom}_k([\![A]\!], [\![B]\!])$;
- $[\![!A]\!] = ![\![A]\!]$.

The denotation of a group of formulas $\Gamma = A_1, \ldots, A_n$ is their tensor product

$$[\![\Gamma]\!] = [\![A_1]\!] \otimes \cdots \otimes [\![A_n]\!] \,.$$

If Γ is empty then $[\![\Gamma]\!] = k$.

Example 5.2. Let A be a type whose denotation is $V = [\![A]\!]$. Then from (4.1),

$$[\![\mathbf{int}_A]\!] = [\![!(A \multimap A) \multimap (A \multimap A)]\!] = \operatorname{Hom}_k(!\operatorname{End}_k(V), \operatorname{End}_k(V)) \,.$$

5.2 Denotations of proofs

In the previous section we associated to each type A a vector space $[\![A]\!]$. In this section we complete the construction of the vector space semantics of linear logic by assigning to each proof π of a sequent $\Gamma \vdash B$ a linear map $[\![\Gamma]\!] \longrightarrow [\![B]\!]$. The essential features of this assignment can be seen already in the following simple example:

Example 5.3. The denotation of the proof $\underline{2}_A$ of $\vdash \mathbf{int}_A$ will be a morphism

$$[\![\underline{2}_A]\!] : k \longrightarrow [\![\mathbf{int}_A]\!] = \operatorname{Hom}_k(!\operatorname{End}_k(V), \operatorname{End}_k(V)) \,, \qquad (5.1)$$

or equivalently, a linear map $!\operatorname{End}_k(V) \longrightarrow \operatorname{End}_k(V)$. What is this linear map? It turns out (see Example 5.8 below for details) that it is the composite

$$!\operatorname{End}_k(V) \xrightarrow{\Delta} !\operatorname{End}_k(V) \otimes !\operatorname{End}_k(V) \xrightarrow{d \otimes d} \operatorname{End}_k(V)^{\otimes 2} \xrightarrow{-\circ-} \operatorname{End}_k(V) \qquad (5.2)$$

where Δ is the coproduct, d is the universal map, and the last map is the composition. How to reconcile this linear map with the corresponding program in the λ-calculus, which has the meaning "square the input function"? As we will explain below, for $\alpha \in \operatorname{End}_k(V)$ there is a naturally associated element $|o\rangle_\alpha \in !\operatorname{End}_k(V)$ with the property that

$$\Delta |o\rangle_\alpha = |o\rangle_\alpha \otimes |o\rangle_\alpha, \qquad d|o\rangle_\alpha = \alpha \,.$$

Then $[\![\underline{2}_A]\!]$ maps this element to

$$|o\rangle_\alpha \longmapsto |o\rangle_\alpha \otimes |o\rangle_\alpha \longmapsto \alpha \otimes \alpha \longmapsto \alpha \circ \alpha \,. \qquad (5.3)$$

This demonstrates how the coalgebra $!\operatorname{End}_k(V)$ may be used to encode nonlinear maps, such as squaring an endomorphism.

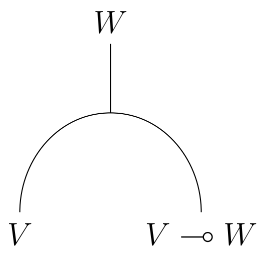
This is almost completely formal: the category of k-vector spaces has all the properties of a category of proof denotations described earlier, including the comonad ! given by taking cofree cocommutative coalgebras, so it is automatic that semantics of intuitionistic linear logic may be constructed within $\mathcal{V}$. However, to explicitly calculate the denotations of proofs we need formulas for promotion and dereliction which are not automatic: in fact, this paper seems to be the first time they have been written down. This is not as surprising as it might seem: although studying semantics of *linear* logic using vector spaces is an natural thing to do, research has focused on full linear logic with negation. This is more complicated than the intuitionistic case, because types must be interpreted by self-dual objects and this leads to topological vector spaces [9, 10, 16, 17, 44, 31, 32]. However these more sophisticated models have the disadvantage that it is inconvenient to write down denotations of proofs, which is why for this introduction we stick to the intuitionistic case. See Remark 5.12 for more on the existing literature.

Our assignment of linear maps to proofs will be presented using string diagrams. One style of string diagrams, called *proof-nets*, were introduced by Girard and have been fundamental to linear logic since the beginning of the subject [37]. However proof-nets are designed for full linear logic and this does not naturally fit with a semantics involving infinite-dimensional vector spaces. Instead we use a style of diagrams which is standard in category theory, following Joyal and Street [51, 52, 57, 54]. Our recommended reference for this approach to proof-nets is Melliès [61, 60].

Let $\mathcal{V}$ denote the category of k-vector spaces (not necessarily finite dimensional). Then $\mathcal{V}$ is symmetric monoidal and for each object V the functor $V \otimes -$ has a right adjoint

$$V \multimap - := \operatorname{Hom}_k(V, -)\,.$$

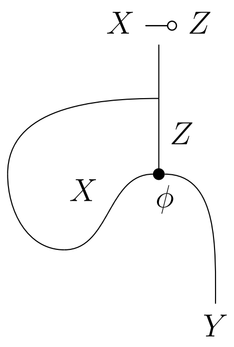
In addition to the usual diagrammatics of a symmetric monoidal category, we draw the counit $V \otimes (V \multimap W) \longrightarrow W$ as



(5.4)

The adjoint $Y \longrightarrow X \multimap Z$ of a morphism $\phi : X \otimes Y \longrightarrow Z$ is depicted as follows:[5]

[5]This is somewhat against the spirit of the diagrammatic calculus, since the loop labelled X is not "real" and is only meant as a "picture" to be placed at a vertex between a strand labelled Y and a strand labelled $X \multimap Z$. This should not cause confusion, because we will never manipulate this strand on its own. The idea is that if X were a finite-dimensional vector space, so that $X \multimap Z \cong X^\vee \otimes Z$, the above diagram would be absolutely valid, and we persist with the same notation even when X is not dualisable. In our judgement the clarity achieved by this slight cheat justifies a little valour in the face of correctness.



(5.5)

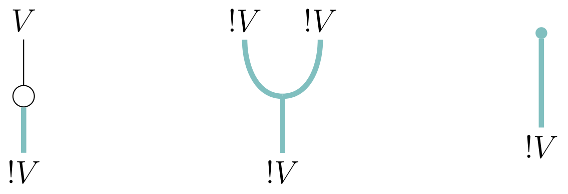
Next we present the categorical construct corresponding to the exponential modality in terms of an adjunction, following Benton [13], see also [61, §7]. Let $\mathcal{C}$ denote the category of counital, coassociative, cocommutative coalgebras in $\mathcal{V}$. In this paper whenever we say *coalgebra* we mean an object of $\mathcal{C}$. This is a symmetric monoidal category in which the tensor product (inherited from $\mathcal{V}$) is cartesian, see [74, Theorem 6.4.5], [8] and [61, §6.5].

By results of Sweedler [74, Chapter 6] the forgetful functor $L : \mathcal{C} \longrightarrow \mathcal{V}$ has a right adjoint R and we set $! = L \circ R$, as in the following diagram:[6]

$$\begin{tikzcd} \mathcal{C} \arrow[r, shift left, "L"] & \mathcal{V} \arrow[l, shift left, "R"] \end{tikzcd} \qquad ! = L \circ R\,.$$

Both L and its adjoint R are monoidal functors.

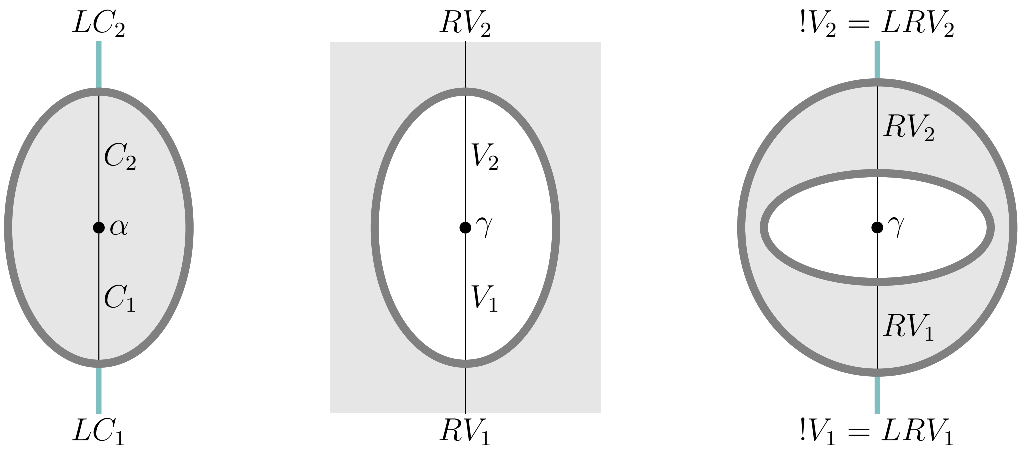
For each V there is a coalgebra $!V$ and a counit of adjunction $d : !V \longrightarrow V$. Since this map will end up being the interpretation of the dereliction rule in linear logic, we refer to it as the *dereliction map*. In string diagrams it is represented by an empty circle. Although it is purely decorative, it is convenient to represent coalgebras in string diagrams drawn in $\mathcal{V}$ by thick lines, so that for $!V$ the dereliction, coproduct and counit are drawn respectively as follows:



(5.6)

In this paper our string diagrams involve both $\mathcal{V}$ and $\mathcal{C}$ and our convention is that white regions represent $\mathcal{V}$ and gray regions stand for $\mathcal{C}$. A standard way of representing monoidal functors between monoidal categories is using coloured regions [61, §5.7]. The image under L of a morphism $\alpha : C_1 \longrightarrow C_2$ in $\mathcal{C}$ is drawn as a vertex in a grey region embedded into a white region. The image of a morphism $\gamma : V_1 \longrightarrow V_2$ under R is drawn using a white region embedded in a gray plane. For example, the diagrams representing $L(\alpha)$, $R(\gamma)$ and $!\gamma = LR(\gamma)$ are respectively

[6]The existence of a right adjoint to the forgetful functor can also be seen to hold more generally as a consequence of the adjoint functor theorem [8].



