Logic and linear algebra: geometry of interaction

Daniel Murfet

July 18, 2015

1 The geometry of interaction

One interesting aspect of linear logic is Girard's program to study the semantics of the cut-elimination process; see [41, §III] and [38, 39, 40]. The purpose of this section is to very briefly explain his idea. As mentioned in the Introduction, the three most common views on what the dynamical process of computation "is" are the execution of a Turing machine, β-reduction in the λ-calculus, and cut-elimination in sequent calculus. At a first approach we notice that common to all three of these models of computation is a tension between the *implicit* and the *explicit*. We use the computation of Section **??** to explain.

Consider the proofs $\underline{\text{mult}}_A(2,-)$ and $\underline{2}_A$ and their cut $\underline{\text{mult}}_A(2,-)\,|\,\underline{2}_A$. The latter is equivalent under cut-elimination to the cut-free proof $\underline{4}$. Since this answer is derived from a deterministic algorithm – cut-elimination – the knowledge is certainly *implicit* in the proof $\underline{\text{mult}}_A(2,-)\,|\,\underline{2}_A$. But some work was necessary to convert this implicit truth into *explicit* truth. Although this example is a trivial one, the reader can easily imagine a similar calculation whose answer is not as apparent as $2 \times 2 = 4$ – or, put aside arithmetic and note that the answer $\underline{4}$ is not obvious from a glance at the diagram (**??**). This *explicitation* is a fundamental aspect of computation, but as the reader can already appreciate, it is difficult to define precisely what we mean by "explicitness". This is the problem that a mathematical model of cut-elimination is designed to solve.

However, this explicitation process is missing from most mathematical models of computation, which assign to the syntactical gadgets of Turing machines, λ-calculus or logic mathematical objects and transformations between them. To elaborate: suppose that in linear logic that we have proofs π of $A \vdash B$ and ρ of $B \vdash C$, and let $\rho\,|\,\pi$ denote the cut of one against the other, as displayed in the following proof tree:

$$\dfrac{\begin{matrix}\pi \\ \vdots \\ A \vdash B\end{matrix} \qquad \begin{matrix}\rho \\ \vdots \\ B \vdash C\end{matrix}}{A \vdash C}\,\text{cut} \tag{1.1}$$

Let $\widetilde{\rho \,|\, \pi}$ denote the cut-free proof of $A \vdash C$ produced from $\rho \,|\, \pi$ by the cut-elimination process. We regard this as the output of the program ρ computed on the input π. In the vector space semantics there is a commutative diagram of linear maps

$$\begin{tikzcd}
 & \llbracket B \rrbracket \arrow[dr, "\llbracket \rho \rrbracket"] & \\
\llbracket A \rrbracket \arrow[ur, "\llbracket \pi \rrbracket"] \arrow[rr, "\llbracket \rho \,|\, \pi \rrbracket = \llbracket \widetilde{\rho \,|\, \pi} \rrbracket"'] & & \llbracket C \rrbracket
\end{tikzcd}$$

On this level, the calculation of output from input is a one-step affair $\llbracket \pi \rrbracket \mapsto \llbracket \rho \rrbracket \circ \llbracket \pi \rrbracket$. This may be contrasted with the syntax, where two steps are involved

$$\pi \longmapsto \rho \,|\, \pi \longmapsto \widetilde{\rho \,|\, \pi} \,. \tag{1.2}$$

The point is that this second step, cut-elimination, is completely invisible in the semantics since both $\rho \,|\, \pi$ and its normalisation have the same denotation – by construction. The explicitation that happens in the syntax is absent in the semantics.

Girard proposed [41] that we should look instead for semantics in which the denotations of $\rho \,|\, \pi$ and $\widetilde{\rho \,|\, \pi}$ are distinct and there are "dynamics" which generate the latter from the former. He refers to the field of study of such dynamics as the *geometry of interaction*.[1] The first example of such a semantics constructed by Girard in [38] has been influential, although arguably it is still a bit mysterious. Since the λ-calculus may be translated into intuitionistic logic, and from there into intuitionistic linear logic, any semantics of linear logic yields a method for the execution of programs in the λ-calculus. One practical application of Girard's geometry of interaction model of linear logic is that it yields a method of executing programs in which the elementary reduction steps are local – that is, they do not dependent on global coordination [26, 27]. This is closely related to famous work of Lamping on optimal reduction in the λ-calculus [45].

1.1 The λ-calculus

At the same time as Turing defined his machines, Church gave another formalisation of the intuitive idea of a computable function in terms of what he called the λ-calculus [24, 68]. Both concepts identify the same class of computable functions $\mathbb{N} \longrightarrow \mathbb{N}$, but the λ-calculus is much more natural from the point of category theory, and it also serves as the theoretical underpinning of programming languages like Lisp [56]. Intuitively, while Turing machines make precise the concept of *logical state* and *state transition*, the λ-calculus captures in a precise mathematical form 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

[1]The categorically minded reader will detect the hint of higher-categories: it would be natural to expect that the denotations of a proof and its cut-free normalisation should be 1-morphisms connected by some structure on the level of 2-morphisms which models cut-elimination.

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

$$(M\ N) \quad \text{or} \quad (\lambda x\,.\,M) \tag{1.3}$$

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

$$T := (\lambda y\,.\,(\lambda x\,.\,(y\,(y\ x))))\,. \tag{1.4}$$

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 and not say any more about it.

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

$$((\lambda x\,.\,M)\ N) \longrightarrow_\beta M[N/x] \tag{1.5}$$

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.[2] This rewrite rule generates an equivalence relation on terms called *β-equivalence* which we will write as $M =_\beta N$ [68, §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 (**??**) 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 (**??**). 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 (possibly numerous) β-reductions are performed until a normal form (the output) is reached.[3]

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

$$(T\ M) = ((\lambda y\,.\,(\lambda x\,.\,(y\,(y\ x))))\ M) \longrightarrow_\beta (\lambda x\,.\,(M\ (M\ x)))\,. \tag{1.6}$$

We see that the result of feeding M to T is itself a program which takes a single input x and returns the resulting of computing $(M\,(M\ x))$. Let's see what happens when we feed this resulting program some other input N

$$((T\ M)\ N) \longrightarrow_\beta ((\lambda x\,.\,(M\ (M\ x)))\ N) \longrightarrow_\beta (M\ (M\ N))\,.$$

[2] 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 [68, §2.3].

