Zeta Functions and the (Linear) Logic of Markov Processes

Thomas Seiller

Zeta Functions and the (Linear) Logic of Markov Processes

Thomas Seiller

CNRS, LIPN – UMR 7030 Université Sorbonne Paris Nord
99, Avenue Jean-Baptiste Clément
93430, Villetaneuse, FRANCE
Email: seiller@lipn.fr

***Abstract*—In a series of papers, Seiller introduced models of linear logic known as "Interaction Graphs". These models generalise Girard's various geometry of interaction constructions, providing a unifying framework for those. In this work, we exhibit how these models can be understood mathematically through a cocycle property satisfied by zeta functions of dynamical systems. Focussing on probabilistic models, we then explain how the notion of graphings used in the models captures a natural class of Markov processes. We further extend previous constructions to provide a model of second-order linear logic as a type system over the set of all (discrete-time) sub-Markov processes.**

Denotational semantics were introduced by Scott [1] as a mathematical theory of computation. This so-called theory of domains provided many insights and applications, as well as open questions, such as the question of *full abstraction*. It is in fact a categorification of the notion of domains[1] [4] that lead Girard to the discovery of linear logic [5]. However, right after his seminal paper introducing Linear Logic [5], Girard proposed a research programme [6] aimed at developing a new semantic approach, which we call here "dynamic semantics".

The motivations for this programme was adequately modelling the dynamics of proof normalisation (cut-elimination), which through the Curry-Howard correspondence amounts to adequately modelling the dynamics of program execution (in lambda-calculus). Indeed, whereas denotational semantics identifies both a proof π – or equivalently a program P applied to an input n – and its normal form ρ – equivalently, the result of the computation $P(n)$, Girard's aim was to model both π and ρ as different objects, together with a mathematical operation – called execution – computing ρ from π. As such, a model satisfying Girard's expectations would correctly model not only programs and data, but also the dynamic process of program execution.

Girard's programme inspired the introduction of game semantics which were used to solve the long-standing problem of *full abstraction* [7], [8]. However, Girard's approach differs from game semantics in that the model is inherently untyped. Indeed, game semantics consider *strategies* which subsume a definition of type, and hence can easily be designed to fit the model of computation under study. As such, they follow a types-as-constraints methodology, where types are used to *restrict* the behaviour of programs and therefore of the strategies considered in the model. On the other hand, Girard's methodology is that of types-as-descriptions philosophy, where the model under study is untyped, and types are constructed through realisability techniques and *describe* the behaviour of programs. As a fundamental example, we consider the reconstruction of simple types by realisability on the pure λ-calculus described by Riba [9]. This can be compared with game semantics models of Church-style lambda-calculus in which many pure λ-terms cannot be written, and thus represented: while the latter follows the classic game semantics approach, the former follows Girard's methodology.

The models defined by Girard were however concerned with less standard underlying models of computation. This is explained by his aim to model *linear logic*, a refinement of intuitionnistic logic: as a consequence one seeks models providing more precise mechanisms than the λ-calculus. The first such models [10], [11], [12], [13] were constructed around an abstraction of programs as bounded operators on Hilbert spaces. However, recent work by Seiller reformulated and generalised the constructions by considering graphs or graph-like structures called *graphings* to account for programs [14], [15], [16], [17]. We will now describe the basic intuitions behind the construction of models in the simple case of graphs, before explaining the need for generalising the approach to graphings.

A. Interaction Graphs

Graphs provide a minimal but natural mathematical structure to represent programs. Indeed, Turing machines and automata can naturally be abstracted as finite graphs. In these models, computation is represented as the computation of paths in the graph: in the case of Turing machines, the graph represents the transition function, and the process of computation corresponds to the iteration of this transition function, i.e. following a path to travel through the graph. The basic notions of the models are thus that of graphs, and that of the *execution* $\mathrm{Ex}(G)$ of a graph G which is defined as some set of maximal paths in G. This alone describes some kind of abstract, untyped, model of computation, which one can structure by defining types depending on how graphs behave.

[1] Which turned out to be a quantitative version of Berry's dI-domains [2], see also the appendix of [3] for the connexion with coherence spaces.

Types are defined as sets of graphs (satisfying some properties), i.e. a type $\mathbb{A}$ is understood as the set of all programs that can be given the type $\mathbb{A}$. The notion of *execution*, which abstractly represents the execution of a program given some input, is the key ingredient to the construction of (linear) implication, i.e. arrow types. Indeed, supposing $\mathbb{A}, \mathbb{B}$ are defined types, then a graph G will have type $\mathbb{A} \multimap \mathbb{B}$ (the linear implication) if and only if for all graph of type A, the graph G applied to A – noted $G\Box A$ – reduces to a graph $\mathrm{Ex}(G\Box A)$ of type $\mathbb{B}$. Let us note that this formalism is extremely expressive; for instance it naturally interprets polymorphism (a graph belongs to many sets of graphs, thus many types), subtyping (the inclusions of sets of graphs), and quantifiers (defined through unions and intersections of sets of graphs).

However, the setting of finite graphs is not rich enough to adequately model expressive computational models such as Turing machines or lambda-calculus. Some information is lost by considering discrete graphs: following an edge corresponds to performing some instruction and therefore modifying the state of the machine; restricting to finite structures in some sense limits the approach to models of computation with finite sets of states. To regain expressivity, Seiller introduced graphings [16]. Graphings are graphs which are realised on a topological or measured space which represent the space of all possible configurations of the machine. Edges are interpreted as specific endomorphisms of this space restricted to some subspace, representing the action of instructions on a chosen subset of configurations.

As an example, for Turing machines one may consider the space of configurations $\mathbf{X} = \{*, 0, 1\}^{|\mathbb{Z}|}$ of $\mathbf{Z}$-indexed sequences of symbols $*, 0, 1$ that are almost always equal to $*$. Moving the working head to the right can be represented as the map $\texttt{right} : \mathbf{X} \to \mathbf{X}, (a_i)_{i\in\mathbf{Z}} \mapsto (a_{i+1})_{i\in\mathbf{Z}}$. The instruction "if the head is reading a 0 or a 1, move to the right" is then represented as an edge of source the subspace $\{(a_i)_{i\in\mathbf{Z}} \in \mathbf{X} \mid a_0 \neq *\}$ and realised by the map $\texttt{right}$.

B. Contributions

The main and more saliant contribution of this work is the definition of a model of second-order linear logic (LL^2) from realisability techniques applied to general sub-Markov kernels. This is, to the author's knowledge, the first model of LL^2 able to accommodate discrete and continuous probability distributions. As a consequence, we expect the construction will naturally provide models of typed lambda-calculus extended with probabilities and specific instructions for sampling discrete and non-discrete distributions.

The obtention of this result goes through several steps, each of which consists in a separate contribution.

- Firstly, we relate Seiller's discrete IG models based on graphs with the work of Ihara on zeta functions of graphs. We explain how the models, and particularly the fact that the notion of type inferred follows the linear logic discipline, then essentially relies on a cocycle relation relating the notion of execution and the zeta functions.
- Secondly, we show that the restriction of Seiller's models [16], [17] to *deterministic* graphings boils down to the representation of programs as partial dynamical systems. In this case, execution is more or less described as the iteration of the considered map, and – extending the discrete case – types are constructed based on some notion of *zeta function* for dynamical systems related to Ruelle's zeta functions of dynamical systems. This exhibits Seiller's models as realisability models on the set of partial dynamical systems.
- Thirdly, if one allows to weight edges with probabilities, then the natural notion of *probabilistic graphings* is shown to coincide with a class of sub-Markov processes that we will describe. This hints at a possible generalisation of the realisability construction on the set of all sub-Markov kernels. We therefore introduce the notions of execution and zeta functions for general sub-Markov kernels and prove they satisfy the cocycle relation.

I. Interaction Graphs: Linear Logic and Zeta functions

In this section, we review the models of linear logic introduced by Seiller under the name "Interaction Graphs". Among these models, the first instances were build around the notion of graph, while the latter used the more general notion of graphing. We first review the former setting to ease the reader in understanding the basic concepts we are pinpointing through a new presentation – stressing in particular the use of zeta functions –, before detailing in the next section how those adapt in the case of graphings.

A. Interaction Graphs: the discrete case in a nutshell

We briefly recall the basics of IG model in the discrete case. We work with weighted directed (multi-)graphs; here we will suppose weights are complex numbers $\mathbf{C}$. Graphs are defined as tuples $G = (V^G, E^G, s^G, t^G, \omega^G)$, where V^G and E^G are sets, s^G and t^G are respectively the source and target maps from E^G to V^G, and $\omega^G : E^G \to \mathbf{C}$ is a weight map.

The basic operation is that of *execution* between two graphs F, G. This interprets program execution (explaining the naming convention) through cut-elimination. The cut is implicitly represented as the common vertices of the two graphs F, G. This eases the expressions, and is equivalent to the more traditional approach where one would consider both F and G, together with a graph representing the cut rule (cf. Figure 1). As execution is defined through alternating paths, the results are equivalent and we urge the reader to use whatever convention she finds more natural.

We start to fix a few notations that will be used in this paper.

Notations 1. Given two sets A, B, we write $A\backslash B$ the set $\{a \in A \mid a \notin B\}$, and $A\triangle B$ their symmetric difference $(A\backslash B) \cup (B\backslash A) = (A \cup B)\backslash(A \cap B)$.

Given two graphs G, H, we write $G \cup H$ the graph $(V^G \cup V^H, E^G \uplus E^H, s^G \uplus s^H, t^G \uplus t^H, \omega^G \uplus \omega^H)$. Note the non-disjoint union of sets of vertices, which is essential to consider alternating paths between the two graphs.

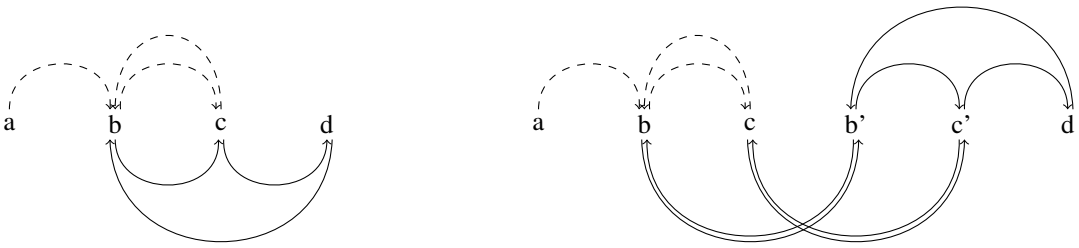


Fig. 1. On the left: implicit cut between two graphs (one is plain, the other is dashed). On the right: explicit cut between the same two graphs (the cut is shown below). Both representations lead to the same result, as the cut-elimination is represented by *execution*, an operation defined from alternating paths. It is easily checked that there is a bijective correspondence between alternating paths on the left and alternating paths on the right.

Definition 2 (Alternating paths)**.** Let G and H be two graphs. An *alternating path* π of length $|\pi| = k$ between G and H is a path (e_i) in $G \cup H$ which satisfy that for all $i = 0, \ldots, k-1$, $e_i \in E^F$ if and only if $e_{i+1} \in E^G$. The source and target of the path are respectively defined as $s^{G\cup H}(\pi) = s^{G\cup H}(e_0)$ and $t^{G\cup H}(\pi) = t^{G\cup H}(e_{k-1})$.

The set of alternating paths will be denoted by $\mathrm{Path}(G, H)$, while $\mathrm{Path}(G, H)_V$ will mean the subset of alternating paths between G and H with source and target in a given (sub)set of vertices V.

Definition 3. Let F and G be two graphs. The *execution* of F and G is the graph $F :: G$ defined by:

$$\begin{aligned} V^{F::G} &= V^F \triangle V^G \\ E^{F::G} &= \mathrm{Path}(F, G)_{V^F \triangle V^G} \\ s^{F::G} &= \pi \mapsto s^{G\cup H}(\pi) \\ t^{F::G} &= \pi \mapsto t^{G\cup H}(\pi) \\ \omega^{F::G} &= \pi = \{e_i\}_{i=0}^n \mapsto \prod_{i=0}^n \omega^{G\square H}(e_i) \end{aligned}$$

When $V^F \cap V^G = \emptyset$, we write $F \cup G$ instead of $F :: G$.

This notion of execution can be related to cut-elimination in proof nets, and it represents the execution of programs through the Curry-Howard correspondence. We will now define the notion of orthogonality which can be related to corrected criterions for proof nets, and is used to define types by means of *testing*. We refer the interested reader to work by Naibo, Petrolo and Seiller [18] for more details and explanations. Defining orthogonality in IG models is is done by quantifying *closed paths* and *prime closed paths*.

Definition 4. Given a graph G, a closed path π (called *circuit* in earlier work by Seiller [14]) of length $|\pi| = k$ is a path $(e_i)_{i=0}^{k-1}$ such that $s^G(e_0) = t^G(e_{k-1})$ and considered up to cyclic permutations. A *prime closed path* (called 1-circuit in IG) is a closed path which is not a proper power of a smaller closed path. We denote by $\mathcal{C}(G)$ the set of prime closed paths in G.

Definition 5. Given graphs F, G, an *alternating closed path* π of length $|\pi| = 2k$ is a closed path $(e_i)_{0\leqslant i\leqslant 2k-1}$ in $F \cup G$ such that for all $i \in \mathbf{Z}/2k\mathbf{Z}$, $e_i \in F$ if and only if $e_{i+1} \in G$. The set of prime alternating closed paths between F and G will be denoted $\mathcal{C}(F, G)$.