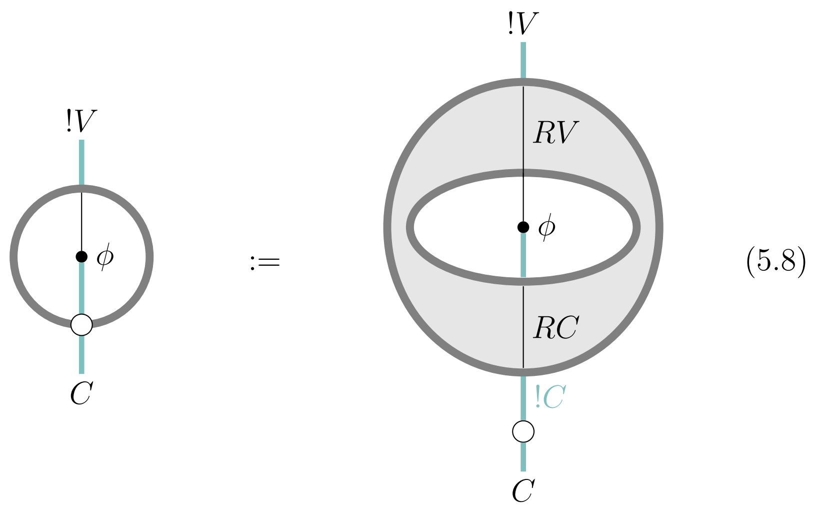
The adjunction between R and L means that for any coalgebra C and linear map $\phi : C \longrightarrow V$ there is a unique morphism of coalgebras $\Phi : C \longrightarrow !V$ making

$$\begin{tikzcd} C \arrow[r, "\phi"] \arrow[dr, "\Phi"'] & V \\ & !V \arrow[u, "d"'] \end{tikzcd} \tag{5.7}$$

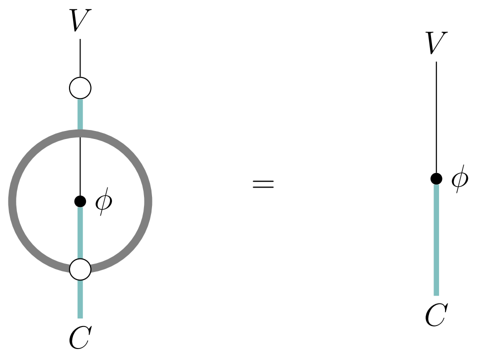
commute. The lifting Φ may be constructed as the unit followed by $!\phi$,

$$\Phi := \; C \longrightarrow !C \xrightarrow{!\phi} !V$$

and since we use an empty circle to denote the unit $C \longrightarrow !C$, this has the diagrammatic representation given on the right hand side of the following diagram. The left hand side is a convenient abbreviation for this morphism, that is, for the lifting Φ:



We follow the logic literature in referring to the grey circle denoting the induced map Φ as a *promotion box*. Commutativity of (5.7) is expressed by the identity



(5.9)

The coalgebra $!V$ and the dereliction map $!V \longrightarrow V$ satisfy a universal property and are therefore unique up to isomorphism. However, to actually understand the denotations of proofs, we will need the more explicit construction which follows from the work of Sweedler [74] and is spelt out in [64]. If V is finite-dimensional then

$$!V = \bigoplus_{P \in V} \mathrm{Sym}_P(V) \tag{5.10}$$

where $\mathrm{Sym}_P(V) = \mathrm{Sym}(V)$ is the symmetric coalgebra. If $e_1, \ldots, e_n$ is a basis for V then as a vector space $\mathrm{Sym}(V) \cong k[e_1, \ldots, e_n]$. The notational convention in [64] is to denote, for elements $\nu_1, \ldots, \nu_s \in V$, the corresponding tensor in $\mathrm{Sym}_P(V)$ using kets

$$|\nu_1, \ldots, \nu_s\rangle_P := \nu_1 \otimes \cdots \otimes \nu_s \in \mathrm{Sym}_P(V)\,. \tag{5.11}$$

And in particular, the identity element of $\mathrm{Sym}_P(V)$ is denoted by a vacuum vector

$$|o\rangle_P := 1 \in \mathrm{Sym}_P(V)\,. \tag{5.12}$$

We remark that if $\nu = 0$ then $|\nu\rangle_P = 0$ is the zero vector, which is distinct from $|o\rangle_P = 1$. We keep in mind that (5.12) denotes the case $s = 0$ of (5.11) and to avoid unwieldy notation we sometimes write $\nu_1 \otimes \cdots \otimes \nu_s \cdot |o\rangle_P$ for $|\nu_1, \ldots, \nu_s\rangle_P$. With this notation the universal map $d : !V \longrightarrow V$ is defined by

$$d|o\rangle_P = P, \quad d|\nu\rangle_P = \nu, \quad d|\nu_1, \ldots, \nu_s\rangle_P = 0 \quad s > 1\,.$$

The coproduct on $!V$ is defined by

$$\Delta|\nu_1, \ldots, \nu_s\rangle_P = \sum_{I \subseteq \{1, \ldots, s\}} |\nu_I\rangle_P \otimes |\nu_{I^c}\rangle_P \tag{5.13}$$

where I ranges over all subsets including the empty set, for a subset $I = \{i_1, \ldots, i_p\}$ we denote by ν_I the sequence $\nu_{i_1}, \ldots, \nu_{i_p}$, and I^c is the complement of I. In particular

$$\Delta|o\rangle_P = |o\rangle_P \otimes |o\rangle_P\,.$$

The counit $!V \longrightarrow k$ is defined by $|o\rangle_P \mapsto 1$ and $|\nu_1, \ldots, \nu_s\rangle_P \mapsto 0$ for $s > 0$.

When V is infinite-dimensional we may define $!V$ as the direct limit over the coalgebras $!W$ for finite-dimensional subspaces $W \subseteq V$. These are sub-coalgebras of $!V$, and we may therefore use the same notation as in (5.11) to denote arbitrary elements of $!V$. Moreover the coproduct, counit and universal map d are given by the same formulas; see [64, §2.1]. A proof of the fact that the map $d : !V \longrightarrow V$ described above is universal among linear maps to V from cocommutative coalgebras is given in [64, Theorem 2.18], but as has been mentioned this is originally due to Sweedler [74], see [64, Appendix B].

To construct semantics of linear logic we also need an explicit description of liftings, as given by the next theorem which is [64, Theorem 2.20]. For a set X the set of partitions of X is denoted $\mathcal{P}_X$.

Theorem 5.4. *Let W, V be vector spaces and $\phi : !W \longrightarrow V$ a linear map. The unique lifting to a morphism of coalgebras $\Phi : !W \longrightarrow !V$ is given by*

$$\Phi|\nu_1, \ldots, \nu_s\rangle_P = \sum_{C \in \mathcal{P}_{\{1,\ldots,s\}}} \phi|\nu_{C_1}\rangle_P \otimes \cdots \otimes \phi|\nu_{C_l}\rangle_P \cdot |o\rangle_Q \tag{5.14}$$

for $P, \nu_1, \ldots, \nu_s \in W$, where $Q = \phi|o\rangle_P$ and l denotes the length of the partition C.

Example 5.5. The simplest example of a coalgebra is the field k. Any $P \in V$ determines a linear map $k \longrightarrow V$ whose lifting to a morphism of coalgebras $k \longrightarrow !V$ sends $1 \in k$ to the vacuum $|o\rangle_P$, as shown in the commutative diagram

$$\begin{tikzcd} k \arrow[r, "P"] \arrow[dr, dotted, "|o\rangle_P"'] & V \\ & !V \arrow[u, "d"'] \end{tikzcd} . \tag{5.15}$$

Such liftings arise from promotions with empty premises, e.g. the proof[7]

$$\dfrac{\dfrac{\overline{A \vdash A}}{\vdash A \multimap A}\ \multimap R}{\vdash !(A \multimap A)}\ \text{prom}$$

5.3 The vector space semantics

Recall from Definition 5.1 the definition of $[\![A]\!]$ for each type A.

[7]Incidentally, this proof helps explain why defining $[\![!A]\!] = \mathrm{Sym}([\![A]\!])$ cannot lead to semantics of linear logic, since the denotation is a morphism of coalgebras $k \longrightarrow !\,\mathrm{End}_k([\![A]\!])$ whose composition with dereliction yields the map $k \longrightarrow \mathrm{End}_k([\![A]\!])$ sending $1 \in k$ to the identity. But this map does not admit a lifting into the symmetric coalgebra, because it produces an infinite sum. However the symmetric coalgebra *is* universal in a restricted sense and is (confusingly) sometimes also called a cofree coalgebra; see [76, §4]. For further discussion of the symmetric coalgebra in the context of linear logic see [19, 62].

Definition 5.6. The *denotation* $[\![\pi]\!]$ of a proof π of $\Gamma \vdash B$ is a linear map $[\![\Gamma]\!] \longrightarrow [\![B]\!]$ defined by inductively assigning a string diagram to each proof tree; by the basic results of the diagrammatic calculus [51] this diagram unambiguously denotes a linear map. The inductive construction is described by the second column in (4.3) – (4.13).

In each rule we assume a morphism has already been assigned to each of the sequents in the numerator of the deduction rule. These inputs are represented by blue circles in the diagram, which computes the morphism to be assigned to the denominator. To simplify the appearance of diagrams, we adopt the convention that a strand labelled by a type A represents a strand labelled by the denotation $[\![A]\!]$. In particular, a strand labelled with a sequence $\Gamma = A_1, \ldots, A_n$ represents a strand labelled by $[\![A_1]\!] \otimes \cdots \otimes [\![A_n]\!]$.

Some comments:

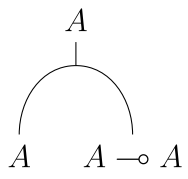
- The diagram for the axiom rule (4.3) depicts the identity of $[\![A]\!]$.
- The diagram for the exchange rule (4.4) uses the symmetry $[\![B]\!] \otimes [\![A]\!] \longrightarrow [\![A]\!] \otimes [\![B]\!]$.
- The diagram for the cut rule (4.5) depicts the composition of the two inputs.
- The right tensor rule (4.6) depicts the tensor product of the two given morphisms, while the left tensor rule (4.7) depicts the identity, viewed as a morphism between two strands labelled $[\![A]\!]$ and $[\![B]\!]$ and a single strand labelled $[\![A]\!] \otimes [\![B]\!]$.
- The diagram for the right $\multimap$ rule (4.8) denotes the adjoint of the input morphism, as explained in (5.5), while the left $\multimap$ rule (4.9) uses the composition map of (5.4).
- The diagram for the promotion rule (4.10) depicts the lifting of the input to a morphism of coalgebras, as explained in (5.8).
- The diagram for the dereliction rule (4.11) depicts the composition of the input with the universal map out of the coalgebra $!V$. The notation for this map, and the maps in the contraction (4.12) and weakening rules (4.13) are as described in (5.6).