[3] 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 [68, §4.2].

This calculation shows how $(T\,M)$ behaves like the "square" of M, in the sense that on an input N it returns M applied twice to N. Obviously we can modify T by nesting more than two function applications, and in this way one defines for each integer $n \geq 0$ a term $\underline{n}$ called the nth *Church numeral* [68, §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.[4]

So far we have considered programs only in the *untyped* λ-calculus. The reader familiar with programming will know that functions in commonly used programming languages are typed: for instance, a function computing the nth Fibonacci number in 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 [68, §6]. In this note we are only interested in the simply-typed λ-calculus, and we therefore use the notation $\underline{2}$ to refer to T_{typed}.

We have now defined the λ-calculus and introduced our basic example of a program, the Church numeral $\underline{2}$. Moreover, we have seen that it is natural think about the "meaning" of programs as *functions* on equivalence classes of terms: for example, if D is the set of β-equivalence classes of terms of type $A \to A$ then there is a function

$$D \longrightarrow D\,, \qquad [M] \mapsto [(T_{typed}\,M)]$$

where $[-]$ denotes β-equivalence classes. This function contains a great deal of information about the term T_{typed}, but this information is not very accessible in light of the fact that we have little handle on the set D. This situation may be compared to the problem of trying to understand a complicated group, and may be approached in the same way: by studying *representations* of programs in the λ-calculus as transformations on mathematical objects. This is referred to as *denotational semantics*, and was initiated as a field by Scott [65] who found a representation of the λ-calculus as continuous maps of certain lattices.

This motivates the following question, which we will keep in mind during the subsequent discussion of linear logic:

Question 1.1. Is there a natural representation of programs in the simply-typed λ-calculus, for example T_{typed}, as linear maps between vector spaces?

[4]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.

Here by a "natural" representation we have in mind something more interesting than linear maps between free vector spaces on sets like D. The obstruction to realising this aim is clear: it would be natural to associate to T_{typed} the map $\alpha \mapsto \alpha^2$ on endomorphisms of some vector space V, but this is not linear. This non-linearity can be traced to the duplication of terms that takes place during β-reduction. To see this, notice how in (**??**) a single occurrence of M on the left becomes a pair of M's on the right, so that during the substitution process the term M is duplicated. This duplication is not linear, at least not in any naive sense, because in general there is no linear map

$$\operatorname{End}_k(V) \longrightarrow \operatorname{End}_k(V) \otimes \operatorname{End}_k(V) \tag{1.7}$$

sending α to $\alpha \otimes \alpha$ for all α, where $\operatorname{End}_k(V)$ denotes the space of linear maps $V \longrightarrow V$. One way to represent T_{typed} as a linear map is to replace $\operatorname{End}_k(V)$ by the universal vector space over it which *is* equipped with a duplication map satisfying some natural axioms; this is the universal cocommutative coalgebra $!\operatorname{End}_k(V)$. Using this coalgebra we may associate to T_{typed} a linear map from which the non-linear map $\alpha \mapsto \alpha^2$ may be recovered; see Example **??** below and Lemma **??**.

But we are getting ahead of ourselves. The first step is to reformulate the λ-calculus in terms of classical logic, then we identify the part of the logic that is responsible for this duplication – called the contraction rule – and explain how the contraction rule is refined in linear logic in such a way as to lead naturally to the coalgebra $!\operatorname{End}_k(V)$.

2 Logic

2.1 Intuitionistic logic

What is logic? Learn it as the study of "what is true" but there is an alternative tradition which sees logic as being about the construction of, and study of the properties of, complex structures objects: namely proofs. Formula as like a boundary condition and proofs as solutions of a DE. Replaces the question of "is there a proof?" analogous to "is there a solution?" to a study of the structure of the set of solutions.

Just like vector spaces of solutions we can try to model logic in such a way that proofs of a formula also form a vector space (or, more honestly, are in some nontrivial way associated with elements of vector spaces built up from the structure of the formula).

Reference introductions to the sequent calculus

This style of formal logic was introduced by Gentzen [34] in order to prove the consistency of Peano arithmetic, and his methods are a fundamental contribution to proof theory.

In propositional intuitionistic logic[5] the connectives are $\vee, \wedge$ and $\Rightarrow$ and examples of

[5]The modifier *propositional* means that we do not include quantifiers $\forall, \exists$ and *intuitionistic* means that sequents are restricted to have only a single formula on the right hand side. This corresponds to the lack of the law of the excluded middle in intuitionistic logic [59, p.14].

formulas include

$$(x \Rightarrow y) \wedge z, \quad ((x \wedge y) \vee z) \Rightarrow y \quad x \vee (y \Rightarrow (z \vee z)), \quad \dots .$$

Some of the deduction rules of the logic are:

$$\frac{}{A \vdash A} \qquad \frac{\Gamma_1 \vdash A \quad \Gamma_2, B \vdash C}{\Gamma_1, \Gamma_2, A \Rightarrow B \vdash C} \Rightarrow L \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \Rightarrow R$$

$$\frac{\Gamma \vdash A \quad A \vdash B}{\Gamma \vdash B}\,\text{cut} \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\,\text{ctr} \ .$$

Here $\Gamma, \Gamma_1, \Gamma_2$ stand for lists of formulas. The format of a deduction rule is that from a proof of each item in the numerator, for instance $\Gamma_1 \vdash A$ and $\Gamma_2, B \vdash C$, we may generate using the deduction rule a proof of the denominator $\Gamma_1, \Gamma_2, A \Rightarrow B \vdash C$. The horizontal bar is labelled with an abbreviation of the name of the deduction rule. The first rule, called the *axiom rule*, has an empty numerator and thus may be introduced at any time; it represents the principle that from A we may obtain A. The second last rule, the *cut rule*, is modus ponens, while in the last rule, the *contraction rule*, we see the principle that if B may be deduced from two copies of A then it may be deduced from one.