This notion is used in previous Interaction Graphs (IG) models to define a *measurement* which in turn defines the orthogonality relation. The orthogonality, in turn, is used to define types in a way reminiscent of classical realisability [19], and related to Hyland and Schalk's double glueing construction [20]. We only recall the measurement here and refer to the first IG paper for more details [14]. The notion of measurement depends on a map that is used to associate to each cycle a positive real number depending on its weight.

Definition 6. Let m be a map $\mathbf{C} \to \mathbf{R}_{\geqslant 0}$. For any two graphs F, G we define the measurement

$$[\![F, G]\!]_m = \sum_{\pi\in\mathcal{C}(F,G)} m(\omega^{F\square G}(\pi)).$$

Based on these two ingredients (execution and measurement), and two essential properties, namely the associativity of execution [14] and the *trefoil property* [15], one can define a myriad of models of Multiplicative Linear Logic (MLL) and Multiplicative-Additive Linear Logic (MALL). These models capture the different models introduced by Girard by choosing carefully the map m used to define the measurement. We will now explain how there is a similarity between the measurement defined by Seiller and just recalled, and the Bowen-Lanford zeta function of graphs. To formalise the connection, we need to consider zeta functions of weighted graphs, but we will start with a quick overview of the theory of zeta functions of (non-weighted) graphs.

B. Bowen-Lanford Zeta Functions

We first recall the definition and some properties of the zeta function of a directed graph. We refer to the book of Terras [21] for more details. We will later on continue with zeta functions for weighted directed graphs, and further with zeta functions for dynamical systems. The graph case is important as it provides intuitions about the later generalisations.

In this subsection only, we consider a directed graph as a tuple $G = (V^G, E^G, s^G, t^G)$ (i.e. without weights) and suppose it is *simple*, i.e. that the map $E^G \mapsto V^G \times V^G; e \mapsto$

$(s^G(e), t^G(e))$ is injective. Given such a graph, its *transition matrix* is defined as the $V^G \times V^G$ matrix whose coeficients are defined by $M_G(v, v') = 1$ if there is an edge $e \in E^G$ such that $s^G(e) = v$ and $t^G(e) = v'$, and $M_G(v, v') = 0$ otherwise. The following definition provides a clear parallel with the famous Euler zeta function.

Definition 7. The Bowen-Lanford zeta function associated with the graph G is defined as:

$$\zeta_G(z) = \prod_{\tau \in \mathcal{C}(G)} (1 - z^{|\tau|})^{-1}$$

which converges provided $|z|$ is sufficiently small.

The two following lemmas are easy to establish (using the identity $\log(1-x) = \sum_{k=1}^{\infty} \frac{x^n}{n}$). The first lemma is essential, as it is the alternative expression of the zeta function that we will be able to generalise later. Indeed, while the formal definition above uses the notion of prime closed paths, this one quantifies over all closed paths.

The second lemma is key to the representation of $\zeta_G(z)$ as a rational function. This relates the zeta function with the determinant of the adjacency matrix of G. Notice that this relation was obtained by Seiller in the special case $z = 1$ [14] and was the initial motivation behind the definition of orthogonality in Interaction Graphs models, as it relates the measurement with the Fuglede-Kadison determinant of operators [22] used in Girard's model [13].

Lemma 8. *Let $N(n)$ denote the number of all possible strings $(v_1, \ldots, v_n)$ representing a closed path in G of length n. Then:*

$$\zeta_G(z) = \exp\left(\sum_{i=1}^{\infty} \frac{z^n}{n} N(n)\right)$$

Lemma 9. *Let G be a graph, $M(G)$ its* transition matrix*:*

$$\mathrm{tr}(M(G)^k) = N(k)$$

Together, these two lemmas yield the following result.

Proposition 10. *Let G be a graph, $M(G)$ its* transition matrix*:*

$$\log(\zeta_G(z)) = -\log(\det(1 - z.M(G))),$$

for sufficiently small values of $|z|$.

Proof. From the following computation:

$$\begin{aligned}
\log(\zeta_G(z)) &= \sum_{i=1}^{\infty} \frac{z^n}{n} N(n) \\
&= \sum_{i=1}^{\infty} \frac{z^n}{n} \mathrm{tr}(M(G)^k) \\
&= \sum_{i=1}^{\infty} \frac{(z.\mathrm{tr}(M(G))^n}{n} \\
&= -\log(\det(1 - z.M(G))),
\end{aligned}$$

where the last equality can be found with a (simple) proof in Seiller's earlier work [14, Lemma 61]. □

As we will show later on, the zeta function of graphs is strongly related to the orthogonality in IG models, as the measurement used in these models boils down to computing the value of some graph zeta function at $z = 1$. In fact, we will show how to define new models by simply considering the zeta function itself instead of its value at 1. But for this we need to define the zeta function of weighted graphs.

C. Zeta functions of weighted directed graphs

Now, we consider weighted directed graphs, i.e. graphs with weights of the edges, and we will restrict to the case of complex numbers as weights. We write ω the weight function, as well as its extension to paths, using the product, i.e.

$$\omega(\pi) = \prod_{e \in \pi} \omega(e).$$

We recall that a (weighted) *simple graph* is a graph $G = (V^G, E^G, s^G, t^G, \omega^G)$ with at most one edge between two given vertices, i.e. the pairing $\langle s^G, t^G \rangle : E^G \to V^G$ is injective. Given a simple weighted graph G, we define its *transition matrix* as the $V^G \times V^G$ matrix with $M_G(v, v') = \omega(e)$ if there exists a (necessarily unique) edge $e \in E^G$ with $\langle s^G(e), t^G(e) \rangle = (v, v')$, and $M_G(v, v') = 0$ otherwise.

Now, given a graph (not necessarily simple) G, we write $G(v, v')$ the set $\{e \in E^G \mid s^G(e) = v, t^G(e) = v'\}$. One can extend the definition of *transition matrix* by associating to G the $V^G \times V^G$ matrix with $M_G(v, v') = \sum_{e \in G(v,v')} \omega(e)$. Alternatively, this matrix can also be defined as $M_{\hat{G}}$ where $\hat{G}$ is the *simple collapse* of G, i.e. the simple graph defined as $\hat{G} = (V^G, \hat{E}^G, \hat{s}^G, \hat{t}^G, \hat{\omega}^G)$ with:

- $\hat{E}^G = \{(v, v') \in V^G \times V^G \mid G(v, v') \neq \emptyset\}$,
- $\hat{s}^G((v, v')) = v$,
- $\hat{t}^G((v, v')) = v'$,
- $\hat{\omega}^G((v, v')) = \sum_{e \in G(v,v')} \omega(e)$.

Let us notice that Seiller proved in earlier work that the measurement defined from the function $m := \lambda x. -\log(1-x)$ satisfies $[\![F, G]\!]_m = [\![\hat{F}, \hat{G}]\!]_m$.

Now, the zeta function of the weighted graph is defined as follows. Note that we take the product of the weights to define the weight ω of a path, while standard work on zeta functions for weighted graphs define the weight ν of a path as a sum; this is formally explained by taking a logarithm, i.e. $\omega = \log \circ \nu$, explaining why we consider the product of expressions $1 - \omega(\pi)z$ while the latter is defined by taking the product of $1 - z^{\nu(\pi)}$.

Definition 11. The zeta function associated with the weighted graph G is defined as:

$$\zeta_G(z) = \prod_{\pi \in \mathcal{C}(G)} (1 - \omega(\pi).z)^{-1}$$

which converges provided $|z|$ is sufficiently small.

Following the same reasoning as in the non-weighted case, one obtains the following general result, which extends Seiller's combinatorial interpretation of the determinant $\det(1 - M(G))$ [14, Corollary 61.1].

Proposition 12. *Let G be a directed weighted graph, $M(G)$ its* transition matrix*:*

$$\log(\zeta_G(z)) = -\log(\det(1 - z.M(G))),$$

for sufficiently small values of $|z|$.

Taking the logarithm we obtain:

$$\log(\zeta_G(z)) = \sum_{\pi \in \mathcal{C}(G)} -\log(1 - \omega(\pi).z)),$$

an expression that appears in the definition of measurement in the previous section. This can be used to relate the measurement defined in interaction graphs for $m := \lambda x. \log(1 - x)$ with the value of the zeta function at $z = 1$:

$$[\![F, G]\!]_m = \log(\zeta_{F \bullet G}(1))$$

where the $\bullet$ operation consists in composing (i.e. taking length-2 paths) the graphs $F + \mathbf{1}_{V^F \setminus V^G}$ and $G + \mathbf{1}_{V^G \setminus V^F}$.

Now, we recall that orthogonality in IG models is defined by $F \perp G$ as $[\![F, G]\!]_m \neq 0, \infty$, i.e. if and only if $-\log(\zeta_{F \bullet G}(1)) \neq 0, \infty$, i.e. if and only if $\zeta_{F \bullet G}(1) \neq 0, 1$.

We will show in the next section how this remark can be used to extend the construction of IG models to provide new models by using zeta functions to define the orthogonality.

D. Zeta, Execution and a Cocycle Property

There are two main properties used to define IG models [15], [23]. The first is the associativity of execution, i.e. that $F :: (G :: H) = (F :: G) :: H$ under some mild hypothesis on the graphs (i.e. that $V^F \cap V^G \cap V^H = \emptyset$). The second main property is the so-called *trefoil property* [15]:

$$[\![F, G :: H]\!]_m + [\![G, H]\!]_m = [\![G, H :: F]\!]_m + [\![H, F]\!]_m. \quad (1)$$

The identity is in fact obtained from the so-called geometric trefoil property [23], [15], satisfied for graphs F, G, H such that $V^F \cap V^G \cap V^H = \emptyset$:

$$\mathcal{C}(F, G :: H) \uplus \mathcal{C}(G, H) \equiv \mathcal{C}(G, H :: F) \uplus \mathcal{C}(H, F), \quad (2)$$

where $\equiv$ denotes a weight-preserving bijection.

The trefoil property can now be rephrased as a special case of a general cocycle condition satisfied by zeta functions. Indeed, remembering that $[\![F, G]\!]_{\lambda x. \log(1-x)} = -\log(\zeta_{F \bullet G}(1))$, the trefoil property (Equation 1) is a straightforward consequence of the following theorem for $z = 1$.

Theorem 13. *Suppose $V^F \cap V^G \cap V^H = \emptyset$. Then:*

$$\zeta_{F \bullet (G :: H)}(z).\zeta_{G \bullet H}(z) = \zeta_{G \bullet (H :: F)}(z).\zeta_{H \bullet F}(z).$$

Proof. By definition and the geometric trefoil property:

$$\begin{aligned}
&\zeta_{F \bullet (G :: H)}(z).\zeta_{G \bullet H}(z) \\
&= \prod_{\pi \in \mathcal{C}(F, G :: H)} (1 - \omega(\pi).z)^{-1} \prod_{\pi \in \mathcal{C}(G, H)} (1 - \omega(\pi).z)^{-1} \\
&= \prod_{\pi \in \mathcal{C}(F, G :: H) \uplus \mathcal{C}(G, H)} (1 - \omega(\pi).z)^{-1} \\
&= \prod_{\pi \in \mathcal{C}(G, H :: F) \uplus \mathcal{C}(H, F)} (1 - \omega(\pi).z)^{-1} \\
&= \prod_{\pi \in \mathcal{C}(G, H :: F)} (1 - \omega(\pi).z)^{-1} \prod_{\pi \in \mathcal{C}(H, F)} (1 - \omega(\pi).z)^{-1} \\
&= \zeta_{G \bullet (H :: F)}(z).\zeta_{H \bullet F}(z) \quad \square
\end{aligned}$$

We can then define families of models of linear logic extending Seiller's approach by considering the following constructs. We change the terminology to avoid conflicts. We use the term *proof object* instead of the term *project*, and we call *types* what is usually called a *conduct*. We also use the term *antipode* for the set of functions defining the orthogonality relation, as the more traditional term "pole" might be confused with the notion of pole from complex analysis.

Definition 14. A *proof-object* of support V is a pair (f, G) of a function $g : \mathbf{C} \to \mathbf{C}$ and a directed weighted graph G with $V^G = V$.

Definition 15. Given two proof objects $\mathfrak{g} = (g, G)$ and $\mathfrak{h} = (h, H)$ we define the *zeta-measurement* as the complex function:

$$\zeta_{\mathfrak{g}, \mathfrak{h}} = g \cdot h \cdot \zeta_{G \bullet H},$$

where $\cdot$ denotes pointwise multiplication of functions.

Definition 16. An antipode P is a family of functions $\mathbf{C} \to \mathbf{C}$. Given two proof objects $\mathfrak{g} = (g, G)$ and $\mathfrak{h} = (h, H)$, they are orthogonal w.r.t. the antipode P – denoted $\mathfrak{g} \perp_P \mathfrak{h}$ – if and only if $\zeta_{\mathfrak{g}, \mathfrak{h}} \in P$.

Given a set E of proof objects, we define its orthogonal as $E^{\perp} = \{\mathfrak{g} \mid \forall \mathfrak{e} \in E, \mathfrak{e} \perp \mathfrak{g}\}$.

We note that many interesting properties of the graph can be used to define orthogonality in this case. Indeed, a number of properties and invariants of a graph can be related to analytic properties of the zeta function of a graph. Note that previous notions of orthogonality [14] can be recovered by considering as an antipode the set of functions which are not equal to 0 or 1 at $z = 1$.

For the sake of self-containment, we define types and explain how models of MLL can be defined from this. We suppose now that an antipode has been fixed until the end of this section. We will therefore omit the subscript when writing the orthogonality.