Remark 5.7. For this to be a valid semantics, two proofs related by cut-elimination must be assigned the same morphism. This is a consequence of the general considerations in [61, §7]. More precisely, $\mathcal{V}$ is a Lafont category [61, §7.2] and in the terminology of *loc.cit.* the adjunction between $\mathcal{V}$ and $\mathcal{C}$ is a linear-nonlinear adjunction giving rise to a model of intuitionistic linear logic. For an explanation of how the structure of a symmetric monoidal category extrudes itself from the cut-elimination transformations, see [61, §2].

Example 5.8. We convert the proof $\underline{2}_A$ of (4.16) to a diagram in stages, beginning with the leaves. Each stage is depicted in three columns: in the first is a partial proof tree, in the second is the diagram assigned to it by Definition 5.6, and in the third is the explicit linear map which is the value of the diagram.

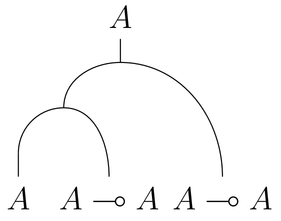
Recall that a strand labelled A actually stands for the vector space $V = [\![A]\!]$, so for instance the first diagram denotes a linear map $V \otimes \mathrm{End}_k(V) \longrightarrow V$:

$$\dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{}{A \vdash A}}{A, A \multimap A \vdash A}\,{\scriptstyle \multimap L}$$



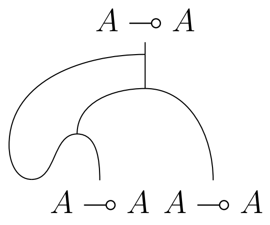
$$a \otimes \alpha \mapsto \alpha(a)$$

$$\dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{}{A \vdash A}}{A, A \multimap A \vdash A}\,{\scriptstyle \multimap L}}{A, A \multimap A, A \multimap A \vdash A}\,{\scriptstyle \multimap L}$$



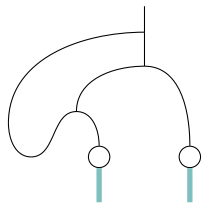
$$a \otimes \alpha \otimes \beta \mapsto \beta(\alpha(a))$$

$$\dfrac{\dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{}{A \vdash A}}{A, A \multimap A \vdash A}\,{\scriptstyle \multimap L}}{A, A \multimap A, A \multimap A \vdash A}\,{\scriptstyle \multimap L}}{A \multimap A, A \multimap A \vdash A \multimap A}\,{\scriptstyle \multimap R}$$



$$\alpha \otimes \beta \mapsto \beta \circ \alpha$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{}{A \vdash A}}{A, A \multimap A \vdash A}\,{\scriptstyle \multimap L}}{A, A \multimap A, A \multimap A \vdash A}\,{\scriptstyle \multimap L}}{A \multimap A, A \multimap A \vdash A \multimap A}\,{\scriptstyle \multimap R}}{!(A \multimap A), A \multimap A \vdash A \multimap A}\,{\scriptstyle \mathrm{der}}}{!(A \multimap A), !(A \multimap A) \vdash A \multimap A}\,{\scriptstyle \mathrm{der}} \tag{5.16}$$

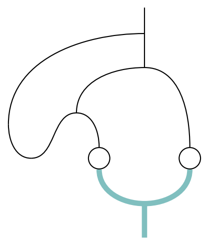


The map $!\operatorname{End}_k(V) \otimes !\operatorname{End}_k(V) \longrightarrow \operatorname{End}_k(V)$ in (5.16) is zero on $|\nu_1, \ldots, \nu_s\rangle_\alpha \otimes |\mu_1, \ldots, \mu_t\rangle_\beta$ for $\alpha, \beta \in \operatorname{End}_k(V)$ unless $s, t \leq 1$, and in those cases it is given by

$$\begin{aligned}
|o\rangle_\alpha \otimes |o\rangle_\beta &\longmapsto \beta \circ \alpha \\
|\nu\rangle_\alpha \otimes |o\rangle_\beta &\longmapsto \beta \circ \nu \\
|o\rangle_\alpha \otimes |\mu\rangle_\beta &\longmapsto \mu \circ \alpha \\
|\nu\rangle_\alpha \otimes |\mu\rangle_\beta &\longmapsto \mu \circ \nu\,.
\end{aligned}$$

The next deduction rule in $\underline{2}_A$ is a contraction:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{}{A \vdash A}}{A, A \multimap A \vdash A}\,{\scriptstyle \multimap L}}{A, A \multimap A, A \multimap A \vdash A}\,{\scriptstyle \multimap L}}{A \multimap A, A \multimap A \vdash A \multimap A}\,{\scriptstyle \multimap R}}{!(A \multimap A), A \multimap A \vdash A \multimap A}\,{\scriptstyle \mathrm{der}}}{!(A \multimap A), !(A \multimap A) \vdash A \multimap A}\,{\scriptstyle \mathrm{der}}}{!(A \multimap A) \vdash A \multimap A}\,{\scriptstyle \mathrm{ctr}} \tag{5.17}$$



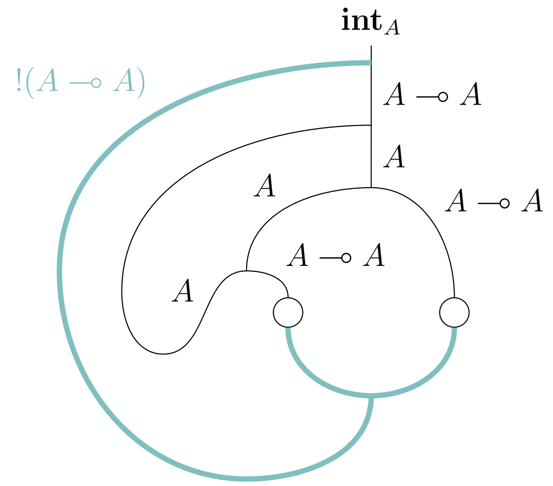
The denotation of this map is the composition $\phi = [\![\underline{2}_A]\!] : !\,\mathrm{End}_k(V) \longrightarrow \mathrm{End}_k(V)$ in (5.2). We may compute using the above that

$$\phi|o\rangle_\alpha = \alpha^2, \qquad \phi|\nu\rangle_\alpha = \{\nu, \alpha\}\,, \qquad \phi|\nu\mu\rangle_\alpha = \{\nu, \mu\}\,. \tag{5.18}$$

For example

$$|\nu\rangle_\alpha \longmapsto |\nu\rangle_\alpha \otimes |o\rangle_\alpha + |o\rangle_\alpha \otimes |\nu\rangle_\alpha \longmapsto \alpha \circ \nu + \nu \circ \alpha = \{\nu, \alpha\}\,.$$

The final step in the proof of $\underline{2}_A$ consists of moving the $!(A \multimap A)$ to the right side of the sequent, which yields the final diagram:



(5.19)

The denotation of this morphism is the map in (5.1). The reader might like to compare this style of diagram for $\underline{2}_A$ to the corresponding proof-net in [37, §5.3.2].

In Example 5.2 we sketched how to recover the function $\alpha \mapsto \alpha^2$ from the denotation of the Church numeral $\underline{2}_A$, but now we can put this on a firmer footing. The translation of the λ-calculus into linear logic encourages us to think of a proof π of a sequent $!B \vdash C$ as a program whose input of type B may be used multiple times. There is *a priori* no linear map $[\![B]\!] \longrightarrow [\![C]\!]$ associated to π but there is a function $[\![\pi]\!]_{nl}$ defined on $P \in [\![B]\!]$ by lifting to $![\![B]\!]$ and then applying $[\![\pi]\!]$:

$$\begin{tikzcd} k \arrow[r, "P"] \arrow[dr, dotted, "|o\rangle_P"'] & {[\![B]\!]} & {[\![C]\!]}\,. \\ & {![\![B]\!]} \arrow[u, "d"] \arrow[ur, "{[\![\pi]\!]}"'] & \end{tikzcd} \tag{5.20}$$

That is,

Definition 5.9. The function $[\![\pi]\!]_{nl} : [\![B]\!] \longrightarrow [\![C]\!]$ is defined by $[\![\pi]\!]_{nl}(P) = [\![\pi]\!]|o\rangle_P$.

The discussion above shows that, with $V = [\![A]\!]$,

Lemma 5.10. $[\![\underline{2}_A]\!]_{nl} : \operatorname{End}_k(V) \longrightarrow \operatorname{End}_k(V)$ *is the map* $\alpha \mapsto \alpha^2$.

This completes our explanation of how to represent the Church numeral $\underline{2}_A$ as a linear map (modulo the evasion discussed in Remark 5.12). From this presentation we see clearly that the non-linearity of the map $\alpha \mapsto \alpha^2$ is concentrated, in fact, not in the duplication step but in the "promotion" step (5.20) where the input vector α is turned into a vacuum $|o\rangle_\alpha \in {!}\operatorname{End}_k(V)$. The promotion step is non-linear since $|o\rangle_\alpha + |o\rangle_\beta$ is not a morphism of coalgebras and thus cannot be equal to $|o\rangle_{\alpha+\beta}$. After this step, the duplication, dereliction and composition shown in (5.3) are all linear. Finally, when $k = \mathbb{C}$ it is interesting to compare $[\![2]\!]_{nl}$ with the map $\alpha \mapsto \alpha^2$ of smooth manifolds $\operatorname{End}_k(V)$, see Appendix B.