An example of a proof of $\vdash (A \Rightarrow A) \Rightarrow (A \Rightarrow A)$ is

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{A \vdash A} \quad \cfrac{\cfrac{}{A \vdash A} \quad \cfrac{}{A \vdash A}}{A, A \Rightarrow A \vdash A} \Rightarrow L}{A, A \Rightarrow A, A \Rightarrow A \vdash A} \Rightarrow L}{A \Rightarrow A, A \Rightarrow A \vdash A \Rightarrow A} \Rightarrow R}{A \Rightarrow A \vdash A \Rightarrow A}\,\text{ctr}}{\vdash (A \Rightarrow A) \Rightarrow (A \Rightarrow A)} \Rightarrow R \qquad (2.1)$$

This depiction is often referred to as a *proof tree*. Obviously this is not the only proof of this sequent, for example we could deduce it in two steps from an axiom rule $A \Rightarrow A \vdash A \Rightarrow A$ followed by the same final step as in (**??**). Usually a sequent has many proofs.

If γ denotes the proof (**??**) taken up to its penultimate step and τ denotes some proof of a sequent $\Gamma \vdash A \Rightarrow A$ then using an instantiation of the cut rule we may generate a new proof of $\Gamma \vdash A \Rightarrow A$ given by the proof tree

$$\cfrac{\begin{matrix} \tau \\ \vdots \\ \Gamma \vdash A \Rightarrow A \end{matrix} \qquad \begin{matrix} \gamma \\ \vdots \\ A \Rightarrow A \vdash A \Rightarrow A \end{matrix}}{\Gamma \vdash A \Rightarrow A}\,\text{cut} \qquad (2.2)$$

This yields up an interpretation of (**??**) as a *function* which maps proofs of $\Gamma \vdash A \Rightarrow A$ to other proofs of $\Gamma \vdash A \Rightarrow A$. At this point we notice the similarity between γ and the program T_{typed} which maps input terms of type $A \to A$ to output terms of type $A \to A$.

This similiarity is an instance of the Curry-Howard isomorphism [68, §6] which matches *types* in the simply-typed λ-calculus (e.g. $A \to A$) with *formulas* in propositional intuitionistic logic (e.g. $A \Rightarrow A$) and *programs* in the λ-calculus with *proofs* in logic. This correspondence pairs (**??**) and γ with T_{typed}. In light of this equivalence, in the proof γ of $A \Rightarrow A \vdash A \Rightarrow A$ it is natural to view the formula on the left hand side of the turnstile as the "input" and the one on the right hand side as the "output". The contraction rule is therefore responsible for the duplication of inputs and, moreover, since the contraction rule may be used at any time on any formula to the left of the turnstile, the duplication of inputs is unrestricted.[6] Although we will not explain the details, by going a little deeper into the content of the Curry-Howard isomorphism one can see how the use of the contraction rule in the penultimate step of (**??**) matches precisely with the duplication of the input term M that takes place in (**??**).

Here we touch on an important point: the feature of logic which corresponds under the Curry-Howard isomorphism to β-reduction in the λ-calculus. The type of a program does not change under β-reduction, so this must be some kind of transformation of proofs of a given sequent. This transformation is called *cut-elimination*. It consists of a collection of transformations on proofs, each of which, when applied to a proof with cuts, generates a new proof of the same sequent with cuts of lower complexity. Gentzen's main theorem in [34] states that his cut-elimination algorithm produces, in a finite number of steps, a proof without any use of the cut rule at all; such a proof is called *cut-free*. Many important properties of logic such as consistency follow immediately from this fact; for more discussion see [59] and [44, Chapter 13]. We will have more to say about cut-elimination in linear logic in Section **??**.

The upshot is that the simply-typed λ-calculus is "the same thing" as propositional intuitionistic logic, and the duplication of terms during substitution in the λ-calculus matches with applications of the contraction rule in logic. Thus, if we want to represent programs as linear maps, it is sufficient to construct a linear representation of *logic*. Moreover, since we have identified duplication of terms as the main obstacle to linearisation of the λ-calculus, a detailed study of the contraction rule appears to be the key. Indeed, this is the insight of Girard with linear logic.

3 GoI

Girard's original geometry of interaction model [38] works as follows: if $\mathbb{H}$ is the Hilbert space of square-summable sequences in $\mathbb{C}$ then he assigns to each proof (in full linear logic) of a sequent $\vdash A_1, \ldots, A_n$ with cuts on formulas $B_1, \ldots, B_m$ a bounded linear operator on $\mathbb{H}^{2m+n}$. Tracing out the $2m$ factors of $\mathbb{H}$ yields an operator on $\mathbb{H}^n$ which agrees with the operator assigned to the cut-free normalisation of the original proof. This operation

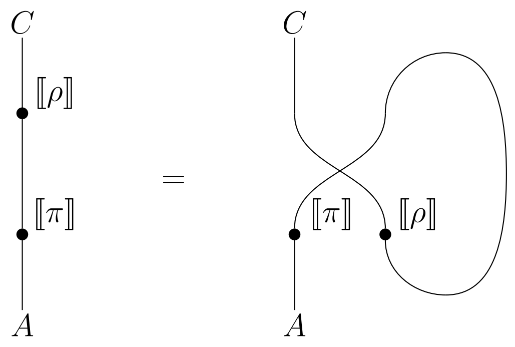
[6] The significance of the contraction rule and duplication as the source of infinity in mathematics is beyond the scope of the present note, but see for instance [41, p.78].

of "tracing out the cuts" is the dynamics which models the process of cut-elimination; it can be rephrased as a sum over paths in a graph associated to the proof [27].

A simplified example using the vector space semantics can capture the key idea. Let us suppose in the above that $[\![B]\!]$ is a finite-dimensional vector space. Then $[\![\rho]\!] \circ [\![\pi]\!]$ may be computed as a trace

$$[\![\rho]\!] \circ [\![\pi]\!] = \operatorname{tr}_{[\![B]\!]} \big(\sigma \circ ([\![\pi]\!] \otimes [\![\rho]\!])\big) \tag{3.1}$$

where $\sigma : [\![B]\!] \otimes [\![C]\!] \longrightarrow [\![C]\!] \otimes [\![B]\!]$ is the symmetry map and $\operatorname{tr}_{[\![B]\!]}$ is a partial trace. Using the standard notation for traces in string diagrams [51] this reads:



The upshot is that we may view the tensor $[\![\pi]\!] \otimes [\![\rho]\!]$ together with the information of the tensor factor $[\![B]\!]$ as an intermediate step between $[\![\pi]\!]$ and the composite $[\![\rho]\!] \circ [\![\pi]\!]$. That is, the two syntactical steps of (1.2) correspond to the sequence

$$[\![\pi]\!] \longmapsto \big([\![\pi]\!] \otimes [\![\rho]\!], [\![B]\!]\big) \longmapsto \operatorname{tr}_{[\![B]\!]} \big(\sigma \circ ([\![\pi]\!] \otimes [\![\rho]\!])\big) = [\![\rho]\!] \circ [\![\pi]\!] \,.$$

Since traces are only available for finite-dimensional spaces this idea cannot be extended to arbitrary $[\![B]\!]$ and is therefore not to be taken seriously, but there are numerous traced monoidal categories where a similar construction can be formalised [4, 2, 3]. In particular, Girard's original model using Hilbert spaces can be formulated in these terms; see the introductory notes by Shirahata [69] and Haghverdi-Scott [46].