Definition 17. A *type* of support V is a set $\mathbb{A}$ of proof-objects of support V such that there exists a set B with $\mathbb{A} = B^{\perp}$. Equivalently, a type is a set $\mathbb{A}$ such that $\mathbb{A} = \mathbb{A}^{\perp\perp}$.

The following constructions on type can then be shown to define a model of Multiplicative Linear Logic. For $\mathbb{A}$ and $\mathbb{B}$ two types, we define:

$$\begin{aligned}\mathbb{A} \otimes \mathbb{B} &= \{\mathfrak{a} :: \mathfrak{b} \mid \mathfrak{a} \in \mathbb{A}, \mathfrak{b} \in \mathbb{B}\}^{\perp\perp} \\ \mathbb{A} \multimap \mathbb{B} &= \{\mathfrak{f} \mid \forall \mathfrak{a} \in \mathbb{A}, \mathfrak{f} :: \mathfrak{a} \in \mathbb{B}\}\end{aligned}$$

A model of Multiplicative-Additive Linear Logic can also be constructed by considering linear combinations of proof-objects [15]. Both these constructions are quite automatic and the results are mainly dependent on the two properties cited above: associativity of execution and the trefoil property.

II. Graphings, dynamical systems and Markov processes

In this section, we review the more general setting of Interaction Graphs based on graphings. We first explain how the notions introduced in the previous section generalises, pinpointing how out the general construction based on zeta functions naturally adapts here. We then provide two results explaining how deterministic, respectively probabilistic, graphings correspond to partial dynamical systems, respectively discrete-image Markov processes.

We first recall briefly the notion of graphing. Interested readers can find more detailed presentations in Seiller's recent work on computational complexity [24], [25]. The definition is usually parametrised by a monoid action $\alpha : M \curvearrowright \mathbf{X}$ on the underlying space $\mathbf{X}$, but the choice of the action will not be important in this paper. Graphing come in different flavours (discrete, topological, measurable), depending on the type of space $\mathbf{X}$ one wishes to consider. If $\mathbf{X}$ is a topological space, the parametrising monoid action will be *continuous*. If $\mathbf{X}$ is a measured space, which will be the case considered here, the action will be *measurable*.

More precisely, for technical purposes related to the definition of the measurement [16], the action is supposed to be by NSMP maps, i.e. maps which are non-singular – i.e. the inverse images of negligible sets are negligible sets – and *measure-preserving* – i.e. the (direct) image of a measurable set is measurable. We fix a measured space $\mathbf{X} = (X, \mathcal{X}, \mu)$. Graphings are equivalence classes of graph-like objects called graphing representatives.

Definition 18. An α-graphing representative G (w.r.t. a monoid action $\alpha : M \curvearrowright \mathbf{X}$) is defined as a set of *edges* E^G and for each element $e \in E^G$ a pair (S_e^G, m_e^G) of a subspace S_e^G of $\mathbf{X}$ – the *source* of e – and an element $m_e^G \in M$ – the *realiser* of e.

Similarly, a *weighted α-graphing representative* G is defined as a set of edges E^G and an E^G-indexed family of triples $\{(S_e^G, m_e^G, \omega_e^G) \mid e \in E^G\}$.

In the following, we identify non-weighted graphings with weighted graphings with constant weight equal to 1.

Graphings are then defined as equivalence classes of graphing representatives. The intuition is that a graphing represents an *action* on the underlying space, which can be represented by different graphings. For instance, consider a graphing representative G with a single edge e of source S_e and realised by the monoid element m_e, and the graphing representative H with two edges e_1, e_2 of respective sources S_{e_1} and S_{e_2} and realised by $m_{e_1} = m_{e_2} = m_e$. Then the graphing representatives G and H represent the same action on the underlying space $\mathbf{X}$ as long as[2] $S_e =_{a.e.} S_{e_1} \cup S_{e_2}$ and $S_{e_1} \cap S_{e_2} =_{a.e} \emptyset$. In fact, H is more than equivalent to G, it is a *refinement* of the latter.

Definition 19 (Refinement)**.** A graphing representative F is a refinement of a graphing representative G, noted $F \leqslant G$, if there exists a partition[3] $(E_e^F)_{e \in E^G}$ of E^F such that $\forall e \in E^G$:

$$\mu\left(\left(\cup_{f \in E_e^F} S_f^F\right) \triangle S_e^G\right) = 0; \quad \forall f \in E_e^F,\ m_f^F = m_e^G$$

$$\forall f \neq f' \in E_e^F,\ \mu(S_f^F \triangle S_{f'}^F) = 0;$$

Two graphing representatives F, G can then be considered equivalent (i.e. having the same action on the underlying space) whenever there exists a graphing representative H which is a refinement of both F and G. The fact that this defines an equivalence relation compatible with the operations that are essential to define models of linear logic, is shown in Seiller's first work on graphings [16].

This leads to the notion of graphing, on which notions of execution and measurement can be defined in order to build models of linear logic [16] that we write $\mathbb{M}[\Omega, \alpha]$, where Ω is the monoid of weights (as already mentioned in the previous section, we will only consider the case $\Omega = \mathbf{C}$ in this paper) and α the monoid action. By extension, the notation $\mathbb{M}[\Omega, \alpha]$ also denotes the set of all Ω-weighted α-graphings.

Definition 20. An *α-graphing* is an equivalence class of α-graphing representatives w.r.t. the equivalence relation generated by refinements: $F \sim G$ if and only if there exists H with $H \leqslant F$ and $H \leqslant G$.

We refer the interested reader to Seiller's work [16], [17] for the definitions of execution and measurement of graphings [16]. We will now show how graphings relate to well-established notions in mathematics.

A. Dynamical Systems

Definition 21. A *measured dynamical system* is a pair $(\mathbf{X}, f)$ of a measured space $\mathbf{X}$ and a measurable map $f : \mathbf{X} \to \mathbf{X}$. A *partial measured dynamical system* is a triple $(\mathbf{X}, D, f)$ where $\mathbf{X}$ is a measured space, $D \subset \mathbf{X}$ a subspace – the domain –, and $f : D \to \mathbf{X}$ is a measurable map.

Measured dynamical systems are a well-studied field of mathematics and applies to a range of physical and biological problems. The measured space $\mathbf{X}$ represent the set of states of the system under consideration, while the map f describes the dynamics, i.e. the time-evolution of the system, based on the assumption that those do not vary with time (e.g. they

[2]Note that working with measured spaces, equalities holds up to a null measure set here, while exact equalities may be considered in e.g. the topological framework.

[3]We allow the sets E_e^F to be empty.

are consequences of physical laws which are supposed not to change over time). It is then the iterated maps f^i that are of interest as they describe how the system will evolve.

It is important to realise that dynamical systems represent deterministic systems, such as those described by classical mechanics. If one wants to describe non-deterministic behaviour, one might have to consider several partial maps. It turns out that the resulting object coincides with the notion of *graphing* without weights. One may be interested in describing probabilistic behaviour, which can be done by considering several partial maps assigned with probabilities; the resulting object is then a graphing with weights in $[0,1]$. While we will consider the latter case in the next section, we now focus on the deterministic case.

Definition 22. A graphing $G = \{S_e^G, \phi_e^G, \omega_e^G \mid e \in E^G\}$ is *deterministic* if $\forall e \in E^G,\ \omega_e^G = 1$ and the following holds:

$$\mu\left(\left\{x \in \mathbf{X} \mid \exists e, f \in E^G, e \neq f \text{ and } x \in S_e^G \cap S_f^G\right\}\right) = 0$$

Theorem 23. *There is a one-to-one correspondence between deterministic graphings and partial non-singular measurable-preserving dynamical systems (up to a.e. equality).*

More precisely, deterministic α-graphings are in one-to-one correspondence with partial measured dynamical systems $(\mathbf{X}, D, f)$ such that the graph of f in included in the measurable preorder $\mathcal{P}(\alpha) = \{(x,y) \in \mathbf{X} \times \mathbf{X} \mid \exists m \in M, \alpha(m)(x) = y\}$.

B. A submodel

We now prove that the set of deterministic graphings is closed under the operation of execution, i.e. if F, G are deterministic graphings, then their execution $F :: G$ is again a deterministic graphing. This shows that the sets of deterministic graphings defines a submodel $\mathbb{M}^{\mathtt{det}}[\Omega, \alpha]$ of $\mathbb{M}[\Omega, \alpha]$, i.e. it is a subset of graphings closed under execution and therefore a model of linear logic can be defined on this subset, using the restriction of the measurement defined on $\mathbb{M}[\Omega, \alpha]$.

Lemma 24. *The execution of two deterministic graphings is a deterministic graphing.*

One can then check that the interpretations of proofs by graphings in earlier papers [16], [26], [17] are all deterministic. This gives us the following theorem as a corollary of the previous lemma.

Theorem 25 (Deterministic model)**.** *Let Ω be a monoid and α a monoid action. The set of Ω-weighted* deterministic *α-graphings yields a model, denoted by $\mathbb{M}^{\mathtt{det}}[\Omega, \alpha]$, of multiplicative-additive linear logic.*

Now, this defines a model $\mathbb{M}^{\mathtt{det}}[\Omega, \alpha]$ of linear logic based on the set of all partial measured dynamical systems whose graph is included in $\mathcal{P}(\alpha)$, based on Theorem 23 and Theorem 25. This model uses the measurement defined by Seiller [16] but, as we will now show, this measurement is also related to a standard notion of zeta function.

C. Zeta Functions for dynamical systems

The Ruelle zeta function [27] is defined from a function $f : M \to M$ where M is a manifold and a function $\phi : M \to \mathfrak{M}_k$ a matrix-valued function. We write $\mathrm{Fix}(g)$ the set of fixed points of g. Then the Ruelle zeta function is defined as (we suppose that $\mathrm{Fix}(f^k)$ is finite for all k):

$$\zeta_{f,\Phi}(z) = \exp\left(\sum_{m \geqslant 1} \frac{z^m}{m} \sum_{x \in \mathrm{Fix}(f^m)} \mathrm{tr}\left(\prod_{i=0}^{m-1} \phi(f^i(x))\right)\right)$$

Its even easier here to consider the logarithm,

$$\log(\zeta_{f,\Phi}(z)) = \sum_{m \geqslant 1} \frac{z^m}{m} \sum_{x \in \mathrm{Fix}(f^m)} \mathrm{tr}\left(\prod_{i=0}^{m-1} \phi(f^i(x))\right)$$

For $d = 1$ and $\phi = 1$ the constant function equal to 1, this is the Artin-Mazur [28] zeta function:

$$\zeta_{f,1}(z) = \exp\left(\sum_{m \geqslant 1} \frac{z^m}{m} \mathrm{Card}(\mathrm{Fix}(f^m))\right)$$

Now, we are working with measured spaces, so it is natural to consider the following measured variant of the Ruelle zeta function (defined for measure-preserving maps[4]). Suppose that we work with a measured space $(M, \mathcal{B}, \mu)$ and that $\mathrm{Fix}(f^m)$ is of finite measure:

$$\zeta_{f,\Phi}(z) = \exp\left(\sum_{m \geqslant 1} \frac{z^m}{m} \int_{\mathrm{Fix}(f^m)} \mathrm{tr}\left(\prod_{i=0}^{m-1} \phi(f^i(x))\right) d\mu(x)\right)$$

For $d = 1$ and $\phi = 1$ (the constant function equal to 1), this becomes:

$$\zeta_{f,1}(z) = \exp\left(\sum_{m \geqslant 1} \int_{\mathrm{Fix}(f^m)} \frac{z^m}{m}\right)$$

which is – at $z = 1$ – the exponential of the measurement on graphings considered in earlier work [16].

Proposition 26. *Given measure-preserving* NSMP *partial dynamical systems $f, g : \mathbf{X} \to \mathbf{X}$, we have the following equality for all constant c:*

$$[\![f, g]\!]_{\lambda x.c} = \log(\zeta_{g \circ f.1}(c)),$$

where $[\![\_,\_]\!]_m$ denotes the measurement between graphings defined in earlier work [16].

Based on this result, models based on deterministic graphings – or equivalently on partial measured dynamical systems – can be defined based on realisability techniques using zeta functions to compute the orthogonality.

[4]Based on the result of Proposition 26, a definition for general NSMP maps could be obtained using the method used by Seiller [16] to define a generalised measurement between graphings. However, we considered this to be out of the scope of this work.

III. Probabilities and kernels

One can also consider several other classes of graphings. We explain here the simplest non-classical model one could consider, namely that of *probabilistic graphings*. In order for this notion to be of real interest, one should suppose that the unit interval $[0,1]$ endowed with multiplication is a submonoid of Ω. We will show how this can be used to define a *sub-probabilistic model*, and how the corresponding objets can be understood as (some) sub-Markov kernels.

A. A probabilistic model

Definition 27. A graphing $G = \{S_e^G, \phi_e^G, \omega_e^G \mid e \in E^G\}$ is *sub-probabilistic* if the following holds:

$$\mu\left(\left\{x \in \mathbf{X} \mid \sum_{e \in E^G, x \in S_e^G} \omega_e^G > 1\right\}\right) = 0$$

It turns out that this notion of graphing also behaves well under composition, i.e. there exists a *sub-probabilistic* submodel of $\mathbb{M}[\Omega, \alpha]$, namely the model of *sub-probabilistic graphings*. As explained below in the more general case of Markov processes, probabilistic graphings are *not* closed under composition.