Example 5.11. Applied to $\underline{2}_A$ the promotion rule generates a new proof

$$\dfrac{\begin{array}{c}\underline{2}_A\\ \vdots \\ !(A \multimap A) \vdash A \multimap A\end{array}}{!(A \multimap A) \vdash {!}(A \multimap A)}\ \text{prom}$$

which we denote $\operatorname{prom}(\underline{2}_A)$. By definition the denotation $\Phi := [\![\operatorname{prom}(\underline{2}_A)]\!]$ of this proof is the unique morphism of coalgebras

$$\Phi : {!}\operatorname{End}_k(V) \longrightarrow {!}\operatorname{End}_k(V)$$

with the property that $d \circ \Phi = \phi$, where $\phi = [\![\underline{2}_A]\!]$ is as in (5.2). From (5.18) and Theorem 5.4 we compute that, for example

$$\begin{aligned}
\Phi|o\rangle_\alpha &= |o\rangle_{\alpha^2}\,,\\
\Phi|\nu\rangle_\alpha &= \{\nu,\alpha\} \cdot |o\rangle_{\alpha^2},\\
\Phi|\nu\mu\rangle_\alpha &= \big(\{\nu,\mu\} + \{\nu,\alpha\} \otimes \{\mu,\alpha\}\big) \cdot |o\rangle_{\alpha^2}\,,\\
\Phi|\nu\mu\theta\rangle_\alpha &= \big(\{\nu,\mu\} \otimes \{\theta,\alpha\} + \{\theta,\mu\} \otimes \{\nu,\alpha\} + \{\nu,\theta\} \otimes \{\mu,\alpha\}\\
&\quad + \{\nu,\alpha\} \otimes \{\mu,\alpha\} \otimes \{\theta,\alpha\}\big) \cdot |o\rangle_{\alpha^2}\,.
\end{aligned}$$

Note that the commutators, e.g. $\{\nu,\alpha\}$ are defined using the product internal to $\operatorname{End}_k(V)$, whereas inside the bracket in the last two lines, the terms $\{\nu,\alpha\}$ and $\{\mu,\alpha\}$ are multiplied in the algebra $\operatorname{Sym}(\operatorname{End}_k(V))$ before being made to act on the vacuum.

Remark 5.12. As we have already mentioned, there are numerous other semantics of linear logic defined using topological vector spaces. The reader curious about how these vector space semantics are related to the "relational" style of semantics such as coherence spaces should consult Ehrhard's paper on finiteness spaces [31].

Indeed one can generate numerous examples of vector space semantics by looking at comonads on the category $\mathcal{V}$ defined by truncations on the coalgebras $!V$. For example, given a vector space V, let $!_0V$ denote the subspace of $!V$ generated by the vacua $|o\rangle_P$.

This is the free space on the underlying *set* of V, and it is a subcoalgebra of $!V$ given as a coproduct of trivial coalgebras. It is easy to see that this defines an appropriate comonad and gives rise to a semantics of linear logic [75, §4.3]. However the semantics defined using $!V$ is more interesting, because universality allows us to mix in arbitrary coalgebras C. In Appendix B we examine the simplest example where C is the dual of the algebra $k[t]/t^2$ and relate this to tangent vectors.

6 Cut-elimination

The dynamical part of linear logic is a rewrite rule on proofs called *cut-elimination*, which defines the way that proofs interact. Collectively these interactions give the connectives their meaning. The full set of rewrite rules is given in [61, Section 3] and in the alternative language of proof-nets in [37, §4], [65, p.18]. While cut-elimination is the analogue in linear logic of β-reduction in the λ-calculus, the definition of the rewrite rules that make up cut-elimination is substantially more complicated; this is the price of consistently tracking the duplication of terms, which is ignored in the λ-calculus.

In order to help the reader grasp the core ideas without getting bogged down in the details, we present a worked example involving our favourite proof $\underline{2}_A$ where we give both the rewrites and the associated transformations of string diagrams.

Definition 6.1. Let $\Gamma \vdash A$ be a sequent of linear logic and $\mathcal{P}$ the set of proofs. There is a binary relation $\rightsquigarrow$ on $\mathcal{P}$ defined by saying that $\pi \rightsquigarrow \rho$ if ρ is an immediate reduction of π by one of the rewrite rules listed in [61, Section 3]. These rewrite rules are sometimes referred to as *cut-elimination transformations.*

We write $=_{cut}$ for the smallest reflexive, transitive and symmetric relation on $\mathcal{P}$ containing $\rightsquigarrow$ and call this *equivalence under cut-elimination.*

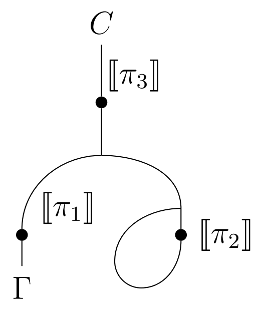
Example 6.2. In the context of Example 4.3 there is a sequence of transformations

$$\underline{2}_A \,|\, \mathrm{prom}(\pi) = \pi_0 \rightsquigarrow \pi_1 \rightsquigarrow \cdots \rightsquigarrow \pi \,|\, \pi$$

which begins with the rewrites [61, §3.9.3] and [61, §3.9.1].

Example 6.3. The cut-elimination transformation [61, §3.8.2] tells us that

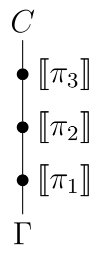
$$\dfrac{\dfrac{\begin{array}{c}\pi_2\\ \vdots\\ A\vdash B\end{array}}{\vdash A \multimap B}{\scriptstyle \multimap R} \qquad \dfrac{\begin{array}{c}\pi_1\\ \vdots \\ \Gamma\vdash A\end{array} \quad \begin{array}{c}\pi_3\\ \vdots\\ B\vdash C\end{array}}{\Gamma, A\multimap B\vdash C}{\scriptstyle \multimap L}}{\Gamma\vdash C}{\scriptstyle \mathrm{cut}}$$



(6.1)

may be transformed to

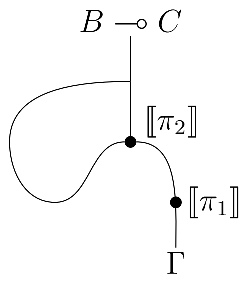
$$\dfrac{\dfrac{\begin{matrix}\pi_1\\ \vdots\\ \Gamma \vdash A\end{matrix} \qquad \begin{matrix}\pi_2\\ \vdots\\ A\vdash B\end{matrix}}{\Gamma\vdash B}\,\text{cut} \qquad \begin{matrix}\pi_3\\ \vdots\\ B\vdash C\end{matrix}}{\Gamma\vdash C}\,\text{cut} \tag{6.2}$$



Observe how this cut-elimination transformation encodes the meaning of the left introduction rule $\multimap L$, in the sense that it takes a proof π_1 of $\Gamma \vdash A$ and a proof π_3 of $B \vdash C$ and says: if you give me a proof of $A \vdash B$ I know how to sandwhich it between π_1 and π_3.

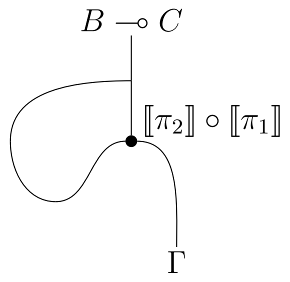
Example 6.4. The cut-elimination transformation [61, §3.11.10] tells us that

$$\dfrac{\begin{matrix}\pi_1\\ \vdots\\ \Gamma \vdash A\end{matrix} \qquad \dfrac{\begin{matrix}\pi_2\\ \vdots\\ B, A\vdash C\end{matrix}}{A\vdash B\multimap C}\,{\multimap R}}{\Gamma\vdash B\multimap C}\,\text{cut} \tag{6.3}$$



is transformed to the proof

$$\dfrac{\dfrac{\begin{matrix}\pi_1\\ \vdots\\ \Gamma \vdash A\end{matrix} \qquad \begin{matrix}\pi_2\\ \vdots\\ B, A\vdash C\end{matrix}}{B,\Gamma\vdash C}\,\text{cut}}{\Gamma\vdash B\multimap C}\,{\multimap R} \tag{6.4}$$



and thus the two corresponding diagrams are equal. In this case, the equality generated by cut-elimination expresses the fact that the Hom-tensor adjunction is natural.

Example 6.5. The cut-elimination transformation [61, §3.6.1] tells us that

$$\dfrac{\dfrac{}{A\vdash A} \qquad \begin{matrix}\pi\\ \vdots\\ A\vdash B\end{matrix}}{A\vdash B}\,\text{cut} \tag{6.5}$$

may be transformed to the proof π.

The analogue in linear logic of a normal form in λ-calculus is

Definition 6.6. A proof is *cut-free* if it contains no occurrences of the cut rule.

The first theorem is that the relation $\rightsquigarrow$ is *weakly normalising*, by which we mean:

Theorem 6.7 (Cut-elimination)**.** *For each proof π in linear logic there is a sequence*

$$\pi = \pi_0 \rightsquigarrow \pi_1 \rightsquigarrow \cdots \rightsquigarrow \pi_n \tag{6.6}$$

where π_n is cut-free.

Proof. The original proof by Girard is given in the language of proof-nets [37] which for various reasons are better suited to the problem. For a proof in the language of sequent calculus see for example [21, Appendix B]. The argument follows Gentzen's proof for cut-elimination in classical logic; see [35] and [46, Chapter 13]. $\square$

This is the analogue of Gentzen's *Hauptsatz* for ordinary logic, which was mentioned in the introduction. At a high level, the sequence of rewrites (6.6) eliminates the implicitness present in the original proof until the fully explicit, cut-free proof π_n is left. In practice there are various kinds of cut-elimination transformations, for example:

- those like (6.3) $\rightsquigarrow$ (6.4) in Example 6.3 which commute cuts past other deduction rules or otherwise rearrange the content of the proof.
- those like (6.1) $\rightsquigarrow$ (6.2) in Example 6.4 which replace a cut on a complex formula $A \multimap B$ by cuts on simpler formulae A, B.
- those like the transformation in Example 6.5 which actually erase part of the proof.

Remark 6.8. It is possible that multiple rewrite rules apply to a proof, so that there is more than one sequence (6.6) with π as its starting point. This raises the question of whether the end result π_n is independent of the choices at each stage. This property of a rewrite rule is called *confluence*. Unfortunately in its sequent calculus presentation linear logic is *not* confluent and in particular it is not true that each cut-elimination equivalence class contains a unique cut-free proof.

However it is true that two cut-free proofs in the same equivalence class are "essentially the same" in the sense that they are assigned the same proof-net. That is, cut-elimination of proof-nets is confluent, according to [65, Remark 4.19]. The reader is warned that in this direction things quickly become technical!

6.1 An extended example

To demonstrate cut-elimination in a nontrivial example, let us prove that $2 \times 2 = 4$.