Next we consider examples of "explicitation" in geometry, which leads to a discussion of semantics of cut-elimination in bicategories of geometric spaces and the ideas behind [61]. In modern geometry we often understand spaces via the vector bundles living on them, and these we in turn understand via their characteristic classes, which are elements in finite-dimensional vector spaces called cohomology groups. However, *implicit* knowledge of a vector in some cohomology group, say produced from a series of abstract maps, should be distinguished from *explicit* knowledge, say coefficients in a preferred basis.

The prototypical example is Hodge theory. Let M be an n-dimensional compact oriented smooth manifold. The de Rham cohomology of M [20] is the cohomology of the complex of global sections of the sheaf of differential forms

$$\begin{tikzcd} 0 \arrow[r] & \Omega^0(M) \arrow[r, "d^0"] & \cdots \arrow[r] & \Omega^p(M) \arrow[r, "d^p"] & \Omega^{p+1}(M) \arrow[r] & \cdots \arrow[r] & \Omega^n(M) \arrow[r] & 0 \end{tikzcd} .$$

If $Z^p(M) = \mathrm{Ker}(d^p)$ denotes the closed p-forms and $B^p(M) = \mathrm{Im}(d^{p-1})$ the exact p-forms, then the pth de Rham cohomology group is

$$H^p(M) = \frac{Z^p(M)}{B^p(M)} \, .$$

By a theorem of Hodge a preferred basis is given for each p by the subspace of harmonic forms $\mathscr{H}^p(M) \subseteq Z^p(M)$. That is, there is an isomorphism

$$\mathscr{H}^p(M) \longrightarrow Z^p(M) \longrightarrow Z^p(M)/B^p(M) = H^p(M) \, .$$

Giving an equivalence class of vectors in $Z^p(M)$ is therefore equivalent to specifying an element in $\mathscr{H}^p(M)$, but since $Z^p(M)$ is infinite-dimensional these two forms of knowledge have a different character. For example, exhibiting the coefficients of the equivalence class of a vector $\eta \in Z^p(M)$ in a chosen basis $\{\omega_i\}_i$ of harmonic forms requires additional work, which takes the form of computing integrals:

$$\eta \equiv \sum_i \langle \eta, \omega_i \rangle \omega_i, \qquad \langle \eta, \omega_i \rangle = \int_M \eta \wedge \star \omega_i \, . \tag{3.2}$$

To connect this back to computation and the sequence (1.2), let M, N be smooth projective varieties over $\mathbb{C}$. Any cohomology class $\rho \in H^*(M \times N)$ determines a linear map of finite-dimensional vector spaces

$$\begin{aligned} I_\rho : H^*(M) &\longrightarrow H^*(N) \, , \\ \pi &\longmapsto (p_N)_*(\rho \cup p_M^*(\pi)) \end{aligned}$$

where p_M, p_N denote the projections from $M \times N$. Suppose that we insist on dealing only with harmonic forms. Then to compute I_ρ on some "input" harmonic form π and get a harmonic form as "output" we apply the following composite

$$\begin{aligned} \mathscr{H}^*(M) \xrightarrow{\cong} H^*(M) \xrightarrow{I_\rho} H^*(N) \xrightarrow{\cong} \mathscr{H}^*(N) \\ \pi \longmapsto I_\rho(\pi) \longmapsto \sum_j \langle I_\rho(\pi), \omega_j \rangle \omega_j \, . \end{aligned}$$

A comparision of this process – first compute I_ρ and then project onto harmonic forms using the integrals in (1.4) – with the two syntactical steps of (1.2) suggests a natural starting point for a search for semantics of cut-elimination in geometry, with integration playing the role of explicitation.

However, given the expectation previously expressed of higher-categorical semantics of cut-elimination, it is natural to start the search not with cohomology and linear maps but categories and functors. In this setting the relationship between the space of harmonic forms $\mathscr{H}^*(M)$ and the de Rham cohomology $H^*(M)$ may be compared to that between the bounded derived category of coherent sheaves $\mathbf{D}^b(\operatorname{coh} M)$ and the full subcategory $\mathbf{D}^b_{\mathrm{coh}}(\operatorname{Qco} M)$ of complexes with bounded, coherent cohomology in the unbounded derived category of quasi-coherent sheaves $\mathbf{D}(\operatorname{Qco} M)$.[7] These are equivalent categories

$$\mathbf{D}^b(\operatorname{coh} M) \xrightarrow{\cong} \mathbf{D}^b_{\mathrm{coh}}(\operatorname{Qco} M) \subset \mathbf{D}(\operatorname{Qco} M) \tag{3.3}$$

which means that for any object $\mathscr{G}$ of $\mathbf{D}^b_{\mathrm{coh}}(\operatorname{Qco} M)$ there exists an isomorphic bounded complex of coherent sheaves $\mathscr{G}_{\mathrm{finite}}$ which we call a *finite model* of $\mathscr{G}$. But given $\mathscr{G}$ it is not obvious how to produce this finite model in any reasonably algorithmic way.[8] Consider that in the special case where M is a point, (1.5) is an equivalence

$$\mathbf{D}^b(\operatorname{vect} \mathbb{C}) \xrightarrow{\cong} \mathbf{D}^b_{\mathrm{fd}}(\operatorname{Vect} \mathbb{C}) \subset \mathbf{D}(\operatorname{Vect} \mathbb{C})$$

between bounded complexes of finite-dimensional vector spaces and unbounded complexes of (infinite-dimensional) vector spaces with finite-dimensional cohomology. For general M, if $f : M \longrightarrow \mathbb{C}$ denotes the structure map then for a coherent sheaf $\mathscr{F}$ on M the derived pushforward $\mathbb{R}f_*(\mathscr{F})$ may be defined as an object of $\mathbf{D}^b_{\mathrm{fd}}(\operatorname{Vect} \mathbb{C})$ by a simple universal property, but explicitly computing the isomorphic *finite* complex in $\mathbf{D}^b(\operatorname{vect} \mathbb{C})$ – that is, computing the cohomology of $\mathscr{F}$ – requires work to be performed.