Theorem 28. *The execution of two sub-probabilistic graphings is a sub-probabilistic graphing.*

Theorem 29 (Probabilistic model)**.** *Let Ω be a monoid and $\alpha : M \curvearrowright \mathbf{X}$ a monoid action. The set of Ω-weighted* probabilistic *α-graphings yields a model, denoted by $\mathbb{M}^{\mathtt{prob}}[\Omega, \alpha]$, of multiplicative-additive linear logic.*

B. Discrete-image sub-Markov processes

We now consider probabilistic systems. More specifically, we consider systems for which evolution is still time-independent, not deterministic, but which obey the principle of probabilistic choices: given a state, t may produce different outputs but these different choices are provided with a probability distribution. The notion of dynamical system, i.e. a map from a measured space to itself, is then no longer the right object to formalise this idea. In fact, a probabilistic time evolution do not act on the states of the system but rather on the set of probability distributions on this set of states.

Definition 30. Let $\mathbf{X}$ be a measured space. We denote $\mathbb{P}(\mathbf{X})$ the set of sub-probability distributions over $\mathbf{X}$, i.e. the set of sub-probability measures on $\mathbf{X}$.

The space of probability distributions is a convex space: if p, q are probability distributions and α, β are positive real numbers such that $\alpha + \beta = 1$, then $\alpha p + \beta q$ is again a probability distribution. It is a topological space, endowed with the weak* topology, and it is weak* compact.

Now, a deterministic system also acts on the set of probability measures by post-composition. If $(\mathbf{X}, f)$ is a measured dynamical system, then given a (sub-)probability distribution (otherwise called a *random variable*) $p : \mathbf{P} \to \mathbf{X}$, the map $f \circ p$ is itself a (sub-)probability distribution. In the same way deterministic graphings, defining dynamical systems, act on the set of (sub-)probability distributions, sub-probabilistic graphings will act on $\mathbb{P}(\mathbf{X})$. In fact, we show that sub-probabilistic graphings define sub-Markov kernels. We recall briefly that sub-probability distributions on $\mathbf{X}$ are Markov kernels from the one-point space $\{*\}$ to $\mathbf{X}$, and the action of a sub-Markov kernel onto $\mathbb{P}(\mathbf{X})$ is defined as post-composition (using the composition of kernels) [29].

Definition 31. Let $\mathbf{X} = (X, \mathcal{X}, \mu)$ and $\mathbf{Y} = (Y, \mathcal{Y}, \nu)$ be measured spaces. A sub-Markov kernel on $\mathbf{X} \times \mathbf{Y}$ is a measurable map $\kappa : X \times \mathcal{Y} \to [0,1]$ with the properties that for all $x \in X$ and $B \in \mathcal{Y}$, $\kappa(x, \_)$ is a subprobability measure on X and $\kappa(\_, B)$ is a measurable function.

If $\kappa(x, \_)$ is a probability measure, κ is a Markov kernel.

Within this section, we furthermore restrict to what we call *discrete-image kernels*.

Notations 32. In this section and the following, we will consider measured spaces $\mathbf{X}$, $\mathbf{Y}$, etc. noted in boldface fonts. The implicit assumption will be that the underlying set is named by the same letter in normal fonts, e.g. X, Y, etc. and the associated σ-algebra is named by the same letter in calligraphic fonts, e.g. $\mathcal{X}$, $\mathcal{Y}$, etc. We do not assume a generic notation for the measures and, should the need to talk about them arise, we would explicitly name them.

Definition 33. A *discrete-image kernel* is a sub-Markov kernel κ on $\mathbf{X} \times \mathbf{Y}$ such that for all $x \in \mathbf{X}$, $\kappa(x, \_)$ is a discrete probability distribution.

Notations 34. To simplify equations, we write $\dot{x}$ instead of the usual dx (or $d\mu(x)$) in the equations. With this notation, the composition of the kernels κ on $\mathbf{X} \times \mathbf{Y}$ and κ' on $\mathbf{Y} \times \mathbf{Z}$ is computed as follows:

$$\kappa' \circ \kappa(x, \dot{z}) = \int_Y \kappa(x, \dot{y}) \kappa'(y, \dot{z}).$$

Theorem 35. *There is a one-to-one correspondence between sub-probabilistic graphings on $\mathbf{X}$ and discrete-image sub-Markov kernels on $\mathbf{X} \times \mathbf{X}$.*

Proof. The fact that sub-probabilistic graphings define sub-Markov processes is quite easy. One defines from a graphing $G = \{S_e^G, \phi_e^G, \omega_e^G \mid e \in E^G\}$ the kernel:

$$\kappa_G : X \times \mathcal{X} \to [0,1]; (x, y) \mapsto \sum_{e \in E^G, x \in S_e^G, \phi_e^G(x) = y} \omega_e^G.$$

The fact that is is a discrete-image sub-Markov kernel is clear.

The converse, i.e. given a kernel κ, define a graphing G_κ is more involved. The difficulty lies in the fact that one has to collect the pairs (x, y) such that $\kappa(x, y) > 0$ into a countable collection of measurable maps. The key ingredients to make this work are: the countability of $\{Y \in \mathcal{X} \mid \kappa(x, Y) > 0\}$ for all $x \in X$ (because κ is supposed to be a discrete-image kernel), the possibility to approximate all real numbers by a (countable) sequence of rational numbers, the measurability of $\kappa(\_, B)$ for all $B \in \mathcal{X}$. □

IV. Markov processes and linear logic

Based on the previous sections, we want to extend the constructions to general Markov processes. For technical reasons we will illustrate below, the model will be defined on sub-Markov kernels.

Notations 36. In the following we write $\mathbf{1}$ the *identity kernel* on $\mathbf{X} \to \mathbf{X}$, i.e. the Dirac delta function $\mathbf{1}(x, \dot{x}) = \delta(x, \dot{x})$ satisfying $\int_A \mathbf{1}(x, \dot{a}) = 1$ whenever $x \in A$ and $\int_A \mathbf{1}(x, \dot{x}) = 0$ otherwise.

We will now define the two key ingredients of the model: the execution and the zeta function. We then proceed to prove the cocycle property which will ensure the construction of linear logic types.

Definition 37 (Iterated kernel)**.** Let κ be a sub-Markov kernel on $\mathbf{X} \times \mathbf{Y}$. For $k > 1$, we define the k-th iterated kernel $\kappa^{(k)}$ as

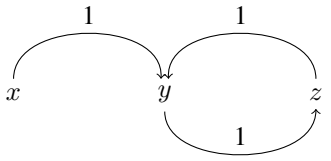
$$\kappa^{(k)}(x_0, \dot{x}_k) = \iint_{(x_1,\dots,x_{k-1}) \in (\mathbf{X} \cap \mathbf{Y})^{k-1}} \prod_{i=0}^{k-1} \kappa(x_i, \dot{x}_{i+1}).$$

By convention, $\kappa^{(1)} = \kappa$.

Definition 38 (Maximal paths – Execution kernel)**.** Let κ be a sub-Markov kernel on $\mathbf{X} \times \mathbf{Y}$. We define the *execution kernel* of κ as the map (in the formula, x_{n+1} is used as a notation for y):

$$\begin{array}{rccl} \mathrm{tr}(\kappa) & : & X \backslash Y \times \mathcal{Y} \backslash \mathcal{X} & \to & [0,1] \\ & & (x, y) & \mapsto & \sum_{n \geqslant 1} \kappa^{(n)}(x, y). \end{array}$$

While this maps could be defined on the whole space $X \times \mathcal{Y}$, the restriction is needed to define a sub-Markov kernel. This can be understood on a very simple Markov chain:



On this figure, the partial sums of $\kappa^{(i)}(x, y)$ is a diverging series. This example also shows why the resulting kernel could be a sub-Markov kernel even when κ is a proper Markov kernel. To ensure that $\mathrm{tr}(\kappa)$ is a Markov kernel, additional assumptions on κ are required. We will discuss these in the conclusion. For the moment, we prove that $\mathrm{tr}^A(\kappa)$ is indeed a sub-Markov kernel.

Lemma 39. *If κ is a sub-Markov kernel, $\mathrm{tr}^A(\kappa)$ is well-defined and a sub-Markov kernel.*

Proof. The gist of the proof is an induction to establish that for all integer k and measurable subset A such that $A \cap \mathbf{X} \cap \mathbf{Y} = \emptyset$, the expression $\int_{a \in A} \sum_{i=1}^{k} \kappa^{(i)}(x, a)$ is bounded by 1. This is clear for $k = 1$ from the assumption that κ is a sub-Markov kernel. The following computation then establishes the induction (we write $x = y_0$ to simplify the equations):

$$\int_{a \in A} \sum_{i=1}^{k+1} \kappa^{(i)}(y_0, \dot{a}) = \int_{a \in A} \kappa(y_0, \dot{a}) + \int_{a \in A} \sum_{i=0}^{k} \kappa^{(i+1)}(y_0, \dot{a}).$$

We now bound the second term as follows, using the induction hypothesis to establish that $\left[\int_{a \in A} \sum_{i=1}^{k} \kappa^{(i)}(y_1, a)\right] \leqslant 1$:

$$\begin{aligned} &\int_{a \in A} \sum_{i=0}^{k} \kappa^{(i+1)}(y_0, \dot{a}) \\ &= \int_{a \in A} \sum_{i=0}^{k} \int_{y_1} \cdots \int_{y_i} \kappa(y_i, \dot{a}) \prod_{j=0}^{i-1} \kappa(y_j, \dot{y}_{j+1}) \\ &= \int_{y_1} \int_{a \in A} \sum_{i=0}^{k} \int_{y_2} \cdots \int_{y_i} \kappa(y_i, \dot{a}) \prod_{j=0}^{i-1} \kappa(y_j, \dot{y}_{j+1}) \\ &= \int_{y_1} \kappa(y_0, \dot{y}_1) \left[\int_{a \in A} \sum_{i=1}^{k} \int_{y_2} \cdots \int_{i} \kappa(y_i, \dot{a}) \prod_{j=0}^{i-1} \kappa(y_j, \dot{y}_{j+1}) \right] \\ &= \int_{y_1} \kappa(y_0, \dot{y}_1) \left[\int_{a \in A} \sum_{i=1}^{k} \kappa^{(i)}(y_1, \dot{a}) \right] \\ &\leqslant \int_{y_1} \kappa(y_0, \dot{y}_1). \end{aligned}$$

Coming back to the initial expression, we obtain the required bound by using the additivity of κ (we recall that A and $\mathbf{X} \cap \mathbf{Y}$ do not intersect):

$$\int_{a \in A} \sum_{i=1}^{k+1} \kappa^{(i)}(y_0, \dot{a}) \leqslant \kappa(y_0, A) + \kappa(y_0, \mathbf{X} \cap \mathbf{Y}) \leqslant 1. \quad \square$$

Now, the execution kernel just defined is the main operation for defining the execution of sub-Markov kernels, as we will explain in the next section. We now define the second ingredient, namely the zeta function. For this, we first define a map which we abusively call the "zeta kernel", although it is not a kernel as we explain below.

Definition 40 (Finite orbits – Zeta kernel)**.** Let κ be a sub-Markov kernel on $\mathbf{X} \times \mathbf{Y}$. The *zeta kernel*, or kernel of finite orbits of κ is a kernel on $\mathbf{X} \times \mathbf{N}$ – where $\mathbf{N}$ denotes the set of natural numbers – defined as:

$$\zeta_\kappa(x_0, \dot{x}_0, n) = \iint_{(x_1,\dots,x_{n-1}) \in (\mathbf{X} \cap \mathbf{Y})^{n-1}} \prod_{i \in \mathbf{Z}/n\mathbf{Z}} \kappa(x_i, \dot{x}_{i+1}).$$

Notice that this expression computes the probability that a given point x_0 lies in an orbit of length n. It is a sub-Markov kernel for each fixed value of n, but the sum over $n \in \mathbf{Z}$ is not. The reason is simple: if a point x lies in a length 2 orbit with probability 1 (e.g. the point y in the example Markov chain above), then it lies in a length $2k$ orbit with probability 1 as well. However, let us remark that the expression

$$\int_{x \in X \cap Y} \zeta_\kappa(x, \dot{x}, n)$$

plays the rôle of the set $\mathrm{Fix}(f^n)$ that appears in dynamical and graph zeta functions.

Definition 41 (Zeta function)**.** We now define the Zeta function associated with a sub-Markov kernel κ on $\mathbf{X} \times \mathbf{Y}$:

$$\zeta_\kappa(z) : z \mapsto \exp\left(\sum_{n=1}^{\infty} \frac{z^n}{n} \int_{x \in \mathbf{X} \cap \mathbf{Y}} \zeta_\kappa(x, \dot{x}, n)\right)$$

A. Execution and the Cocycle Property

Definition 42. Given two sub-Markov kernels κ on $\mathbf{X} \times \mathbf{X}'$ and κ' on $\mathbf{Y} \times \mathbf{Y}'$, we define their execution $\kappa :: \kappa'$ as the kernel $\mathrm{tr}(\kappa \bullet \kappa')$ where:

$$\kappa \bullet \kappa' = (\kappa + \mathbf{1}_{\mathbf{Y} \setminus \mathbf{X}'}) \circ (\kappa' + \mathbf{1}_{\mathbf{X}' \setminus \mathbf{Y}})$$

The reader with notions from *traced monoidal categories* [30], [31], [32] should not be surprised of this definition and the following properties.