Definition 6.9. We define $\underline{\mathrm{mult}}_A(m, -)$ to be the proof (writing $E = A \multimap A$)

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\underline{m}_A\\ \vdots\\ !E \vdash E\end{matrix}}{!E \vdash !E}\,\text{prom} \qquad \dfrac{}{E \vdash E}}{!E, \mathbf{int}_A \vdash E}\,\multimap L}{\mathbf{int}_A \vdash \mathbf{int}_A}\,\multimap R \tag{6.7}$$

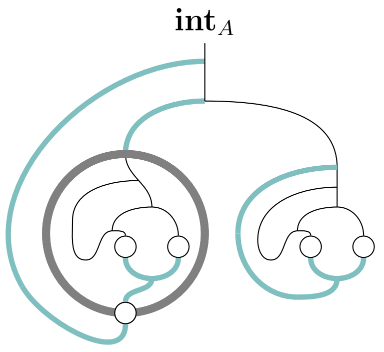
This proof represents multiplication by m, in the following sense.

Lemma 6.10. *The proof obtained from* $\underline{\mathrm{mult}}_A(m,-)$ *by cutting against* $\underline{n}_A$

$$\dfrac{\begin{matrix}\underline{n}_A\\ \vdots\\ \vdash \mathbf{int}_A\end{matrix} \qquad \begin{matrix}\underline{\mathrm{mult}}_A(m,-)\\ \vdots\\ \mathbf{int}_A \vdash \mathbf{int}_A\end{matrix}}{\vdash \mathbf{int}_A}\,\textit{cut} \tag{6.8}$$

is equivalent under cut-elimination to $\underline{mn}_A$.

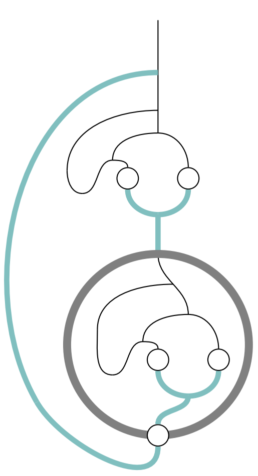
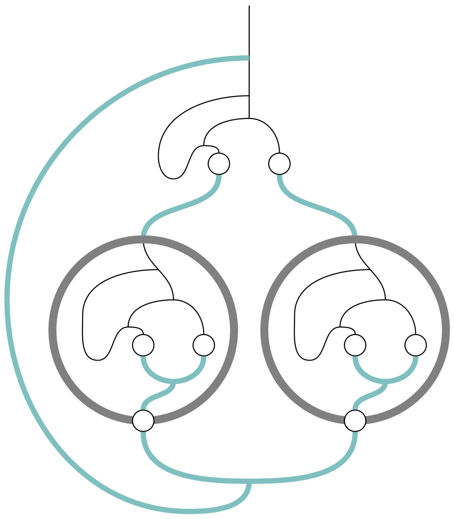
Rather than give a complete proof of this lemma, we examine the special case where $m = n = 2$. If we denote the proof in (6.8) by $\underline{\mathrm{mult}}_A(2,-)\,|\,\underline{2}_A$ then its string diagrams is



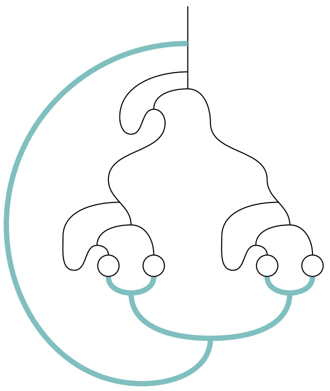
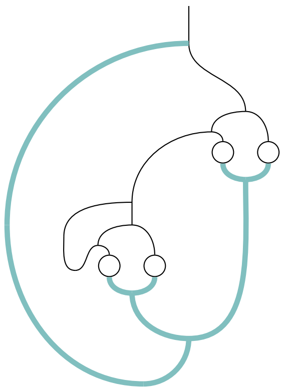
(6.9)

To prove that the cut-free normalisation of $\underline{\mathrm{mult}}_A(2,-)\,|\,\underline{2}_A$ is the Church numeral $\underline{4}_A$ we run through the proof transformations generated by the cut-elimination algorithm, each of which yields a new proof with the same denotation. This sequence of proofs represents a particular sequence of manipulations of the string diagram.

To (6.9) we apply naturality of the Hom-tensor adjunction, as in (6.3) $\rightsquigarrow$ (6.4). Then we are in the position of (6.1), with π_2 a part of $\underline{2}_A$ and π_1 the promoted Church numeral. The step (6.1) $\rightsquigarrow$ (6.2) is to take the left leg and feed it as an input to the right leg. This yields the first equality below. The second equality follows from the fact that a promotion box represents a morphism of coalgebras, and thus can be commuted past the coproduct whereby it is duplicated:

(6.9) $=$  $=$  (6.10)

At this point the promotions cancel with the derelictions by the identity (5.9), "releasing" the pair of Church numerals contained in the promotion boxes. This yields the first equality below, while the second is an application of the general form of (6.1) $\rightsquigarrow$ (6.2):

(6.10) $=$  $=$ 

This last diagram is the denotation of $\underline{4}_A$, so we conclude that (at least at the level of the denotations) the output of the program $\underline{\text{mult}}_A(2,-)$ on the input $\underline{2}_A$ is $\underline{4}_A$.

7 Second-order linear logic

The propositional linear logic introduced in the previous sections is not very "expressive" in the sense that only a limited range of functions on integers may be encoded as proofs. To remedy this we need to add quantifiers, and the resulting language is referred to as *second-order* intuitionistic linear logic. The corresponding semantics is much less clear than the propositional case, so we will not speak about it here; see however [**?**, **?**].

Our aim in this section is to begin by writing down some arithmetic in the propositional part of linear logic, and then to show how iteration on higher types leads naturally to the need for quantifiers in order to properly represent towers of exponentials. Recall that the type of integers on A is $\mathbf{int}_A = !(A \multimap A) \multimap (A \multimap A)$.

The following examples are taken from [37, §5.3.2] and [26, §4].

Definition 7.1. We define $\underline{\mathrm{comp}}_A$ to be the proof

$$\dfrac{\dfrac{\dfrac{}{A \vdash A} \qquad \dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{A \vdash A}}{A, A \multimap A \vdash A}\,{\scriptstyle \multimap L}}{A, A \multimap A, A \multimap A \vdash A}\,{\scriptstyle \multimap L}}{A \multimap A, A \multimap A \vdash A \multimap A}\,{\scriptstyle \multimap R} \tag{7.1}$$

As discussed in Remark 5.8 the denotation of this proof is the function which *composes* two endomorphisms of $[\![A]\!]$.

Example 7.2 (Addition). We define $\underline{\mathrm{add}}_A$ to be the proof (writing $E = A \multimap A$)

$$\dfrac{\dfrac{\dfrac{\dfrac{}{!E \vdash !E} \qquad \dfrac{\dfrac{}{!E \vdash !E} \qquad \begin{array}{c}\underline{\mathrm{comp}}_A\\ \vdots \\ E, E \vdash E\end{array}}{!E, E, \mathbf{int}_A \vdash E}\,{\scriptstyle \multimap L}}{!E, !E, \mathbf{int}_A, \mathbf{int}_A \vdash E}\,{\scriptstyle \multimap L}}{!E, \mathbf{int}_A, \mathbf{int}_A \vdash E}\,{\scriptstyle \mathrm{ctr}}}{\mathbf{int}_A, \mathbf{int}_A \vdash \mathbf{int}_A}\,{\scriptstyle \multimap R} \tag{7.2}$$

which encodes addition on A-integers in the following sense: if $\underline{\mathrm{add}}_A$ is cut against two proofs $\underline{m}_A$ and $\underline{n}_A$ the resulting proof is equivalent under cut-elimination to $\underline{m+n}_A$.

Let us call the result of the cut $\underline{\mathrm{add}}_A(m, n)$, which is a proof of $\mathbf{int}_A$. One way to see that this reduces to $\underline{m+n}_A$ without laboriously performing cut-elimination by hand is to use a term calculus, as presented in for example [1, 14] or [26, §4], to understand the computational content of the proof. Alternatively, with the vector space semantics in mind, one can understand the proof as follows: given a vacuum $|o\rangle_\alpha$ at an endomorphism α of $V = [\![A]\!]$, the contraction step duplicates this to $|o\rangle_\alpha \otimes |o\rangle_\alpha$. The left hand copy of $|o\rangle_\alpha$ is fed into $\underline{m}_A$ yielding α^m and the right hand copy is fed into $\underline{n}_A$ yielding α^n. Then the composition "machine" composes these outputs to yield α^{m+n}.

To generate a hierarchy of increasingly complex proofs from addition and multiplication we employ *iteration*.

Example 7.3 (Iteration). Given a proof β of A and β' of $A \multimap A$ we define $\underline{\mathrm{rec}}_A(\beta, \beta')$ to be the proof

$$\dfrac{\dfrac{\begin{array}{c}\beta'\\ \vdots\\ \vdash A \multimap A\end{array}}{\vdash\, !(A\multimap A)}\,\text{prom} \qquad \dfrac{\begin{array}{c}\beta\\ \vdots\\ \vdash A\end{array} \quad \dfrac{}{A\vdash A}}{A\multimap A\vdash A}\,{\multimap}\, L}{\mathbf{int}_A \vdash A}\,{\multimap}\, L \tag{7.3}$$

Cut against $\underline{n}_A$ this yields the nth power of β' applied to β. By analogy with induction in ordinary mathematical argument, often β' is referred to as the *step function* and β as the *base case*.

This game gets more interesting when we get to exponentials. From Definition 6.9 we know how to define a proof $\underline{\text{mult}}_A(m,-)$ whose cut against $\underline{n}_A$ yields something equivalent under cut-elimination to $\underline{mn}_A$. But how do we construct a proof which, when cut against $\underline{n}_A$, yields a proof equivalent to $\underline{m^n}_A$? Naturally we can iterate multiplication by m, but the catch is that this requires integers of type $\mathbf{int}_{\mathbf{int}_A}$.

Example 7.4 (Exponentials). We define $\underline{\text{exp}}_{A,m}$ to be

$$\underline{\text{exp}}_{A,m} = \underline{\text{rec}}_{\mathbf{int}_A}\big(\underline{1}_{\mathbf{int}_A}, \underline{\text{mult}}_A(m,-)\big)\,.$$

That is,

$$\dfrac{\dfrac{\begin{array}{c}\underline{\text{mult}}_A(m,-)\\ \vdots\\ \vdash \mathbf{int}_A \multimap \mathbf{int}_A\end{array}}{\vdash\, !(\mathbf{int}_A\multimap \mathbf{int}_A)}\,\text{prom} \qquad \dfrac{\begin{array}{c}\underline{1}_{\mathbf{int}_A}\\ \vdots\\ \vdash \mathbf{int}_A\end{array} \quad \dfrac{}{\mathbf{int}_A\vdash \mathbf{int}_A}}{\mathbf{int}_A\multimap \mathbf{int}_A\vdash \mathbf{int}_A}\,{\multimap}\, L}{\mathbf{int}_{\mathbf{int}_A} \vdash \mathbf{int}_A}\,{\multimap}\, L \tag{7.4}$$

Cut against $\underline{n}_{\mathbf{int}_A}$ this yields the desired numeral $\underline{m^n}_A$.