To complete the analogy with the above, suppose we are given a complex of coherent sheaves $\mathscr{R} \in \mathbf{D}^b(\operatorname{coh} M \times N)$, which will play the role of ρ. There is a naturally associated triangulated functor

$$I_{\mathscr{R}} : \mathbf{D}^b_{\mathrm{coh}}(\operatorname{Qco} M) \longrightarrow \mathbf{D}^b_{\mathrm{coh}}(\operatorname{Qco} N), \tag{3.4}$$

$$\mathscr{F} \longmapsto (p_N)_*(\mathscr{R} \otimes p_M^*(\mathscr{F})) \tag{3.5}$$

where $p_N : M \times N \longrightarrow N$ and $p_M : M \times N \longrightarrow M$ are the projections and $(p_N)_*, p_M^*$ denote derived functors. Note that the functor $I_{\mathscr{R}}$ naturally produces infinite complexes of quasi-coherent sheaves with bounded coherent cohomology; rarely will the construction output an actual bounded complex of coherent sheaves. If we want to work with finite objects, computing $I_{\mathscr{R}}$ on some input therefore requires two steps

$$\mathbf{D}^b(\operatorname{coh} M) \xrightarrow{\cong} \mathbf{D}^b_{\mathrm{coh}}(\operatorname{Qco} M) \xrightarrow{I_{\mathscr{R}}} \mathbf{D}^b_{\mathrm{coh}}(\operatorname{Qco} N) \xrightarrow{\cong} \mathbf{D}^b(\operatorname{coh} N) \tag{3.6}$$

$$\mathscr{F} \longmapsto I_{\mathscr{R}}(\mathscr{F}) \longmapsto I_{\mathscr{R}}(\mathscr{F})_{\mathrm{finite}}\,. \tag{3.7}$$

[7] We do not intend to compare harmonic forms with arbitrary complexes of coherent sheaves. The point of the analogy is that $\mathscr{H}^*(M)$ provides an explicit finite-dimensional model for de Rham cohomology. The precise nature of the classes that make up this model is of secondary importance.

[8] Unfortunately this is not a very precise statement, since it depends what we mean by "reasonable". While we can reconstruct a complex up to quasi-isomorphism from truncations and in this way construct an explicit complex of coherent sheaves, this is not an algorithm of the kind that we want – consider the enormous complexity of the complexes produced by taking global sections of an explicit resolution of a sheaf $\mathscr{F}$ by injective quasi-coherent sheaves on a non-trivial scheme. In principle the data in this resolution is determined by $\mathscr{F}$, but it would be madness to build a model of computation on top of objects that are so difficult to handle.

That is, given the input complex $\mathscr{F}$ we first compute $I_{\mathscr{R}}$ and then we take a finite model, which is the form of explicitation natural to algebraic geometry.

This suggests that in a model of computation where proofs or programs are interpreted by functors between triangulated categories, cut-elimination should be interpreted as the algorithmic construction of finite models. But what do we mean when we say that some kind of mathematical structure "interprets" cut-elimination? Since categories, functors and natural transformations form a bicategory, our answer is given in terms of bicategorical semantics where types are assigned objects and proofs are assigned 1-morphisms.

Taking Girard's geometry of interaction as our guide, we ask for (at least) the following two desiderata. The algorithm computing finite models should be compatible with

(A) the structure of ambient symmetric monoidal bicategory, and

(B) the comonad interpretating the exponential modality.

The precise form of these requirements will depend on the bicategory under consideration. In the rest of this section we first sketch the kind of bicategorical semantics of linear logic that we have in mind, and then we explain how the cut systems of [61] provide one way of making precise the notion of an "algorithm for computing finite models". In the present discussion the two most relevant examples of bicategories are

- The bicategory $\mathcal{V}ar$ [21] where objects are smooth projective varieties $M, N, \ldots$ and 1-morphisms $M \longrightarrow N$ are bounded complexes of coherent sheaves

  $$\mathcal{V}ar(M,N) = \mathbf{D}^b(\operatorname{coh} M \times N)\,.$$

  For an example of the composition rule, note that if $\mathscr{R} \in \mathcal{V}ar(M,N)$ and S is a point then composition with $\mathscr{R}$ in this bicategory is a functor $\mathcal{V}ar(S,M) \longrightarrow \mathcal{V}ar(S,N)$ which is in fact (1.6).

- The bicategory $\mathcal{LG}$ of Landau-Ginzburg models [57, 22, 23] where objects are isolated hypersurface singularities $W, V, \ldots$ and 1-morphisms $W \longrightarrow V$ are finite-rank matrix factorisations of $V - W$

  $$\mathcal{LG}(W,V) = \operatorname{hmf}(V-W)^{\omega}\,.$$

  Here $\operatorname{hmf}(V-W)$ denotes the homotopy category of matrix factorisations, and $(-)^{\omega}$ the idempotent completion [75].

The bicategories $\mathcal{V}ar$ and $\mathcal{LG}$ are symmetric monoidal but do not have enough colimits to interpret the exponential modality. However, suitable enlargements of both examples yield bicategories $\mathcal{C}$ which are symmetric monoidal with an internal Hom and admit a cofree cocommutative coalgebra $!V$ for each object V. It therefore makes sense to define a semantics of intuitionistic linear logic in $\mathcal{C}$ (or at least, in the classifying category of $\mathcal{C}$)

where the denotation of a proof is a 1-morphism. For example, the interpretation of the Church numeral $\underline{2}$ is a 1-morphism

$$!\operatorname{End}(V) \xrightarrow{\Delta} !\operatorname{End}(V) \otimes !\operatorname{End}(V) \xrightarrow{d \otimes d} \operatorname{End}(V)^{\otimes 2} \xrightarrow{-\circ-} \operatorname{End}(V) \tag{3.8}$$

where $\operatorname{End}(V)$ denotes an object of $\mathcal{C}$ computed using the internal Hom. If we had started with a bicategory of rings and bimodules, this construction would yield the program (**??**) from the introduction.