Definition 43. Three sub-Markov kernels κ on $\mathbf{X} \times \mathbf{X}'$, κ' on $\mathbf{Y} \times \mathbf{Y}'$, and κ'' on $\mathbf{Z} \times \mathbf{Z}'$ are said to be *in general position*[5] when the following condition is met:

$$\mu(\mathbf{X}' \cap \mathbf{Y} \cap \mathbf{Z}) = \mu(\mathbf{Y}' \cap \mathbf{Z} \cap \mathbf{X}) = \mu(\mathbf{Z}' \cap \mathbf{X} \cap \mathbf{Y}) = 0,$$
$$\mu(\mathbf{X} \cap \mathbf{Y}' \cap \mathbf{Z}') = \mu(\mathbf{Y} \cap \mathbf{Z}' \cap \mathbf{X}') = \mu(\mathbf{Z} \cap \mathbf{X}' \cap \mathbf{Y}') = 0.$$

Note that if $\mathbf{X} = \mathbf{X}'$, $\mathbf{Y} = \mathbf{Y}'$ and $\mathbf{Z} = \mathbf{Z}'$, the condition becomes $\mu(\mathbf{X} \cap \mathbf{Y} \cap \mathbf{Z}) = 0$, which is the condition of application of the associativity of execution and of the trefoil property in the graph case.

Lemma 44. *Given three sub-Markov kernels κ_0 on $\mathbf{X} \times \mathbf{X}'$, κ_1 on $\mathbf{Y} \times \mathbf{Y}'$, and κ_2 on $\mathbf{Z} \times \mathbf{Z}'$ in general position:*

$$(\kappa_0 :: \kappa_1) :: \kappa_2 = \kappa_0 ::(\kappa_1 :: \kappa_2).$$

Lemma 45. *Given two sub-Markov kernels κ on $\mathbf{X} \times \mathbf{X}'$ and κ' on $\mathbf{Y} \times \mathbf{Y}'$ such that $\mathbf{X} \cap \mathbf{Y} = \mathbf{X}' \cap \mathbf{Y}' = \emptyset$:*

$$\kappa :: \kappa' = \kappa' :: \kappa.$$

However, having a well-defined associative execution is not enough to model linear logic. Following what was exposed in the first sections, we now define a zeta function associated to a sub-Markov process, and show that this zeta function and the execution satisfy the required cocycle property.

Definition 46. Given two kernels κ, κ', we define their *zeta-measurement* $\zeta_{\kappa,\kappa'}$ as the function $\zeta_{\kappa \bullet \kappa'}(z)$.

Proposition 47 (Cocycle)**.** *Given three sub-Markov kernels κ on $\mathbf{X} \times \mathbf{X}'$, κ' on $\mathbf{Y} \times \mathbf{Y}'$, and κ'' on $\mathbf{Z} \times \mathbf{Z}'$ in general position:*

$$\zeta_{\kappa,\kappa'}(z)\zeta_{\kappa :: \kappa',\kappa''}(z) = \zeta_{\kappa' :: \kappa'',\kappa}(z)\zeta_{\kappa',\kappa''}(z)$$

To construct the model of linear logic, we will now follows the usual process. We need to consider not only kernels, but pairs of a kernel and a function. This is used to capture the information about closed paths appearing during the execution, as in the graph case. This is discussed by Seiller in earlier work [14], and is essential to obtain Theorem 54.

[5]The reader will realise the terminology is inspired from algebraic geometry, but no formal connections should be expected.

B. A first model of Linear Logic

To obtain a model of linear logic, one has to consider sub-Markov kernels with a set of states. Following Seiller's construction of a model of second-order linear logic [17], we will consider here that the set of states is equal to the segment $[0, 1]$. Note however that fragments of linear logic can be modelled when the set of states is chosen to be discrete [26], so a model where all sets of states are considered is possible to describe although it would be in some cases impossible to interpret some constructions.

Definition 48. A *proof-object* of support $\mathbf{X}$ is a pair $\mathfrak{f} = (f, F)$ of a function $\mathbf{C} \to \mathbf{C}$ and a sub-Markov kernel F on $(\mathbf{X} \times [0, 1]) \times (\mathbf{X} \times [0, 1])$.

We define the operations $(\_)^\dagger$ and $(\_)^\ddagger$ that will be used throughout the constructions. These operations are meant to ensure that the set of states of two proof-objects do not interact. Indeed, those should be understood as sets of control states, such as the states of automata, and the set of state of a composition is defined as the product of the sets of states of the two objects composed. Given a sub-Markov kernel $F : (\mathbf{X} \times [0, 1]) \times (\mathbf{X} \times [0, 1]) \to [0, 1]$, we define $(\kappa)^\dagger$ and $(\kappa)^\ddagger$ as the following sub-Markov kernels $(\mathbf{X} \times [0, 1] \times [0, 1]) \times (\mathbf{X} \times [0, 1] \times [0, 1]) \to [0, 1]$:

$$\begin{array}{rccl} (\kappa)^\dagger : & ((x, e, f), (\dot{x}, \dot{e}, \dot{f})) & \mapsto & \kappa((x, e), (\dot{x}, \dot{e}))\mathbf{1}(f, \dot{f}) \\ (\kappa)^\ddagger : & ((x, e, f), (\dot{x}, \dot{e}, \dot{f})) & \mapsto & \kappa((x, f), (\dot{x}, \dot{f}))\mathbf{1}(e, \dot{e}). \end{array}$$

Definition 49. Given two proof objects $\mathfrak{f} = (f, \kappa_\mathrm{F})$ and $\mathfrak{g} = (g, \kappa_\mathrm{G})$ we define the *zeta-measurement* as the complex function:

$$\zeta_{\kappa_\mathrm{F},\kappa_\mathrm{H}} : z \mapsto f(z).g(z).\zeta_{\kappa_\mathrm{F}^\dagger \bullet \kappa_\mathrm{F}^\ddagger}(z).$$

Definition 50. The execution of two proof objects $\mathfrak{f} = (f, \kappa_\mathrm{F})$ and $\mathfrak{g} = (g, \kappa_\mathrm{G})$, of respective supports $\mathbf{X}$ and $\mathbf{Y}$, is defined as the proof-object:

$$(f.g.\zeta_{\kappa_\mathrm{F},\kappa_\mathrm{G}}, \kappa_\mathrm{F}^\dagger :: \kappa_\mathrm{G}^\ddagger).$$

Note that this is a proof-object up to isomorphism between $[0, 1]$ and $[0, 1] \times [0, 1]$.

Based on Lemma 44 and Lemma 45 and the associativity and commutativity of the pointwise product of functions, this notion of execution is associative and commutative.

We now define the orthogonality relation through the zeta function. This follows what we exposed above in the case of graphs.

Definition 51. An antipode P is a family of functions $\mathbf{C} \to \mathbf{C}$. Given two proof objects $\mathfrak{f} = (f, \kappa_\mathrm{F})$ and $\mathfrak{g} = (g, \kappa_\mathrm{G})$ of support $\mathbf{X}$, they are orthogonal w.r.t. the antipode P – denoted $\mathfrak{f} \perp_P \mathfrak{g}$ – if and only if $\zeta_{\mathfrak{f},\mathfrak{g}} \in P$.

We suppose now that an antipode has been fixed until the end of this section. We will therefore omit the subscript when writing the orthogonality.

Definition 52. A *type* of support $\mathbf{V}$ is a set $\mathbb{A}$ of proof-objects of support $\mathbf{V}$ such that there exists a set B with $\mathbb{A} = B^{\perp}$. Equivalently, a type is a set $\mathbb{A}$ such that $\mathbb{A} = \mathbb{A}^{\perp\perp}$.

Definition 53. For $\mathbb{A}$ and $\mathbb{B}$ two types of disjoint supports, we define:

$$\begin{aligned} \mathbb{A} \otimes \mathbb{B} &= \{\mathfrak{a} :: \mathfrak{b} \mid \mathfrak{a} \in \mathbb{A}, \mathfrak{b} \in \mathbb{B}\}^{\perp\perp} \\ \mathbb{A} \multimap \mathbb{B} &= \{\mathfrak{f} \mid \forall \mathfrak{a} \in \mathbb{A}, \mathfrak{f} :: \mathfrak{a} \in \mathbb{B}\} \end{aligned}$$

The following is a direct consequence of the cocycle property. We omit the proof as it follows the proof of the same statement in Interaction Graphs models [14], [15], [16].

Theorem 54. *For any two types $\mathbb{A}, \mathbb{B}$ with disjoint support:*

$$(\mathbb{A} \otimes \mathbb{B}^{\perp})^{\perp} = \mathbb{A} \multimap \mathbb{B}.$$

Now, to define exponentials, one has to restrict to specific spaces. Indeed, not all sub-Markov kernels can be exponentiated. This is easy to understand: if a proof-object uses several copies of its argument, it uses it through its set of states. To understand how states allow for this, consider two automata that are composed. If the first automata has two states, it can ask the first automata to perform a computation, change state, and then ask again, triggering two computations of the second machine. This works perfectly provided the second machine ends its computation on its initial state, otherwise it would not run correctly the second time as there is not way to reinitiate it. This issue is dealt with in the models by exponentiation, as only exponentiated processes can be used multiple times. To ensure the latter end their computation in the same state as they started, exponentiation erases the states to define a single-state machine, while the states are encoded in the configurations of the machine to avoid information loss. This encoding requires the underlying space $\mathbf{X}$ to be large enough, that is to contain the space $[0,1]^{\mathbf{N}}$. Exponentiation is therefore defined as long as the underlying space $\mathbf{X}$ contains $[0,1]^{\mathbf{N}}$.

We now restrict to the spaces of the form $\mathbf{X} = \mathbf{Y} \times [0,1]^{\mathbf{N}}$ in order to define exponentials.

Definition 55. A proof-object $\mathfrak{f} = (f, \kappa_{\mathrm{F}})$ is *balanced* if $f = 0$. If E is a set of proof-objects, we write $\mathrm{bal}(E)$ the subset of balanced proof-objects in E.

Following Seiller's model [17], we will define the exponential through the following maps for all space $\mathbf{X}$ as above:

$$B_{\mathbf{X}} : \left\{ \begin{array}{rcl} \mathbf{Y} \times [0,1]^{\mathbf{N}} \times [0,1] & \to & \mathbf{Y} \times [0,1]^{\mathbf{N}} \\ (a,s,d) & \mapsto & (a, d:s) \end{array} \right.$$

where : denotes here the concatenation. This map is used to define $!\kappa$ from a sub-Markov kernel $\kappa : (\mathbf{X} \times [0,1]) \times (\mathbf{X} \times [0,1]) \to [0,1]$ (we recall that the copies of $[0,1]$ here represent the set of states of the proof-object). We first define[6] $B_{\mathbf{X}}^{-1} \circ \kappa \circ B_{\mathbf{X}}$, which is a sub-Markov kernel $\mathbf{X} \times \mathbf{X} \to [0,1]$, and then $!\kappa : (\mathbf{X} \times [0,1]) \times (\mathbf{X} \times [0,1]) \to [0,1]$ can be defined as follows:

$$!\kappa : (x, e, \dot{x}, \dot{e}) \mapsto B_{\mathbf{X}}^{-1} \circ \kappa \circ B_{\mathbf{X}}(x, \dot{x}) \mathbf{1}(e, \dot{e}).$$

Note that the information of the states of κ is encoded in $!\kappa$ within the space $\mathbf{X}$ and the latter acts on the set of states as the identity, i.e. as if it has a single state.

Definition 56 (Perennisation)**.** Let $\mathfrak{f} = (0, \kappa_{\mathrm{F}})$ be a balanced proof-object. We define its *perennisation* $!\mathfrak{f} = (0, !\kappa_{\mathrm{F}})$.

Definition 57 (Exponential)**.** Let $\mathbb{A}$ be a type. We define the perrenial type $!\mathbb{A}$ as the bi-orthogonal closure $!\mathbb{A} = (\sharp\mathbb{A})^{\perp\perp}$ where $\sharp\mathbb{A}$ is the set $\sharp\mathbb{A} = \{!\mathfrak{a} \mid \mathfrak{a} \in \mathrm{bal}(\mathbb{A})\}$.

As a consequence, we get a model of second-order linear logic (LL^2) using the construction from Seiller's work on graphings [17].

Theorem 58. *Restricting to spaces $\mathbf{X} = \mathbf{Y} \times [0,1]^{\mathbf{N}}$, proof-objects and types define a sound model of* LL^2.

Proof. All the work has been done already. Indeed, the interpretations of linear logic proofs are defined in IG for full linear logic [17], and it is sufficient to remark that those are interpreted by *deterministic graphings*. As such, they are in fact interpreted by dynamical systems by Theorem 23, which in turn define sub-Markov kernels. □

Obviously, this interpretation of linear logic can now be extended with terms for sampling distributions in a natural way, as well as probabilistic sums of proofs (this aspect is already discussed in previous work [17]). However, as we will show in the next section, we can model LL^2 in the unrestricted setting of sub-Markov kernels.

V. Sub-Markov Processes and Linear Logic

Notations 59. When writing down explicit formulas for the value of a sub-kernel κ on $(\mathbf{X} \times [0,1]) \times (\mathbf{Y} \times [0,1])$, we will notationally separate the set of states and the spaces $\mathbf{X}$ and $\mathbf{Y}$. I.e. we will write $\kappa : \mathbf{X} \cdot [0,1] \times \mathbf{Y} \cdot [0,1]$ and write explicit definitions as $\kappa(x; \dot{y}) \cdot (e; \dot{f})$ to denote $\kappa(x, e, \dot{y}, \dot{f})$. This will help clarify the definitions and computations in the following, especially when the spaces $\mathbf{X}$ and $\mathbf{Y}$ contain copies of $[0,1]$.