We begin to see the general pattern: to represent more complicated functions $\mathbb{N} \longrightarrow \mathbb{N}$ we need to use more complicated base types B for the integers $\mathbf{int}_B$ on the left hand side. At the next step, when we try to iterate exponentials, we see that it is hopeless to continue without introducing a way to parametrise over these base types. That is precisely the role of quantifiers in second-order logic.

The iteration of the function $n \mapsto 2^n$ yields a tower of exponentials of variable height. More precisely, $T : \mathbb{N} \longrightarrow \mathbb{N}$ is defined recursively by $T(0) = 1, T(n+1) = 2^{T(n)}$ so that

$$T(n) = 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}$$

where the total number of occurrences of 2 on the right hand side is n. To represent the function T as a proof in linear logic we need to introduce integer types with iterated subscripts by the recursive definition

$$\mathbf{int}_A(0) = \mathbf{int}_A, \qquad \mathbf{int}_A(r+1) = \mathbf{int}_{\mathbf{int}_A(r)}\,.$$

From Example 7.4 we have, for each r, the proof $\underline{\mathrm{exp}}_{\mathbf{int}_A(r),2}$ of $\mathbf{int}_A(r+1) \vdash \mathbf{int}_A(r)$. By iteration of the exponential we mean the cut of the following proofs:

$$\begin{array}{ccccc} \underline{\mathrm{exp}}_{\mathbf{int}_A(n),2} & & \underline{\mathrm{exp}}_{\mathbf{int}_{\mathbf{int}_A},2} & \underline{\mathrm{exp}}_{\mathbf{int}_A,2} \\ \vdots & \cdots & \vdots & \vdots \\ \mathbf{int}_A(n+1) \vdash \mathbf{int}_A(n) & & \mathbf{int}_{\mathbf{int}_{\mathbf{int}_A}} \vdash \mathbf{int}_{\mathbf{int}_A} & \mathbf{int}_{\mathbf{int}_A} \vdash \mathbf{int}_A \end{array} \qquad (7.5)$$

The result of these cuts is a proof of $\mathbf{int}_A(n+1) \vdash \mathbf{int}_A$ which, when cut against $\underline{n}_{\mathbf{int}_A(n)}$, yields a proof which reduces under cut-elimination to $\underline{T(n)}_A$.

However this proof is constructed "by hand" for each integer n. It is clear from the definition of (7.5) that what we are doing in essence is iterating the exponential function, but we cannot express this formally in the language of propositional linear logic because the base type on our integers changes from iteration to iteration. This leads us to second-order linear logic. After defining it, we will return to the problem of encoding towers of exponentials as proofs.

Definition 7.5 (Second-order linear logic). The formulas of *second-order linear logic* are defined recursively as follows: any formula of propositional linear logic is a formula, and if A is a formula then so is $\forall x \,.\, A$ for any propositional variable x. There are two new deduction rules:

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x \,.\, A}\,\forall R \qquad\qquad \frac{\Gamma, A[B/x] \vdash C}{\Gamma, \forall x \,.\, A \vdash C}\,\forall L$$

where Γ is a sequence of formulas, possibly empty, and in the right introduction rule we require that x is not free in any formula of Γ. Here $A[B/x]$ means a formula A with all free occurrences of x replaced by a formula B (as usual, there is some chicanery necessary to avoid free variables in B being captured, but we ignore this).

Intuitively, the right introduction rule takes a proof π and "exposes" the variable x, for which any type may be substituted. The left introduction rule, dually, takes a formula B in some proof and binds it to a variable x. The result of cutting a right introduction rule against a left introduction rule is that the formula B will be bound to x throughout the proof π. That is, cut-elimination transforms

$$\dfrac{\dfrac{\begin{array}{c}\pi\\ \vdots\\ \Gamma\vdash A\end{array}}{\Gamma\vdash\forall x\,.\,A}\,{\scriptstyle\forall R} \qquad \dfrac{\begin{array}{c}\rho\\ \vdots\\ \Delta, A[B/x]\vdash C\end{array}}{\Delta,\forall x\,.\,A\vdash C}\,{\scriptstyle\forall L}}{\Gamma,\Delta\vdash C}\,{\scriptstyle\text{cut}} \tag{7.6}$$

to the proof

$$\dfrac{\begin{array}{c}\pi[B/x]\\ \vdots\\ \Gamma\vdash A[B/x]\end{array} \qquad \begin{array}{c}\rho\\ \vdots\\ \Delta, A[B/x]\vdash C\end{array}}{\Gamma,\Delta\vdash C}\,{\scriptstyle\text{cut}} \tag{7.7}$$

where $\pi[B/x]$ denotes the result of replacing all occurrences of x in the proof π with B. In the remainder of this section we provide a taste of just what an *explosion* of possibilities the addition of quantifiers to the language represents.

Example 7.6 (Integers). The type of integers is

$$\mathbf{int} = \forall x\,.\,!(x \multimap x) \multimap (x \multimap x)\,.$$

For each integer $n \geq 0$ we define $\underline{n}$ to be the proof

$$\dfrac{\begin{array}{c}\mathbf{int}_x\\ \vdots\\ \vdash \mathbf{int}_x\end{array}}{\vdash \mathbf{int}}\,{\scriptstyle\forall R}$$

Example 7.7 (Exponentials). We define $\underline{\exp}_m$ to be

$$\dfrac{\dfrac{\begin{array}{c}\underline{\exp}_{x,m}\\ \vdots\\ \mathbf{int}_{\mathbf{int}_x}\vdash \mathbf{int}_x\end{array}}{\mathbf{int}\vdash \mathbf{int}_x}\,{\scriptstyle\forall L}}{\mathbf{int}\vdash \mathbf{int}}\,{\scriptstyle\forall R}$$

Example 7.8 (Hyper-exponentials). We define $\underline{\text{hypexp}}$ to be the iteration of $\underline{\exp}_2$, in the following sense:

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\underline{\mathrm{exp}_2}\\ \vdots\\ \vdash \mathbf{int} \multimap \mathbf{int}\end{matrix}}{\vdash !(\mathbf{int} \multimap \mathbf{int})}\,\text{prom} \qquad \dfrac{\begin{matrix}\underline{1}\\ \vdots\\ \vdash \mathbf{int}\end{matrix} \quad \dfrac{}{\mathbf{int} \vdash \mathbf{int}}}{\mathbf{int} \multimap \mathbf{int} \vdash \mathbf{int}}\,{\multimap L}}{\mathbf{int}_{\mathbf{int}} \vdash \mathbf{int}}\,{\multimap L}}{\mathbf{int} \vdash \mathbf{int}}\,\forall L \tag{7.8}$$

Lemma 7.9. *The cut of* $\underline{n}$ *against* $\underline{\mathrm{hypexp}}$ *reduces to* $\underline{T(n)}$.

Once we have added quantifiers, the expressive power of the language is immense. We can for example easily iterate hyper-exponentials to obtain a tower of exponentials whose height is itself a tower of exponentials, and by iterating at the type $\mathbf{int} \multimap \mathbf{int}$ obtain monsters like the Ackermann function [45, §6.D].

Theorem 7.10. *Any function* $\mathbb{N} \longrightarrow \mathbb{N}$ *which is provably total in second-order Peano arithmetic can be encoded into second-order linear logic as a proof of* $\mathbf{int} \vdash \mathbf{int}$.

A Example of cut-elimination

We examine the beginning of the cut-elimination process applied to the proof (6.9). Our reference for cut-elimination is Melliès [61, §3.3]. Throughout a formula A is fixed and we write $E = A \multimap A$ so that $\mathbf{int}_A = !E \multimap E$. We encourage the reader to put the following series of proof trees side-by-side with the evolving diagrams in the above to see the correspondence between cut-elimination and diagram manipulation.

To begin, we expose the first layer of structure within $\underline{\mathrm{mult}}_A(2,-)$ to obtain

$$\dfrac{\begin{matrix}\underline{2}_A\\ \vdots\\ \vdash \mathbf{int}_A\end{matrix} \qquad \dfrac{\begin{matrix}\underline{\mathrm{mult}}_A(2,-)\\ \vdots\\ !E, \mathbf{int}_A \vdash E\end{matrix}}{\mathbf{int}_A \vdash \mathbf{int}_A}\,{\multimap R}}{\vdash \mathbf{int}_A}\,\text{cut} \tag{A.1}$$

For a cut against a proof whose last deduction rule is a right introduction rule for $\multimap$, the cut elimination procedure [61, §3.11.10] prescribes that (A.1) be transformed to

$$\dfrac{\dfrac{\begin{matrix}\underline{2}_A\\ \vdots\\ \vdash \mathbf{int}_A\end{matrix} \qquad \begin{matrix}\underline{\mathrm{mult}}_A(2,-)\\ \vdots\\ !E, \mathbf{int}_A \vdash E\end{matrix}}{!E \vdash E}\,\text{cut}}{\vdash \mathbf{int}_A}\,{\multimap R} \tag{A.2}$$

If we fill in the content of $\underline{\text{mult}}_A(2,-)$, this proof may be depicted as follows:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\underline{2}'_A\\ \vdots\\ !E\vdash E\end{array}}{\vdash \mathbf{int}_A} \qquad \dfrac{\dfrac{\begin{array}{c}\underline{2}'_A\\ \vdots\\ !E\vdash E\end{array}}{!E\vdash !E}\,\text{prom} \qquad \dfrac{}{E\vdash E}}{!E,\mathbf{int}_A\vdash E}\,\multimap L}{!E\vdash E}\,\text{cut}}{\vdash \mathbf{int}_A}\,\multimap R \tag{A.3}$$

The next cut-elimination step [61, §3.8.2] transforms this proof to

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\underline{2}'_A\\ \vdots\\ !E\vdash E\end{array}}{!E\vdash !E}\,\text{prom} \qquad \begin{array}{c}\underline{2}'_A\\ \vdots\\ !E\vdash E\end{array}}{!E\vdash E}\,\text{cut} \qquad \dfrac{}{E\vdash E}}{!E\vdash E}\,\text{cut}}{\vdash \mathbf{int}_A}\,\multimap R \tag{A.4}$$