To explain the interpretation of cut-elimination in such bicategorical semantics we begin by supposing that we have a bicategory $\mathcal{B}$ and for each pair of objects M, N a full subcategory $\mathcal{A}(M, N) \subseteq \mathcal{B}(M, N)$ such that the inclusion is an equivalence. The intuition is that $\mathcal{A}$ is a collection of "finite models" and that every 1-morphism is isomorphic to some finite model. We do not assume that $\mathcal{A}$ is a sub-bicategory: the whole point is that the composition in $\mathcal{B}$ of 1-morphisms in $\mathcal{A}$ will not in general belong to $\mathcal{A}$.

Example 3.1. In the case of $\mathcal{B} = \mathcal{V}ar$,

$$\mathcal{A}(M, N) = \mathbf{D}^b(\operatorname{coh} M \times N), \qquad \mathcal{B}(M, N) = \mathbf{D}^b_{\operatorname{coh}}(\operatorname{Qco} M \times N)\,. \tag{3.9}$$

In the case of $\mathcal{B} = \mathcal{LG}$ both W, V are the defining equations of singularities and

$$\mathcal{A}(W, V) = \operatorname{hmf}(V - W)^\omega, \qquad \mathcal{B}(W, V) = \operatorname{HMF}_{\text{fd}}(V - W) \tag{3.10}$$

where $\operatorname{HMF}_{\text{fd}}(W)$ denotes the homotopy category of infinite-rank matrix factorisations which are direct summands of finite rank ones.

The guiding principle is that the denotation in $\mathcal{B}$ of a cut-free proof should be *finite*, that is, it should be a 1-morphism in $\mathcal{A}$. Denotations of proofs with cuts will be defined using the composition operation in $\mathcal{B}$ and may generate 1-morphisms outside of $\mathcal{A}$. The idea is to look for the shadow of the cut-elimination process in $\mathcal{B}$ in terms of the algorithm which reflects these 1-morphisms back into $\mathcal{A}$. We refine this idea into precise mathematics in a series of steps. For any object S the composition in $\mathcal{B}$ gives a functor

$$\mathcal{A}(S, M) \otimes \mathcal{A}(M, N) \longrightarrow \mathcal{B}(S, N), \tag{3.11}$$
$$\mathscr{F} \otimes \mathscr{R} \mapsto \mathscr{R} \circ \mathscr{F}\,. \tag{3.12}$$

By assumption there is an object $(\mathscr{R} \circ \mathscr{F})_{\text{finite}}$ of $\mathcal{A}(S, N)$ isomorphic to $\mathscr{R} \circ \mathscr{F}$. Adding this step of taking these finite models to the end of (1.13) we have

$$\mathcal{A}(S, M) \otimes \mathcal{A}(M, N) \longrightarrow \mathcal{B}(S, N) \longrightarrow \mathcal{A}(S, N) \tag{3.13}$$
$$\mathscr{F} \otimes \mathscr{R} \longmapsto \mathscr{R} \circ \mathscr{F} \longmapsto (\mathscr{R} \circ \mathscr{F})_{\text{finite}}\,. \tag{3.14}$$

The problem with this construction is that in the examples of interest there is no algorithmic construction of these finite models (of the kind we need) starting from just the data of the object $\mathscr{R} \circ \mathscr{F}$. But what we can do for $\mathcal{LG}$ is refine the composition operation so that it lands not in $\mathcal{B}(S,N)$ but in an auxiliary category, the Clifford thickening of $\mathcal{A}(S,N)$, for which an algorithmic construction of objects in $\mathcal{A}(S,N)$ is under good control.

From now on we assume that $\mathcal{B}$ is a superbicategory [61, §2.1], which is true of $\mathcal{LG}$ but not of $\mathcal{V}ar$. In particular the $\mathcal{A}(M,N)$ are all supercategories. The Clifford thickening of a supercategory is defined using Clifford algebras A_n of dimension 2^{2n} with $A_0 = k$. Each of these algebras has a basic representation, the spinor representation S_n, such that $A_n = \mathrm{End}_k(S_n)$. This means that every representation Q of A_n in the category of vector spaces is of the form $Q \cong S_n \otimes_k V$ for some $\mathbb{Z}_2$-graded vector space V, but the crucial point is that there is some computational "cost" associated with extracting the vector space V from Q. This may be represented either as computing a tensor product with the dual of the spinor representation $S_n^* = \mathrm{Hom}_k(S_n, k)$

$$V \cong S_n^* \otimes_{A_n} Q \tag{3.15}$$

or equivalently as splitting idempotents on Q arising from the Clifford action. When Q is not a vector space but an object of the supercategory $\mathcal{A}(M,N)$ this extraction process is the form of explicitation which models cut-elimination in a cut-system.

To elaborate, we need to use the *Clifford thickening* $\mathcal{A}(M,N)^\bullet$ which is the category of all representations of the various Clifford algebras A_n in $\mathcal{A}(M,N)$. More precisely, an object of this category is object of $\mathcal{A}(M,N)$ together with the action of a Clifford algebra A_n for some $n \geq 0$. There is an equivalence of supercategories

$$\mathcal{A}(M,N) \xrightarrow{\cong} \mathcal{A}(M,N)^\bullet \tag{3.16}$$

which sends an object $\mathscr{F}$ of $\mathcal{A}(M,N)$ to itself viewed as a representation of $A_0 = k$. The point is that the inverse of this equivalence has a very regular form: given an object $(\mathscr{G}, \zeta)$ of $\mathcal{A}(M,N)^\bullet$, which is an object $\mathscr{G}$ together with a morphism of algebras

$$\zeta : A_n \longrightarrow \mathrm{End}^*_{\mathcal{A}(M,N)}(\mathscr{G})$$

for some n, the tensor product

$$\widetilde{\mathscr{G}} := S_n^* \otimes_{A_n} \mathscr{G} \tag{3.17}$$

is an object of $\mathcal{A}(M,N)$ and the map $(\mathscr{G},\zeta) \mapsto \widetilde{\mathscr{G}}$ gives an inverse to (1.18). This inverse is analogous to the extraction of the vector space V from the representation Q. Note that while $\mathscr{G}$ and $\widetilde{\mathscr{G}}$ are isomorphic objects in the Clifford thickening they represent different states of knowledge: in $\mathscr{G}$ the underlying object of $\mathcal{A}(M,N)$ is still implicit, while in $\widetilde{\mathscr{G}}$ it has been made explicit. Let us now assume that there is an algorithm for computing the tensor product (1.19) so that the inverse to (1.18) is under control.[9] Thus it only remains to refine the composition in (1.13) to land in $\mathcal{A}(S,N)^\bullet$.