We restricted the previous model to spaces of the form $\mathbf{X} = \mathbf{Y} \times [0,1]^{\mathbf{N}}$ to ease the presentation and proofs. However, it can be shown that the set of proof-objects and types – without any restrictions on the underlying spaces – is also a model of linear logic. Indeed, instead of considering that κ and $!\kappa$ should act on the same space, it is possible to consider that while κ is defined on $\mathbf{X} \times \mathbf{Y}$, $!\kappa$ is defined on $(\mathbf{X} \times [0,1]) \times (\mathbf{Y} \times [0,1])$. In that case, no assumptions need to be made on $\mathbf{X}$ and $\mathbf{Y}$, and $!\kappa$ is simply defined as the kernel on

$$(\mathbf{X} \times [0,1]) \cdot [0,1] \times (\mathbf{Y} \times [0,1]) \cdot [0,1]$$

where the second copy of $[0,1]$ represent the set of states and $!\kappa(x, e; \dot{x}, \dot{e}) \cdot (f; \dot{f}) = \kappa(x; \dot{x}) \cdot (e; \dot{e}) \times \mathbf{1}(f, \dot{f})$.

[6] Here B is a bijective map, and not a kernel, but we implicitly use the kernel composition by considering the kernel form of B and B^{-1}.

Notations 60. In the following, when considering proof-objects (f, κ_{F}), we will say κ_{F} is a sub-Markov kernel *from* $\mathbf{X}$ *to* $\mathbf{Y}$ to express that κ_{F} has type $\mathbf{X} \cdot [0,1] \times \mathbf{Y} \cdot [0,1] \to [0,1]$.

We now detail this construction, which will require us to redefine the interpretations of linear logic proofs by adapting previous techniques [17].

A. Multiplicatives

We here sketch the basic definitions. The definition of orthogonality, types, and multiplicative connectives follows the constructions exposed in Section IV-B. Additive connectives and quantifiers can also be constructed, following blindly previous constructions [15], [16], [26], [17]. Due to space constraint, we skip those to focus on the real novelty: the construction of exponentials in this larger setting.

Definition 61. A *general proof-object* of support $\mathbf{X} \to \mathbf{Y}$ is a pair $\mathfrak{f} = (f, \kappa_{\mathrm{F}})$ of a complex function f and a sub-Markov kernel κ_{F} from $\mathbf{X}$ to $\mathbf{Y}$.

In the following, as above, an antipode P is a family of functions $\mathbf{C} \to \mathbf{C}$.

Definition 62. Two general proof-objects $\mathfrak{f}, \mathfrak{g}$ of respective supports $\mathbf{X} \to \mathbf{Y}$ and $\mathbf{Y} \to \mathbf{X}$ are orthogonal w.r.t. an antipode P, which is denoted by $\mathfrak{f} \perp_P \mathfrak{g}$, when $\zeta_{\mathfrak{f},\mathfrak{g}} \in P$.

Definition 63. Given two general proof-objects $\mathfrak{f}, \mathfrak{g}$ of respective supports $\mathbf{X} \to \mathbf{Y}$ and $\mathbf{X}' \to \mathbf{Y}'$, their execution is the proof-object of support $(\mathbf{X} \cup \mathbf{X}') \backslash (\mathbf{Y} \cup \mathbf{Y}') \to (\mathbf{Y} \cup \mathbf{Y}') \backslash (\mathbf{X} \cup \mathbf{X}')$ defined as $\mathfrak{f} :: \mathfrak{g} = (\zeta_{\mathfrak{f},\mathfrak{g}}, \kappa_{\mathrm{F}} :: \kappa_{\mathrm{G}})$.

Remark 64. Notice that the execution is not commutative here. Commutativity can be shown as long as one requires that $\mathbf{X} \cap \mathbf{X}'$ and $\mathbf{Y} \cap \mathbf{Y}'$ are negligible.

Definition 65. Let $\mathfrak{f}, \mathfrak{g}$ be two general proof-objects of respective supports $\mathbf{X} \to \mathbf{Y}$ and $\mathbf{X}' \to \mathbf{Y}'$, where $\mathbf{X}, \mathbf{X}', \mathbf{Y}, \mathbf{Y}'$ are pairwise disjoint. We write $\mathfrak{f} \otimes \mathfrak{g}$ the execution of $\mathfrak{f}$ and $\mathfrak{g}$. Note that $\mathfrak{f} \otimes \mathfrak{g} = \mathfrak{g} \otimes \mathfrak{f}$ by the above remark.

B. Exponentials

We now redefine exponentials for arbitrary proof-objects.

Definition 66. Let $\mathfrak{f} = (\mathbf{1}, \kappa)$ be a proof-object, where κ is a sub-Markov process from $\mathbf{X}$ to $\mathbf{Y}$. We define $!\mathfrak{f}$ as the proof object $(\mathbf{1}, !\kappa)$ where $!\kappa$ is the sub-Markov process from $\mathbf{X} \times [0,1]$ to $\mathbf{Y} \times [0,1]$ defined as:

$$!\kappa((x, e; \dot{x}, \dot{e}) \cdot (f; \dot{f})) = \kappa(x; \dot{x}) \cdot (e; \dot{e}) \mathbf{1}(f; \dot{f})$$

Definition 67 (Exponentiation)**.** Let $\mathfrak{f} = (\mathbf{1}, \kappa_{\mathrm{F}})$ be a balanced proof-object. We define its *exponential* $!\mathfrak{f} = (\mathbf{1}, !\kappa_{\mathrm{F}})$.

We will now show that the principles of linear logic can be interpreted faithfully. To do this, we introduce some notations.

Notations 68. Given a map $f : \mathbf{X} \to \mathbf{Y}$, it induces a kernel κ_f on $\mathbf{X} \times \mathbf{Y}$ defined as $\kappa_f(x, \dot{y}) = \mathbf{1}(f(x), \dot{y})$. It also induces a kernel κ_f^* on $\mathbf{Y} \times \mathbf{X}$ defined as $\kappa_f^*(y, \dot{x}) = \mathbf{1}(f(x), \dot{y})$. Note that if f is bijective, $\kappa_f^* = \kappa_{f^{-1}}$.

We will also use the sum symbol $+$ to denote the parallel composition of kernels, i.e. given kernels κ on $\mathbf{X} \times \mathbf{Y}$ and κ' on $\mathbf{Z} \times \mathbf{W}$, the kernel $\kappa + \kappa'$ on $(\mathbf{X} + \mathbf{Z}) \times (\mathbf{Y} \times \mathbf{W})$ is defined as

$$(u, v) \mapsto \begin{cases} \kappa(u, \dot{v}) & \text{if } u \in \mathbf{X},\ \dot{v} \in \mathbf{Y} \\ \kappa'(u, \dot{v}) & \text{if } u \in \mathbf{Z},\ \dot{v} \in \mathbf{W} \\ 0 & \text{otherwise} \end{cases}$$

Lastly, if κ is a kernel on $\mathbf{X} \times \mathbf{Y}$, we write $\bar{\kappa}$ the kernel κ extended with a dialect on which it acts as the identity, i.e. $\bar{\kappa}$ is the kernel on $(\mathbf{X} \times [0,1]) \times (\mathbf{Y} \times [0,1])$ defined as $\bar{\kappa}(x; \dot{y}) \cdot (e; \dot{f}) = \kappa(x, \dot{y}) \mathbf{1}(e, \dot{f})$.

The following claim, established by Seiller [26, Proposition 37], will be particularly useful in the following proofs. It states that to prove a proof-object $\mathfrak{f}$ belongs to $\mathbb{A} \multimap \mathbb{B}$, it is enough to prove $\mathfrak{f} :: \mathfrak{a} \in \mathbb{B}$ when $\mathfrak{a}$ ranges over a *generating* set for $\mathbb{A}$.

Claim 69. *Let $\mathbb{A}, \mathbb{B}$ be types and E a generating set for $\mathbb{A}$, i.e. $\mathbb{A} = E^{\perp\perp}$. If $\mathfrak{f}$ is such that $\forall \mathfrak{a} \in \mathbb{A}, \mathfrak{f} :: \mathfrak{a} \in \mathbb{B}$, then $\mathfrak{f}$ belongs to the type $\mathbb{A} \multimap \mathbb{B}$.*

Proposition 70. *The digging rule can be interpreted.*

Sketch. Suppose $\mathbb{A}$ is of support $\mathbf{X} \to \mathbf{Y}$. This map is implemented as a project $\mathfrak{dig}_{\mathbf{X} \to \mathbf{Y}} = (\mathbf{1}, \kappa_{\mathrm{dig}_{\mathbf{Y}}} + \kappa^*_{\mathrm{d}_{\mathbf{X}}})$ with $\kappa^*_{\mathrm{dig}_{\mathbf{X}}}$ the kernel on $(\mathbf{X} \times [0,1]^2) \cdot [0,1] \times (\mathbf{X} \times [0,1]) \cdot [0,1]$ defined by

$$\kappa^*_{\mathrm{d}_{\mathbf{X}}}(x, e, e'; \dot{x}', \dot{f}) \cdot (e''; \dot{f}') = \mathbf{1}(x, \dot{x}') \mathbf{1}(e, \dot{f}) \mathbf{1}(\varphi(e', e''), \dot{f}'),$$

and $\kappa_{\mathrm{d}_{\mathbf{Y}}}$ is the kernel on $(\mathbf{Y} \times [0,1]) \cdot [0,1] \times (\mathbf{Y} \times [0,1]^2) \cdot [0,1]$ defined by

$$\kappa_{\mathrm{d}_{\mathbf{Y}}}(x, e; \dot{x}', \dot{f}, \dot{f}') \cdot (e'; \dot{f}'') = \mathbf{1}(x, \dot{x}') \mathbf{1}(e, \dot{f}) \mathbf{1}(e', \varphi(\dot{f}', \dot{f}'')),$$

where φ is a fixed bijection between $[0,1]$ and $[0,1]^2$.

Which proves that $!\mathfrak{a} :: \mathfrak{dig}_{\mathbf{X} \to \mathbf{Y}} = !!\mathfrak{a}$ (up to the isomorphism φ between $[0,1]^2$ and $[0,1]$ on the stateset).

This ends the proof: since $!\mathbb{A}$ is generated by the elements of the form $!\mathfrak{a}$, the fact that $\mathfrak{dig}_{\mathbf{X} \to \mathbf{Y}}$ maps every element of the form $!\mathfrak{a}$ to an element of $!!\mathbb{A}$ suffices to establish that it belongs to $!\mathbb{A} \multimap !!\mathbb{A}$, by Claim 69. □

Proposition 71. *The dereliction rule can be interpreted.*

Proof of Proposition 71. Now, dereliction is a map $!A \multimap A$. Suppose $\mathbb{A}$ is of support $\mathbf{X} \to \mathbf{Y}$. This map is easily implemented as a project $\mathfrak{der} = (\mathbf{1}, \kappa^*_{\mathrm{der}_{\mathbf{X}}} + \kappa_{\mathrm{der}_{\mathbf{Y}}})$ with $\kappa^*_{\mathrm{der}_{\mathbf{X}}}$ the kernel on $(\mathbf{X}) \cdot [0,1] \times \mathbf{X} \times [0,1] \cdot [0,1]$ defined by:

$$\kappa^*_{\mathrm{der}_{\mathbf{X}}}(x; \dot{x}', \dot{f}) \cdot (e, \dot{f}) = \mathbf{1}(x, x') \mathbf{1}(e, \varphi(f, f)),$$

and $\kappa_{\mathrm{der}_{\mathbf{Y}}}$ the kernel on $(\mathbf{Y} \times [0,1]) \cdot [0,1] \times \mathbf{Y} \cdot [0,1]$ defined by:

$$\kappa_{\mathrm{der}_{\mathbf{Y}}}(x, e; \dot{x}') \cdot (\dot{e}'; \dot{f}) = \mathbf{1}(x, x') \mathbf{1}(\varphi(e, e'), f),$$

where φ is a fixed bijection between $[0,1]$ and $[0,1]^2$.

To show that this implements dereliction is a simple computation. Taking $\mathfrak{a}$ a balanced project, we consider $!\mathfrak{a}$, and show that $!\mathfrak{a} :: \mathfrak{der}_{\mathbf{X} \to \mathbf{Y}} = \mathfrak{a}$ up to some bijection on the stateset. Now, we compute $!\mathfrak{a} :: \mathfrak{der}$ with $!\mathfrak{a} = (\mathbf{1}, !\kappa)$. Again, given the

definition of $\mathfrak{Det}$, this consists in computing paths of length 3. This shows that $!\mathfrak{a} :: \mathfrak{Det}_{\mathbf{X}\to\mathbf{Y}} = \mathfrak{a}$ up to the isomorphism between $[0,1]^3$ and $[0,1]^2$ defined by $(a,b,c) \mapsto (\varphi(a,c),b)$.

Again, by Claim 69 this is enough to establish that $\mathfrak{Det}_{\mathbf{X}\to\mathbf{Y}}$ belongs to $!\mathbb{A} \multimap \mathbb{A}$. □

Proposition 72. *The functorial promotion can be interpreted in the model.*

Based on this construction and usual methods for interpreting multiplicatives, additives and quantifiers [17], we obtain another model of LL^2.

Theorem 73. *General proof-objects and types define a sound model of* LL^2.