As may be expected, cutting against an axiom rule does nothing, so this is equivalent to

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\underline{2}'_A\\ \vdots\\ !E\vdash E\end{array}}{!E\vdash !E}\,\text{prom} \qquad \dfrac{\begin{array}{c}\underline{2}''_A\\ \vdots\\ !E,!E\vdash E\end{array}}{!E\vdash E}\,\text{ctr}}{!E\vdash E}\,\text{cut}}{\vdash \mathbf{int}_A}\,\multimap R$$

where $\underline{2}''_A$ is a sub-proof of $\underline{2}_A$. Here is the important step: cut-elimination replaces a cut of a promotion against a contraction by a pair of promotions [61, §3.9.3]. This step corresponds to the doubling of the promotion box in (6.10)

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\underline{2}'_A\\ \vdots\\ !E\vdash E\end{array}}{!E\vdash !E} \qquad \dfrac{\dfrac{\begin{array}{c}\underline{2}'_A\\ \vdots\\ !E\vdash E\end{array}}{!E\vdash !E} \qquad \begin{array}{c}\underline{2}''_A\\ \vdots\\ !E,!E\vdash E\end{array}}{!E,!E\vdash E}\,\text{cut}}{!E,!E\vdash E}\,\text{cut}}{!E\vdash E}\,\text{ctr}}{\vdash \mathbf{int}_A}\,\multimap R$$

We only sketch the rest of the cut-elimination process: next, the derelictions in $\underline{2}''_A$ will be annihilate with the promotions in the two copies of $\underline{2}'_A$ according to [61, §3.9.1]. Then there are numerous eliminations involving the right and left $\multimap$ introduction rules.

B Tangents and proofs

Example B.1. Let $\mathcal{T}$ denote the dual of the finite-dimensional algebra $k[t]/(t^2)$. It has a k-basis $1 = 1^*$ and $\varepsilon = t^*$ and coproduct Δ and counit u defined by

$$\Delta(1) = 1 \otimes 1, \quad \Delta(\varepsilon) = 1 \otimes \varepsilon + \varepsilon \otimes 1, \quad u(1) = 1, \quad u(\varepsilon) = 0\,.$$

Recall that a tangent vector at a point x on a scheme X is a morphism $\operatorname{Spec}(k[t]/t^2) \longrightarrow X$ sending the closed point to x. Given a finite-dimensional vector space V and $R = \operatorname{Sym}(V^*)$ with $X = \operatorname{Spec}(R)$, this is equivalent to a morphism of k-algebras

$$\varphi : \operatorname{Sym}(V^*) \longrightarrow k[t]/t^2$$

with $\varphi^{-1}((t)) = x$. Such a morphism of algebras is determined by its restriction to V^*, which as a linear map $\varphi|_{V^*} : V^* \longrightarrow k[t]/t^2$ corresponds to a pair of elements (P, Q) of V, where $\varphi(\tau) = \tau(P) \cdot 1 + \tau(Q) \cdot t$. Then φ sends a polynomial f to

$$\varphi(f) = f(P) \cdot 1 + \partial_Q(f)|_P \cdot t\,.$$

The map $\varphi|_{V^*}$ is also determined by its dual, which is a linear map $\phi : \mathcal{T} \longrightarrow V$. By the universal property, this lifts to a morphism of coalgebras $\Phi : \mathcal{T} \longrightarrow\, !V$. If ϕ is determined by a pair of points $(P, Q) \in V^{\oplus 2}$ as above, then it may checked directly that

$$\Phi(1) = |o\rangle_P, \qquad \Phi(\varepsilon) = |Q\rangle_P$$

is a morphism of coalgebras lifting ϕ.

Motivated by this example, we make a preliminary investigation into tangent vectors at proof denotations. Let A, B be types with finite-dimensional denotations $[\![A]\!], [\![B]\!]$.

Definition B.2. Given a proof π of $\vdash A$ a *tangent vector* at π is a morphism of coalgebras $\theta : \mathcal{T} \longrightarrow\, ![\![A]\!]$ with the property that $\theta(1) = |o\rangle_{[\![\pi]\!]}$, or equivalently that the diagram

$$\begin{tikzcd} k \arrow[r, "{[\![\pi]\!]}"] \arrow[d, "1"'] & {[\![A]\!]} \\ \mathcal{T} \arrow[r, "\theta"'] & {![\![A]\!]} \arrow[u, "d"'] \end{tikzcd} \tag{B.1}$$

commutes. The space of tangent vectors at π is denoted T_π.

It follows from Example B.1 that there is a linear isomorphism

$$[\![A]\!] \longrightarrow T_\pi$$

sending $Q \in [\![A]\!]$ to the coalgebra morphism θ with $\theta(1) = |o\rangle_{[\![\pi]\!]}$ and $\theta(\varepsilon) = |Q\rangle_{[\![\pi]\!]}$.

Note that the denotation of a program not only maps inputs to outputs (if we identify inputs and outputs with vacuum vectors) but also tangent vectors to tangent vectors. To wit, if ρ is a proof of a sequent $!A \vdash B$ with denotation $\lambda : ![\![A]\!] \longrightarrow [\![B]\!]$, then composing a tangent vector θ at a proof π of $\vdash A$ with the lifting Λ of λ leads to a tangent vector at the cut of ρ against the promotion of π. That is, the linear map

$$\mathcal{T} \xrightarrow{\quad\theta\quad} ![\![A]\!] \xrightarrow{\quad\Lambda\quad} ![\![B]\!] \tag{B.2}$$

is a tangent vector at the following proof, which we denote $\rho \,|\, \pi$

$$\dfrac{\dfrac{\begin{matrix}\pi\\ \vdots\\ \vdash A\end{matrix}}{\vdash\, !A}\,\text{prom} \qquad \begin{matrix}\rho\\ \vdots\\ !A \vdash B\end{matrix}}{\vdash B}\,\text{cut}$$

By Theorem 5.4 the linear map of tangent spaces induced in this way by ρ is

$$\begin{aligned} [\![A]\!] \cong T_\pi &\longrightarrow T_{\rho\,|\,\pi} \cong [\![B]\!] \\ Q &\longmapsto \lambda|Q\rangle_{[\![\pi]\!]} \end{aligned} \tag{B.3}$$

When ρ computes a smooth map of differentiable manifolds, this map can be compared with an actual map of tangent spaces. We examine $\rho = \underline{2}_A$ below. It would be interesting to understand these maps in more complicated examples; this seems to be related to the differential λ-calculus [33, 34], but we have not tried to work out the precise connection.

Example B.3. When $k = \mathbb{C}$ and $Z = [\![\underline{2}_A]\!]_{nl}$ we have by Lemma 5.10

$$Z : M_n(\mathbb{C}) \longrightarrow M_n(\mathbb{C}), \qquad Z(\alpha) = \alpha^2\,.$$

The tangent map of the smooth map of manifolds Z at α is $(Z_*)_\alpha(\nu) = \{\nu, \alpha\}$. When α is the denotation of some proof π of $\vdash A \multimap A$ this agrees with the tangent map assigned in (B.3) to the proof $\underline{2}_A$ at π, using (5.18).

C Translation

We have already mentioned the equivalence of the simply-typed λ-calculus and propositional intuitionistic logic under the Curry-Howard isomorphism [71, §6.5] (TODO no we haven't). There is an embedding of propositional intuitionistic logic into propositional intuitionistic *linear* logic [37, §5.1] making use of the additive connectives of linear logic which we have omitted in the above. This means that every program in the simply-typed λ-calculus may be assigned a proof in linear logic (with additives) in such a way that β-reduction in the λ-calculus corresponds to cut-elimination in linear logic. For more on linear logic proofs as computer programs, see [55, 1, 14].

For example, if A, B denote types in propositional intuitionistic logic, and A°, B° the corresponding types in linear logic, then $(A \Rightarrow B)^\circ := (!A^\circ) \multimap B^\circ$ where for atoms A we declare $A^\circ = A$. There is a corresponding translation of proofs of $\vdash A$ to proofs of $\vdash A^\circ$. The Church numeral $\underline{2}_A$ is a λ-term of type $(A \to A) \to (A \to A)$ which corresponds to a proof in intuitionistic logic of the sequent $\vdash (A \Rightarrow A) \Rightarrow (A \Rightarrow A)$. If A is atomic, the translation of this type to linear logic is $\mathbf{int}'_A = !(!A \multimap A) \multimap (!A \multimap A)$. Thus a Church numeral in the λ-calculus determines a proof of $\vdash \mathbf{int}'_A$ not $\vdash \mathbf{int}_A$ under this translation. However, this translation is not necessarily the most useful or economical one because of the over-use of exponentials [37, §5.3]. For reasons of clarity we follow standard practice in using the "linear logic version" of the Church numeral $\underline{2}_A$ in Example 4.2 above, rather than the literal translation.

References

[1] S. Abramsky, *Computational interpretations of linear logic*, Theoretical Computer Science, 1993.

[2] S. Abramsky, *Retracting some paths in process algebra*, In CONCUR 96, Springer Lecture Notes in Computer Science **1119**, 1–17, 1996.

[3] S. Abramsky, *Geometry of Interaction and linear combinatory algebras*, Mathematical Structures in Computer Science, **12**, 625–665, 2002.

[4] S. Abramsky and R. Jagadeesan, *New foundations for the Geometry of Interaction*, Information and Computation **111** (1), 53–119, 1994.

[5] M. Anel, A. Joyal, *Sweedler theory of (co)algebras and the bar-cobar constructions*, [arXiv:1309.6952]

[6] M. Atiyah, *Topological quantum field theories*, Publications Mathématique de l'IHÉS 68, 175–186, 1989.

[7] J. Baez and M. Stay, *Physics, topology, logic and computation: a Rosetta stone*, in B. Coecke (ed.) New Structures for Physics, Lecture Notes in Physics 813, Springer, Berlin, 95–174, 2011

[8] M. Barr, *Coalgebras over a commutative ring*, Journal of Algebra 32, 600–610, 1974.

[9] ———, *$\star$-autonomous categories*, Number 752 in Lecture Notes in Mathematics. Springer-Verlag, 1979.

[10] ———, *Accessible categories and models of linear logic*, Journal of Pure and Applied Algebra, 69(3):219–232, 1990.