[9]This algorithm depends on the underlying algebraic objects that make up the bicategory. In the case of $\mathcal{LG}$, computing the tensor product is done by splitting idempotents, which reduces to a computation in commutative algebra using well-understood algorithms.

That is, we ask for a collection of functors, indexed by triples M, N, S

$$\mathcal{A}(S,M) \otimes \mathcal{A}(M,N) \longrightarrow \mathcal{A}(S,N)^{\bullet}\,, \tag{3.18}$$
$$\mathscr{F} \otimes \mathscr{R} \longmapsto \mathscr{R} \,|\, \mathscr{F} \tag{3.19}$$

which satisfy some natural axioms. We call $\mathscr{F} \,|\, \mathscr{R}$ the *cut* of $\mathscr{F}$ and $\mathscr{R}$. It is an object of $\mathcal{A}(S,N)$ together with the action of some Clifford algebra. To say that these cut functors refine the composition operation in $\mathcal{B}$ is to ask that the following diagram commutes up to natural isomorphism:

$$\begin{tikzcd}
\mathcal{B}(S,M) \otimes \mathcal{B}(M,N) \arrow[r, "-\circ-"] & \mathcal{B}(S,N) & \mathcal{A}(S,N) \arrow[l, "\cong"'] \arrow[d, "\cong"] \\
\mathcal{A}(S,M) \otimes \mathcal{A}(M,N) \arrow[u, "\cong"] \arrow[rr, "-\,|\,-"'] & & \mathcal{A}(S,N)^{\bullet}
\end{tikzcd} \quad . \tag{3.20}$$

Commutativity of this diagram allows us to replace the sequence of steps in (1.15), that is, the clockwise direction from the bottom left of the diagram (1.22) to the top right

$$\mathscr{F} \otimes \mathscr{R} \longmapsto \mathscr{R} \circ \mathscr{F} \longmapsto (\mathscr{R} \circ \mathscr{F})_{\text{finite}} \tag{3.21}$$

with the series of steps in the anti-clockwise direction

$$\mathscr{F} \otimes \mathscr{R} \longmapsto \mathscr{R} \,|\, \mathscr{F} \longmapsto \widetilde{\mathscr{R} \,|\, \mathscr{F}} := S_n^* \otimes_{A_n} (\mathscr{R} \,|\, \mathscr{F}) \tag{3.22}$$

The point is that the last step of (1.23) is not under algorithmic control, but every step of (1.24) is algorithmic.

In summary the *cut system* $\mathcal{A}$ which is the collection of categories $\mathcal{A}(M,N)$ and the cut functors (1.20) gives a precise meaning to the idea of an algorithmic construction of finite models for the result of composition of 1-morphisms in $\mathcal{B}$, and this is the structure in the bicategorical semantics which we intend as a model of cut-elimination. The main result of [61] is that in the case $\mathcal{B} = \mathcal{LG}$ there is a natural cut system realising the above. Regarding the desiderata A and B, which must be present before we can call this a model of cut-elimination: while compatibility with the basic structure of the bicategory is part of the axiomatics of a cut system [61, §3], this first paper does not include the compatibility with the tensor product, the internal Hom or finite products. These arguments, together with the verification of desiderata B, will be the subject of future work.

To connect this explicitly back to Girard's ideas in the geometry of interaction, suppose that $\mathcal{B}$ carries an interpretation of linear logic and that we have a pair of proofs π, ρ as in (1.1). Then using the cut system structure we would make the assignments

$$\begin{aligned}
[\![\rho]\!] &= \mathscr{R}, \\
[\![\pi]\!] &= \mathscr{F}, \\
[\![\rho \,|\, \pi]\!] &= \mathscr{R} \,|\, \mathscr{F}, \\
[\![\widetilde{\rho \,|\, \pi}]\!] &= \widetilde{\mathscr{R} \,|\, \mathscr{F}}\,.
\end{aligned}$$

The "dynamics" which generates $\widetilde{\mathscr{R}\,|\,\mathscr{F}}$ from $\mathscr{R}\,|\,\mathscr{F}$ is the computation of the tensor product with the dual of the spinor representation S_n^* [61, §3.1].

Example 3.2. The polynomial 0 is an object of $\mathcal{B} = \mathcal{LG}$ and with $\mathcal{A}$ as in (1.12) for any hypersurface singularity W, we have

$$\begin{aligned} \mathcal{A}(0,W) &= \operatorname{hmf}(W)^\omega, & \mathcal{A}(W,0) &= \operatorname{hmf}(-W)^\omega, \\ \mathcal{A}(0,0) &= \mathcal{C}_2(k), & \mathcal{B}(0,0) &= \mathcal{C}_2^\infty(k)\,. \end{aligned}$$

where $\mathcal{C}_2(k)$ is the homotopy category of $\mathbb{Z}_2$-graded complexes of finite-dimensional k-vector spaces and $\mathcal{C}_2^\infty(k)$ is the homotopy category of infinite-rank complexes with finite-dimensional cohomology. Moreover $\mathcal{A}(0,0)^\bullet$ is the category of A_n-representations in $\mathcal{C}_2(k)$ for various n. In this case the diagram (1.22) becomes

$$\begin{tikzcd} \mathcal{C}_2^\infty(k) & & \mathcal{C}_2(k) \arrow[ll, "\cong"'] \arrow[d, "\cong"] \\ \operatorname{hmf}(W)^\omega \otimes \operatorname{hmf}(-W)^\omega \arrow[u, "-\otimes-"] \arrow[rr, "-|-"'] & & \mathcal{C}_2(k)^\bullet \end{tikzcd} \tag{3.23}$$