VI. Conclusion and Perspectives

We have established that sub-Markov processes do provide a model of Linear Logic. It should be clear that probabilistic languages with sampling instructions can be interpreted in this model. Indeed, while axiom rules are interpreted by the identity kernel – i.e. the Dirac delta function –, generalised rules introducing non-trivial Markov kernels can very well be accommodated in our model. We expect strong connections with game semantics models dealing with such languages [33], although our approach differs from the start by its intention. In particular, types arise from the behaviour of processes and are therefore not predefined. Moreover, the typing discipline is very rich, and can incorporate dependent types [16], [17], [13], and could be used to consider new type constructions adapted to probabilistic computation [34].

It is also worth noting that the formal relation with zeta function could turn out to be of a great interest with respect to the recasting of complexity theory by means of Interaction Graphs models [35], [24]. Indeed, it is hoped that invariants from dynamical systems (and the group/monoid action used to restrict graphings) to be related to the expressivity of the models, and I already showed with Pellissier how strong algebraic lower bounds can be expressed and strengthened using the notion of *topological entropy* [25]. This work thus provides an additional element with respect to these ideas, as the orthogonality, which is used to characterise the complexity classes, is here shown to be related to the zeta function of the underlying dynamical systems.

Lastly, it is important to stress that while the consideration of sub-kernels is important here, the typing discipline may be used to ensure that composition of well-typed Markov kernels are Markov kernels. Indeed, we believe that one can extend the definition of the the zeta function to define a function $\zeta^{\infty}_{\kappa}(z)$ that incorporates infinite orbits. We expect then to show that this modified zeta function will still satisfy the properties needed to build models of linear logic, and show that $\zeta^{\infty}_{\kappa\bullet\kappa'}(z) = 0$ implies that the execution $\kappa :: \kappa'$ is a Markov kernel (and not a sub-Markov kernel) as long as κ and κ' are.

References

[1] D. Scott, "Outline of a mathematical theory of computation," OUCL, Tech. Rep. PRG02, November 1970.

[2] G. Berry, "Stable models of typed λ-calculi," in *International Colloquium on Automata, Languages, and Programming*. Springer, 1978, pp. 72–89.

[3] J.-Y. Girard, "The system f of variable types, fifteen years later," *Theoretical Computer Science*, vol. 45, pp. 159 – 192, 1986. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0304397586900447

[4] ——, "Normal functors, power series and λ-calculus," *Annals of Pure and Applied Logic*, vol. 37, no. 2, pp. 129–177, 1988.

[5] ——, "Linear logic," *Theoretical Computer Science*, vol. 50, no. 1, pp. 1–101, 1987.

[6] ——, "Towards a geometry of interaction," in *Proceedings of the AMS Conference on Categories, Logic and Computer Science*, 1989.

[7] J. M. E. Hyland and C.-H. L. Ong, "On full abstraction for PCF: I, II, and III," *Information and Computation*, vol. 163, no. 2, pp. 285–408, 2000.

[8] S. Abramsky, R. Jagadeesan, and P. Malacaria, "Full abstraction for PCF," in *TACS '94: Proceedings of the International Conference on Theoretical Aspects of Computer Software*, ser. Lecture Notes in Computer Science, no. 789. London, UK: Springer-Verlag, 1994, pp. 1–15.

[9] C. Riba, "Strong normalization as safe interaction," in *LICS 2007 Proceedings*, 2007.

[10] J.-Y. Girard, "Geometry of interaction I: Interpretation of system F," in *In Proc. Logic Colloquium 88*, 1989.

[11] ——, "Geometry of interaction II: Deadlock-free algorithms," in *Proceedings of COLOG*, ser. Lecture Notes in Computer Science, no. 417. Springer, 1988, pp. 76–93.

[12] ——, "Geometry of interaction III: Accommodating the additives," in *Advances in Linear Logic*, ser. Lecture Notes Series, no. 222. Cambridge University Press, 1995, pp. 329–389.

[13] ——, "Geometry of interaction V: Logic in the hyperfinite factor." *Theoretical Computer Science*, vol. 412, pp. 1860–1883, 2011.

[14] T. Seiller, "Interaction graphs: Multiplicatives," *Annals of Pure and Applied Logic*, vol. 163, pp. 1808–1837, December 2012.

[15] ——, "Interaction graphs: Additives," *Annals of Pure and Applied Logic*, vol. 167, pp. 95 – 154, 2016.

[16] ——, "Interaction graphs: Graphings," *Annals of Pure and Applied Logic*, vol. 168, no. 2, pp. 278–320, 2017.

[17] ——, "Interaction graphs: Full linear logic," in *IEEE/ACM Logic in Computer Science (LICS)*, 2016. [Online]. Available: http://arxiv.org/pdf/1504.04152

[18] A. Naibo, M. Petrolo, and T. Seiller, "On the computational meaning of axioms," in *Epistemology, Knowledge and the Impact of Interaction*. Springer, 2016, pp. 141–184.

[19] J.-L. Krivine, "Typed lambda-calculus in classical zermelo-fraenkel set theory," *Arch. Mathematical Logic*, vol. 40, 2001.

[20] M. Hyland and A. Schalk, "Glueing and orthogonality for models of linear logic," *Theoretical Computer Science*, vol. 294, 2003.

[21] A. Terras, *Zeta functions of graphs: a stroll through the garden*. Cambridge University Press, 2010, vol. 128.

[22] B. Fuglede and R. V. Kadison, "Determinant theory in finite factors," *Annals of Mathematics*, vol. 56, no. 2, 1952.

[23] T. Seiller, "Logique dans le facteur hyperfini : géometrie de l'interaction et complexité," Ph.D. dissertation, Université Aix-Marseille, 2012. [Online]. Available: http://tel.archives-ouvertes.fr/tel-00768403/

[24] ——, "Interaction graphs: Nondeterministic automata," *ACM Transaction in Computational Logic*, vol. 19, no. 3, 2018.

[25] L. Pellissier and T. Seiller, "Prams over integers do not compute maxflow efficiently," 2021, https://hal.archives-ouvertes.fr/hal-01921942.

[26] T. Seiller, "Interaction Graphs: Exponentials," *Logical Methods in Computer Science*, vol. Volume 15, Issue 3, Aug. 2019. [Online]. Available: https://lmcs.episciences.org/5730

[27] D. Ruelle, "Zeta-functions for expanding maps and anosov flows," *Inventiones mathematicae*, vol. 34, no. 3, pp. 231–242, 1976.

[28] M. Artin and B. Mazur, "On periodic points," *Annals of Mathematics*, pp. 82–99, 1965.

[29] P. Panangaden, "The category of markov kernels," *Electronic Notes in Theoretical Computer Science*, vol. 22, pp. 171 – 187, 1999, pROBMIV'98, First International Workshop on Probabilistic Methods in Verification. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066105806024

[30] A. Joyal, R. Street, and D. Verity, "Traced monoidal categories," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 119, no. 3, pp. 447–468, 1996.

[31] M. Hasegawa, "Recursion from cyclic sharing: Traced monoidal categories and models of cyclic lambda calculi." Springer Verlag, 1997, pp. 196–213.
[32] E. Haghverdi and P. Scott, "A categorical model for the geometry of interaction," *Theoretical Computer Science*, vol. 350, no. 2, pp. 252–274, 2006.
[33] S. Castellan and H. Paquet, "Probabilistic programming inference via intensional semantics," in *Programming Languages and Systems - 28th European Symposium on Programming, ESOP 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, ser. Lecture Notes in Computer Science, L. Caires, Ed., vol. 11423. Springer, 2019, pp. 322–349. [Online]. Available: https://doi.org/10.1007/978-3-030-17184-1_12
[34] A. Naibo, M. Petrolo, and T. Seiller, "Logical constants from a computational point of view," *in preparation*.
[35] T. Seiller, "Towards a *Complexity-through-Realizability* theory," 2015, http://arxiv.org/pdf/1502.01257.

APPENDIX

Proof of Theorem 23. Clearly, a partial dynamical system $(\mathbf{X}, V, \Phi)$ where Φ is a NSMP map defines a graphing of support V with a single edge realised by Φ.

Now, let us explain how a deterministic graphing G defines a partial non-singular measurable-preserving dynamical system. Since G it is deterministic, we can consider a representative $\bar{G}$ of G such that the set

$$\left\{x \in \mathbf{X} \mid \exists e, f \in E^G, e \neq f \text{ and } x \in S_e^G \cap S_f^G\right\}$$

is the empty set. Then, one defines the partial dynamical system $(\mathbf{X}, \cup_{e\in E^{\bar{G}}} S_e^{\bar{G}}, \Phi)$, where:

$$\Phi(x) = \begin{cases} \phi_e^{\bar{G}}(x) \text{ if } x \in S_e^{\bar{G}} \\ 0 \text{ otherwise} \end{cases}$$

Moreover the map Φ is NSMP as a (disjoint) union of partial NSMP maps.

To end the proof, we need to show that the choice of representative in the previous construction is irrelevant. We prove this by showing that G is equivalent to the graphing H induced by $(\mathbf{X}, \cup_{e\in E^{\bar{G}}} S_e^{\bar{G}}, \Phi)$. But this is obvious, as $\bar{G}$ is a refinement of H, and G and $\bar{G}$ are equivalent. This is sufficient because of the following claim: if G and G' are equivalent, then the induced partial dynamical systems are a.e. equal. □

Proof of Lemma 24. A *deterministic graphing* F satisfies that for every edges $e, f \in E^F$, $S_e^F \cap S_f^F$ is of null measure. Suppose that the graphing $F :: G$ is not deterministic. Then there exists a Borel B of non-zero measure and two edges $e, f \in E^{F::G}$ such that $B \subset S_e^{F::G} \cap S_f^{F::G}$. The edges e, f correspond to paths π_e and π_f alternating between F and G. It is clear that the first step of these paths belong to the same graphing, say F without loss of generality, because the Borel set B did not belong to the *cut*. Thus π_e and π_f can be written $\pi_e = f_0\pi_e^1$ and $\pi_f = f_0\pi_f^1$. Thus the domains of the paths π_e^1 and π_f^1 coincide on the Borel set $\phi_{f_0}^F(B)$ which is of non-zero measure since all maps considered are non-singular. One can then continue the reasoning up to the end of one of the paths and show that they are equal up to this point. Now, if one of the paths ends before the other we have a contradiction because it would mean that the Borel set under consideration would be at the same time inside and outside the cut, which is not possible. So both paths have the same length and are therefore equal. Which shows that $F :: G$ is deterministic since we have shown that if the domain of two paths alternating between F and G coincide on a non-zero measure Borel set, the two paths are equal (hence they correspond to the same edge in $F :: G$). □

Proof of Proposition 26. On one hand, we have

$$-\log(\zeta_{g\circ f.1}(1)) = \sum_{m\geqslant 1} \int_{\mathrm{Fix}((g\circ f)^m)} \frac{1}{m} \sum_{m\geqslant 1} \frac{\mu(\mathrm{Fix}((g\circ f)^m))}{m}.$$

On the other hand, the measurement $[\![f, g]\!]_m$ defined on general graphings [16, Definitions 37 and 57] is given by the formula shown in Figure 2 in which ρ_ϕ is a measurable map associating to each point the length of the orbit it belongs to [14, Corollary 45], $\mathcal{P}[f, g]$ denotes the set of prime closed paths alternating between f and g, and generally $h_*\mu$ denotes the pullback measure of μ along h.

As established by Seiller, this expression simplifies in the measure-preserving case [14, Proposition 52], and can be expressed as

$$[\![f, g]\!]_m = \sum_{\pi=e_0\dots e_n\in\mathcal{P}[f,g]} \int_{\mathrm{supp}(\pi)} \frac{m(\omega(\pi)^{\rho_{\phi_\pi}(x)})}{\rho_{\phi_\pi}(x)}$$

Now, we can split this expression by considering the partition of $\mathrm{supp}(\pi)$ given by the preimage of ρ_ϕ. I.e. this partitions $\mathrm{supp}(\pi)$ into (measurable) subsets $S_i^\pi = \rho_\phi^{-1}(\mathrm{supp}(\pi))$ containing the points $x \in \mathrm{supp}(\pi)$ such that the orbit of x is of length i.

As the value of ρ_ϕ is constant on these sets, this gives:

$$[\![f, g]\!]_m = \sum_{\pi=e_0\dots e_n\in\mathcal{P}[f,g]} \sum_{i=0}^{\infty} \int_{S_i^\pi} \frac{m(\omega(\pi)^i)}{i}$$

Now, we are considering the case where $m(x) = z$, and we know all weights in the graphing are equal to 1. Hence:

$$[\![f, g]\!]_m = \sum_{\pi\in\mathcal{P}[f,g]} \sum_{i=0}^{\infty} \int_{S_i^\pi} \frac{z}{i}.$$

On the other hand, we have that, writing $\mathrm{AltCycle}(f, g)_m$ the set of all alternating cycle between f and g of length m:

$$\begin{aligned}\log(\zeta_{g\circ f,1}(z)) &= \sum_{m\geq 1} \int_{\mathrm{Fix}((g\circ f)^m)} \frac{z}{m} \\ &= \sum_{m\geq 1} \sum_{\pi\in\mathrm{AltCycle}(F,G)_m} \int_{S_m^\pi} \frac{z}{m}\end{aligned}$$

since each fixpoint belongs to exactly one alternating cycle of length m between f and g (because the graphings are deterministic).