[11] ______, ?-autonomous categories and linear logic, Mathematical Structures in Computer Science, 1(2):159–178, 1991.

[12] ______, *The Chu construction: history of an idea*, Theory and Applications of Categories, Vol. 17, No. 1, 10–16, 2006.

[13] N. Benton, *A mixed linear and non-linear logic; proofs, terms and models*, in Proceedings of Computer Science Logic 94, vol. 933 of Lecture Notes in Computer Science, Verlag, 1995.

[14] N. Benton, G. Bierman, V. de Paiva and M. Hyland, *Term assignment for intuitionistic linear logic*, Technical report 262, Computer Laboratory, University of Cambridge, 1992.

[15] R. Block, P. Leroux, *Generalized dual coalgebras of algebras, with applications to cofree coalgebras*, J. Pure Appl. Algebra 36, no. 1, 15–21, 1985.

[16] R. Blute, *Hopf algebras and linear logic*, Mathematical Structures in Computer Science, 6(2):189–217, 1996.

[17] R. Blute and P. Scott, *Linear Laüchli semantics*, Annals of Pure and Applied Logic, 77:101–142, 1996.

[18] ______, *Category theory for linear logicians*, Linear Logic in Computer Science 316: 3–65, 2004.

[19] R. Blute, P. Panangaden, R. Seely, *Fock space: a model of linear exponential types*, in: Proc. Ninth Conf. on Mathematical Foundations of Programming Semantics, Lecture Notes in Computer Science, Vol. 802, Springer, Berlin, 1–25, 1994.

[20] R. Bott and L.W. Tu, *Differential forms in Algebraic Topology*, Graduate Texts in Mathematics, **82**, Springer, 1982.

[21] T. Brauner, *Introduction to linear logic*, Basic Research in Computer Science, Lecture Notes (1996).

[22] A. Căldăraru and S. Willerton, *The Mukai pairing, I: a categorical approach*, New York Journal of Mathematics **16**, 61–98, 2010 [arXiv:0707.2052].

[23] N. Carqueville and D. Murfet, *Adjunctions and defects in Landau-Ginzburg models*, [arXiv:1208.1481].

[24] N. Carqueville and I. Runkel, *On the monoidal structure of matrix bifactorisations*, J. Phys. A: Math. Theor. **43** 275–401, 2010 [arXiv:0909.4381].

[25] A. Church, *The Calculi of Lambda-conversion*, Princeton University Press, Princeton, N. J. 1941.

[26] V. Danos and J.-B. Joinet, *Linear logic and elementary time*, Information and Computation 183, 123–127, 2003.

[27] V. Danos and L. Regnier, *Local and Asynchronous beta-reduction (an analysis of Girard's execution formula)* in: Springer Lecture Notes in Computer Science **8**, 296–306, 1993.

[28] V. Danos and L. Regnier, *Proof-nets and the Hilbert space*, in (Girard *et. al.* 1995), 307–328, 1995.

[29] P. J. Denning, *Ubiquity symposium "What is computation?"*: opening statement, Ubiquity 2010. Available on the Ubiquity website.

[30] T. Dyckerhoff and D. Murfet, *Pushing forward matrix factorisations*, Duke Math. J. Volume 162, Number 7 1249–1311, 2013 [arXiv:1102.2957].

[31] T. Ehrhard, *Finiteness spaces*, Math. Structures Comput. Sci. 15 (4) 615–646, 2005.

[32] ———, *On Köthe sequence spaces and linear logic*, Mathematical Structures in Computer Science 12.05, 579–623, 2002.

[33] T. Ehrhard and L. Regnier, *The differential lambda-calculus*, Theoretical Computer Science 309.1: 1–41, 2003.

[34] ———, *Differential interaction nets*, Theoretical Computer Science 364.2: 166–195, 2006.

[35] G. Gentzen, *The Collected Papers of Gerhard Gentzen*, (Ed. M. E. Szabo), Amsterdam, Netherlands: North-Holland, 1969.

[36] E. Getzler, P. Goerss, *A model category structure for differential graded coalgebras*, preprint, 1999.

[37] J.-Y. Girard, *Linear Logic*, Theoretical Computer Science 50 (1), 1–102, 1987.

[38] ———, *Normal functors, power series and the λ-calculus* Annals of Pure and Applied Logic, 37: 129–177, 1988.

[39] ———, *Geometry of Interaction I: Iinterpretation of System F*, in Logic Colloquium '88, ed. R. Ferro, et al. North-Holland, 221–260, 1988.

[40] ———, *Geometry of Interaction II: Deadlock-free Algorithms*, COLOG-88, Springer Lecture Notes in Computer Science **417**, 76–93, 1988.

[41] ———, *Geometry of Interaction III: Accommodating the Additives*, in (Girard *et al.* 1995), pp.1–42.

[42] ———, *Towards a geometry of interaction*, In J. W. Gray and A. Scedrov, editors, Categories in Computer Science and Logic, volume 92 of Contemporary Mathematics, 69–108, AMS, 1989.

[43] ———, *Light linear logic*, Information and Computation 14, 1995.

[44] ———, *Coherent Banach spaces: a continuous denotational semantics*, Theoretical Computer Science, 227: 275–297, 1999.

[45] ———, *The Blind Spot: lectures on logic*, European Mathematical Society, 2011.

[46] J.-Y. Girard, Y. Lafont, and P. Taylor, *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science 7 ,Cambridge University Press, 1989.

[47] G. Gontheir, M. Abadi and J.-J. Lévy, *The geometry of optimal lambda reduction*, in 9th Annual IEEE Symp. on Logic in Computer Science (LICS), 15–26, 1992.

[48] E. Haghverdi and P. Scott, *Geometry of Interaction and the dynamics of prood reduction: a tutorial*, in New Structures for Physics, Lecture notes in Physics **813**, 357–417, 2011.

[49] H. Hazewinkel, *Cofree coalgebras and multivariable recursiveness*, J. Pure Appl. Algebra 183, no. 1–3, 61–103, 2003.

[50] M. Hyland and A. Schalk, *Glueing and orthogonality for models of linear logic*, Theoretical Computer Science, 294: 183–231, 2003.

[51] A. Joyal and R. Street, *The geometry of tensor calculus I*, Advances in Math. **88**, 55–112, 1991.

[52] A. Joyal and R. Street, *The geometry of tensor calculus II*, draft available at http://maths.mq.edu.au/~street/GTCII.pdf

[53] A. Joyal, R. Street and D. Verity, *Traced monoidal categories*, Math. Proc. Camb. Phil. Soc. 119, 447–468, 1996.

[54] M. Khovanov, *Categorifications from planar diagrammatics*, Japanese J. of Mathematics **5**, 153–181, 2010 [arXiv:1008.5084].

[55] Y. Lafont, *The Linear Abstract Machine*, Theoretical Computer Science, 59 (1,2):157–180, 1988.

[56] J. Lambek and P. J. Scott, *Introduction to higher order categorical logic*, Cambridge Studies in Advanced Mathematics, vol. 7, Cambridge University Press, Cambridge, 1986.

[57] A. D. Lauda, *An introduction to diagrammatic algebra and categorified quantum* $\mathfrak{sl}_2$, Bulletin of the Institute of Mathematics Academia Sinica (New Series), Vol. **7**, No. 2, 165–270, 2012 [arXiv:1106.2128].

[58] J. McCarthy, *Recursive functions of symbolic expressions and their computation by machine, Part I.*, Communications of the ACM 3.4: 184–195, 1960.

[59] D. McNamee, *On the mathematical structure of topological defects in Landau-Ginzburg models*, MSc Thesis, Trinity College Dublin, 2009.

[60] P.-A. Melliès, *Functorial boxes in string diagrams*, In Z. Ésik, editor, Computer Science Logic, volume 4207 of Lecture Notes in Computer Science, pages 1–30, Springer Berlin / Heidelberg, 2006.

[61] P-A. Melliès, *Categorical semantics of linear logic*, in : Interactive models of computation and program behaviour, Panoramas et Synthèses 27, Société Mathématique de France, 2009.

[62] P.-A. Melliès, N. Tabareau, C. Tasson, *An explicit formula for the free exponential modality of linear logic*, in: 36th International Colloquium on Automata, Languages and Programming, July 2009, Rhodes, Greece, 2009.

[63] D. Murfet, *Computing with cut systems*, [arXiv:1402.4541].

[64] D. Murfet, *On Sweedler's cofree cocommutative coalgebra*, [arXiv:1406.5749].

[65] M. Pagani and L. Tortora de Falco, *Strong normalization property for second order linear logic*, Theoretical Computer Science 411.2 (2010): 410–444.

[66] A. Polishchuk and A. Vaintrob, *Chern characters and Hirzebruch-Riemann-Roch formula for matrix factorizations*, Duke Mathematical Journal 161.10: 1863–1926, 2012 [arXiv:1002.2116].

[67] U. Schreiber, *Quantization via Linear homotopy types*, [arXiv:1402.7041].

[68] D. Scott, *Data types as lattices*, SIAM Journal of computing, 5:522–587, 1976.

[69] ———, *The Lambda calculus, then and now*, available on YouTube with lecture notes, 2012.

[70] R. Seely, *Linear logic, star-autonomous categories and cofree coalgebras*, Applications of categories in logic and computer science, Contemporary Mathematics, 92, 1989.

[71] P. Selinger, *Lecture notes on the Lambda calculus*, [arXiv:0804.3434].

[72] M. Shirahata, *Geometry of Interaction explained*, available online.

[73] R. I. Soare, *Computability and Incomputability*, in CiE 2007: Computation and Logic in the Real World, LNCS 4497, Springer, 705–715.

[74] M. Sweedler, *Hopf Algebras*, W. A. Benjamin, New York, 1969.

[75] B. Valiron and S. Zdancewic, *Finite vector spaces as model of simply-typed lambda-calculi*, [arXiv:1406.1310].

[76] D. Quillen, *Rational homotopy theory*, The Annals of Mathematics, Second Series, Vol. 90, No. 2, 205–295, 1969.

[77] The Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*, Institute for Advanced Study (Princeton), 2013.

[78] Y. Yoshino, *Cohen-Macaulay modules over Cohen-Macaulay rings*, London Mathematical Society Lecture Note Series, vol. 146, Cambridge University Press, Cambridge, 1990.

[79] E. Witten, *Topological quantum field theory*, *Communications in Mathematical Physics*, 117 (3), 353–386, 1988.

Department of Mathematics, University of Southern California
E-mail address: murfet@usc.edu