The vertical functor on the left, which comes from the composition in $\mathcal{B}$, is simply a tensor product. Every matrix factorisation of $-W$ is the dual $Y^\vee$ of some matrix factorisation Y of W, and the functor sends $X \otimes Y^\vee$ to $\operatorname{Hom}(Y,X) \in \mathcal{C}_2^\infty(k)$. The clockwise direction around this square is the sequence (1.23)

$$X \otimes Y^\vee \longmapsto \operatorname{Hom}(Y,X) \longmapsto H^* \operatorname{Hom}(Y,X)$$

whereas the anti-clockwise direction is (1.24)

$$X \otimes Y^\vee \longmapsto Y^\vee\,|\,X \longmapsto S_n^* \otimes_{A_n} (Y^\vee\,|\,X)\,.$$

The cut $Y^\vee\,|\,X$ is a representation of some Clifford algebra A_n in $\mathbb{Z}_2$-graded complexes of finite-dimensional vector spaces, that is, there are closed operators $a_1,\dots,a_n,a_1^\dagger,\dots,a_n^\dagger$ on $Y^\vee\,|\,X$ satisfying the anti-commutation relations

$$a_i a_j = a_j a_i, \qquad a_i^\dagger a_j^\dagger = a_j^\dagger a_i^\dagger, \qquad a_i a_j^\dagger = \delta_{ij} + a_j^\dagger a_i\,.$$

An example with explicit matrices for these operators is given in [61, Example 4.15]. Since A_n is Morita trivial we know that this representation must be of the form $S_n \otimes_k V$ for some complex V, and the point of the commutativity of (1.25) is that this V is precisely the cohomology $H^* \operatorname{Hom}(Y,X)$. One can show that V is the image of the following idempotent on the complex $Y^\vee\,|\,X$ constructed from the Clifford action [61, Definition 2.18]

$$e_n := a_1^\dagger \cdots a_n^\dagger a_n \cdots a_1\,.$$

The content of the cut system in this special case is that splitting this idempotent provides a more controlled way of extracting the vector space $H^* \operatorname{Hom}(Y,X)$ from the data of X,Y than taking cohomology. In particular, the cut system links invariants of the cohomology (such as the Euler characteristic) to invariants of X and Y (such as the Chern characters) in a way that is obscured by taking cohomology; see below.

This concludes our sketch of the links between cut-elimination and geometry. While Girard's original geometry of interaction model immediately yielded insight into the execution of programs in the λ-calculus, it is not clear what geometric models have to say about practical questions in computer science. But it is intriguing that in the example of $\mathcal{LG}$ the Clifford actions, which we may view as algorithmic structure "hidden inside" the composition operation, are both the natural interpretation of cut-elimination and also encode fundamental geometric properties of matrix factorisations such as the analogue of Hirzebruch-Riemann-Roch theorem. Roughly, this arises as follows: in the situation of Example 1.2 the cut system provides a diagram of morphisms of complexes

$$\begin{tikzcd} H^* \operatorname{Hom}(Y,X) \arrow[r, shift left, "g"] & Y^\vee \,|\, X \arrow[l, shift left, "f"] \end{tikzcd}$$

with $f \circ g = 1$ and $g \circ f = e_n$. Then

$$\begin{aligned} \chi\big(H^* \operatorname{Hom}(Y,X)\big) &= \operatorname{str}(1_{H^* \operatorname{Hom}(Y,X)}) \\ &= \operatorname{str}(f \circ g) \\ &= \operatorname{str}(g \circ f) \\ &= \operatorname{str}(a_1^\dagger \cdots a_n^\dagger a_n \cdots a_1)\,. \end{aligned}$$

At this point the fact that the operators $a_i^\dagger$ are Atiyah classes allows us to transform this supertrace into a residue, which presents the Euler characteristic as an integral involving the Chern characters of X and Y. The details are given in [29], see also [61, Remark 4.13]. This recovers the Hirzebruch-Riemann-Roch formula for matrix factorisations proven by Polishchuk-Vaintrob [63] and hints at a link between cut-elimination in logic and index theorems in geometry, which may ultimately help to shed light on fundamental questions about the tension between local and global aspects of computation.[10]

An important recent topic in proof theory is homotopy type theory [74] where spaces in the sense of homotopy theory are assigned to types in Martin-Löf type theory. Since in the bicategorical semantics above geometric spaces of one kind or another are assigned to types in linear logic, there is an obvious connection. Indeed, Schreiber [64] has discussed linear logic and algebraic geometry in the context of homotopy type theory and quantum field

[10]What does an algebraist stand to learn from bicategorical models of logic? The interaction with logic can lead to ideas that might otherwise go unexplored – for instance, from the logical perspective it is natural to take an isolated hypersurface singularity W and form the universal cocommutative coalgebra $!W$ mapping to W, but otherwise the importance of this construction would not be obvious. To take this particular example a bit further: in *implicit computational complexity* restrictions on the use of the exponential modality are used to characterise complexity classes of programs, such as polynomial time [42] and elementary time [25]. In the vector space semantics of this paper, where proofs or programs are interpreted as morphisms, this provides some notion of complexity for linear maps. But in a bicategorical semantics where proofs are interpreted as objects in cocomplete triangulated categories of 1-morphisms one should study the complexity of programs constructing objects from a set of fixed objects.

theory. However it is not obvious to the author how to make the connection precise, since our main emphasis is on explicit models of cut-elimination on the level of 2-morphisms which does not appear in [64]. One observation that seems relevant is that the key idea in the construction of a cut system in [61] is an application of the homological perturbation lemma to compute the convolution of kernels in tractable terms, and from this the Clifford actions emerge. It would be not be surprising if this sort of structure appeared naturally in the setting of homotopy type theory and, more to the point, maybe the theory can be used to define something like a cut system for $\mathcal{V}ar$. But this is probably not easy!

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

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

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

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

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

[26] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[41] ———, *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.

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

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

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

[45] 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.

[46] 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.

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

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

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

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

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

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

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

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

[55] 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].

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

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

[58] 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.

[59] 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.

[60] 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.

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

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

[63] 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].

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

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

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

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

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

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

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

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

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

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

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

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

[76] 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