Now each alternating cycle of length m between f and g can be written uniquely as a product of alternating prime cycles, we deduce (this is essentially Proposition 60 in Seiller's first paper on Interaction Graphs [14]):

$$\begin{aligned}\log(\zeta_{g\circ f,1}(z)) &= \sum_{m\geq 1} \sum_{\pi\in\mathcal{P}[f,g]} \int_{S_m^\pi} \frac{z}{m} \\ &= \sum_{\pi\in\mathcal{P}[f,g]} \sum_{m\geq 1} \int_{S_m^\pi} \frac{z}{m} \\ &= [\![f, g]\!]_m\end{aligned}$$

□

Proof of Theorem 28. If the weights of edges in F and G are elements of $[0, 1]$, then it is clear that the weights of edges in $F :: G$ are also elements of $[0, 1]$. We therefore only need to check that the second condition is preserved.

Let us denote by $\mathrm{Out}(F :: G)$ the set of $x \in X$ which are source of paths whose added weight is greater than 1, and by $\mathrm{Out}(F \cup G)$ the set of x which are source of edges (either in F or G) whose added weight is greater than 1. First, we

$$\sum_{\pi=e_0e_1\dots e_n\in\mathcal{P}[f,g]}\sum_{j=0}^{n}\int_{\mathrm{supp}(\pi)}\sum_{k=0}^{\rho_{\phi_\pi}(x)-1}\frac{m(\omega(\pi)^{\rho_{\phi\pi}(\phi_\pi^k(x))})}{(n+1)\rho_{\phi_\pi}(x)\rho_{\phi_\pi}(\phi_\pi^k(x))}d(\phi_{e_n}\circ\phi_{e_{n-1}}\circ\cdots\circ\phi_{e_j})_*\lambda(x)$$

Fig. 2. General measurement function on graphings.

notice that if $x\in\mathrm{Out}(F::G)$ then either $x\in\mathrm{Out}(F\cup G)$, or x is mapped – through at least one edge – to an element y which is itself in $\mathrm{Out}(F\cup G)$. To prove this statement, let us write $\mathrm{paths}(x)$ (resp. $\mathrm{edges}(x)$) the set of paths in $F::G$ (resp. edges in F or G) whose source contain x. We know the sum of all the weights of these paths is greater than 1, i.e. $\sum_{\pi\in\mathrm{paths}(x)}\omega(\pi)>1$. But this sum can be rearranged by ordering paths depending on theirs initial edge, i.e. $\sum_{\pi\in\mathrm{paths}(x)}\omega(\pi)=\sum_{e\in\mathrm{edges}(x)}\sum_{\pi=e\rho\in\mathrm{paths}(x)^e}\omega(\pi)$, where $\mathrm{paths}(x)^e$ denotes the paths whose first edge is e. Now, since the weight of e appears in all $\omega(e\rho)=\omega(e)\omega(\rho)$, we can factorize and obtain the following inequality.

$$\sum_{e\in\mathrm{edges}(x)}\omega(e)\left(\sum_{\pi=e\rho\in\mathrm{paths}(x)^e}\omega(\rho)\right)>1$$

Since the sum $\sum_{e\in\mathrm{edges}(x)}\omega(e)$ is not greater than 1, we deduce that there exists at least one $e\in\mathrm{edges}(x)$ such that $\sum_{\pi=e\rho\in\mathrm{paths}(x)^e}\omega(\rho)>1$. However, this means that $\phi_e(x)$ is an element of $\mathrm{Out}(F::G)$.

Now, we must note that x is not element of a closed path. This is clear from the fact that x lies in the carrier of $F::G$.

Then, an induction shows that x is an element of $\mathrm{Out}(F::G)$ if and only if there is a (finite, possibly empty) path from x to an element of $\mathrm{Out}(F\cup G)$, i.e. $\mathrm{Out}(F::G)$ is at most a countable union of images of the set $\mathrm{Out}(F\cup G)$. But since all maps considered are non-singular, these images of $\mathrm{Out}(F\cup G)$ are negligible subsets since $\mathrm{Out}(F\cup G)$ is itself negligible. This ends the proof as a countable union of copies of negligible sets are negligible (by countable additivity), hence $\mathrm{Out}(F::G)$ is negligible. □

Proof of Lemma 44. The fact that the Markov kernels are in general position allows us to write the composition in a traced monoidal category style (Figure 3).

The above theorem then states that the feedbacks commute with the composition. I.e. that Figure 4 computes the same kernel as the one below which represent the left-hand side of the equation.

The fact that this is true is a consequence of the fact that kernels are in general position since the integrals are taken over disjoint domains. The underlying explanation is that the execution kernel $\kappa::\kappa'$ is computed by integrating over *alternating paths* between κ and κ', i.e. it is computed as an integral over the sequences $x_1,x_2,\dots,x_k$ of the alternating product of $\kappa(x_i,\dot{x}_{i+1})$ and $\kappa'(x_{i+1},\dot{x}_{i+2})$. These paths can be seen in the above figures. Taking the iterated composition $(\kappa_0::\kappa_1)::\kappa_2$ thus integrates over alternating paths between κ_2 and alternating paths between κ_0 and κ_1. As in the case of graphs [14], this can be shown to be the same as integrating over all alternating paths between κ_0 and alternating paths between κ_1 and κ_2. □

Proof of Proposition 47. The proof consists in heavy computations, but without any technical difficulties. The main ingredient is again the geometric adjunction. The pictures shown in the proof of 44 can be used here to have better insights on the situation. The zeta function quantifies the finite orbits, i.e. the proportion of points that can be reached from themselves by alternating iterations of the involved kernels (weighted by the probabilities of such dynamics occurring). The main ingredient of the proof is then that a closed path alternating between F, G, and H is either a closed path alternating between F and G, or a closed path alternating between H and alternating paths between F and G. Since the roles of F, G and H are symmetric in this statement, we obtain three different splittings of the initial set of closed paths. Now, since zeta functions measure sets of closed paths, these three equal but different expressions yield three different products of two zeta functions. The statement above simply corresponds to stating the equality of two of those. □

Proof of Proposition 70. Suppose $\mathbb{A}$ is of support $\mathbf{X}\to\mathbf{Y}$. This map is easily implemented as a project $\mathfrak{dig}_{\mathbf{X}\to\mathbf{Y}}=(\mathbf{1},\kappa_{\mathrm{dig}_{\mathbf{Y}}}+\kappa^*_{\mathrm{dig}_{\mathbf{X}}})$ with

$$\begin{array}{l}\kappa^*_{\mathrm{dig}_{\mathbf{X}}}:\\ (\mathbf{X}\times[0,1]\times[0,1])\cdot[0,1]\times(\mathbf{X}\times[0,1])\cdot[0,1]\to[0,1]\\ (x,e,e')\cdot e'',(x',f)\cdot f'\mapsto\mathbf{1}(x,x')\mathbf{1}(e,f)\mathbf{1}(\varphi(e',e''),f')\end{array}$$

$$\begin{array}{l}\kappa_{\mathrm{dig}_{\mathbf{Y}}}:\\ (\mathbf{Y}\times[0,1])\cdot[0,1]\times(\mathbf{Y}\times[0,1]\times[0,1])\cdot[0,1]\to[0,1]\\ (x,e)\cdot e',(x',f,f')\cdot f''\mapsto\mathbf{1}(x,x')\mathbf{1}(e,f)\mathbf{1}(e',\varphi(f',f''))\end{array}$$

where φ is a fixed bijection between $[0,1]$ and $[0,1]^2$.

To show that this implements digging is a simple exercise. Taking $\mathfrak{a}$ a balanced project, we consider $!\mathfrak{a}$, and show that $!\mathfrak{a}::\mathfrak{dig}_{\mathbf{X}\to\mathbf{Y}}=!!\mathfrak{a}$. In fact, it is easy to convince oneself that the kernel of $!\mathfrak{a}::\mathfrak{dig}_{\mathbf{X}\to\mathbf{Y}}$ is computed as the set of "length

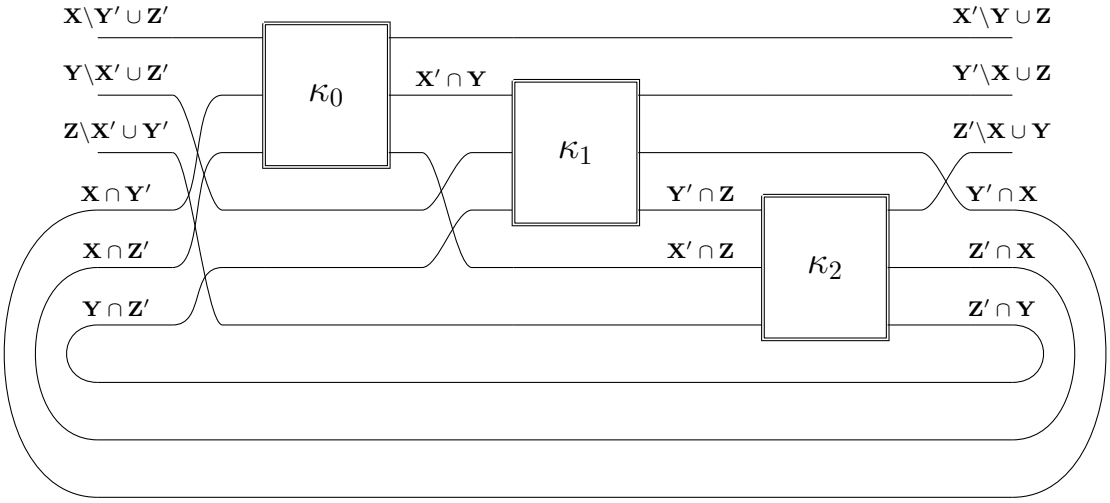


Fig. 3. Proof of Lemma 44, first figure

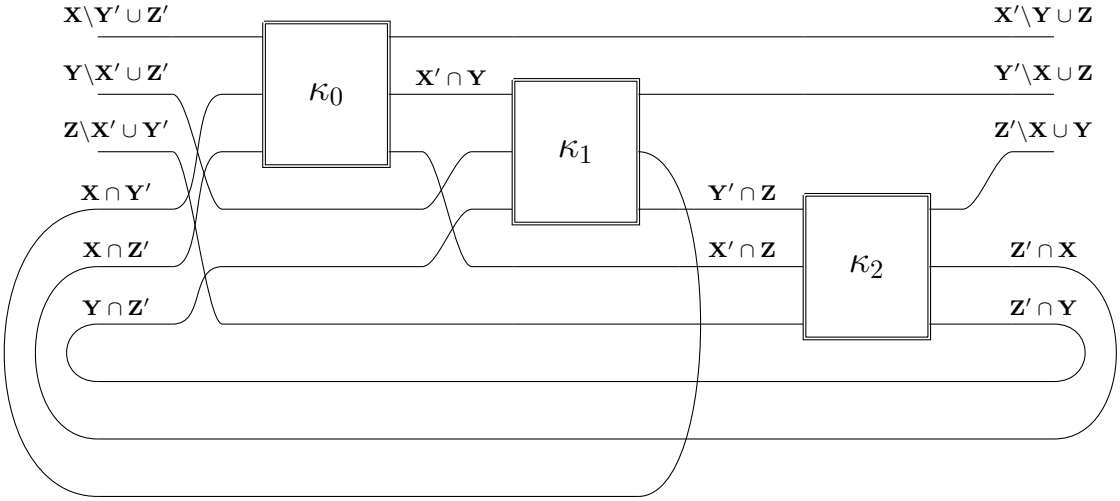


Fig. 4. Proof of Lemma 44, second figure

3 paths". i.e.

$$
\begin{aligned}
&\mathrm{tr}(!\kappa_{\mathrm{A}}^{\dagger} \bullet (\kappa_{\mathrm{dig}_{\mathbf{Y}}} + \kappa_{\mathrm{dig}_{\mathbf{X}}}^{*})^{\ddagger})((x,e,f);(\dot{x},\dot{e},\dot{f}))\cdot(g,h;\dot{g},\dot{h}) \\
&= \int_{(y,u,d,d'),(z,v,c,c')} \left[\begin{array}{l} \kappa_{\mathrm{dig}_{\mathbf{X}}}^{*}((x,e,f),(\dot{y},\dot{u}))\cdot(g,\dot{d})\mathbf{1}(h,\dot{d'}) \\ \quad\times !\kappa_{\mathrm{A}}((y,u);(\dot{z},\dot{v}))\cdot(d',\dot{c'})\mathbf{1}(d,\dot{c}) \\ \quad\quad\times\kappa_{\mathrm{dig}_{\mathbf{Y}}}((z,v),(\dot{x},\dot{e},\dot{f}))\cdot(c,\dot{g})\mathbf{1}(c',\dot{h}) \end{array}\right] \\
&= \int_{(y,u,d,d')} \left[\begin{array}{l} \mathbf{1}(x,\dot{y})\mathbf{1}(e,\dot{u})\mathbf{1}(\varphi(f,g),\dot{d})\mathbf{1}(h,\dot{d'}) \\ \quad\times\int_{(z,v,c,c')}\left[\begin{array}{l} !\kappa_{\mathrm{A}}((y,u);(\dot{z},\dot{v}))\cdot(d',\dot{c'})\mathbf{1}(d,\dot{c}) \\ \quad\quad\times\kappa_{\mathrm{dig}}((z,v),(\dot{x},\dot{e},\dot{f}))\cdot(c,\dot{g})\mathbf{1}(c',\dot{h}) \end{array}\right] \end{array}\right] \\
&= \int_{(z,v,c,c')} \left[\begin{array}{l} !\kappa_{\mathrm{A}}((x,e);(\dot{z},\dot{v}))\cdot(h,\dot{c'})\mathbf{1}(\varphi(f,g),\dot{c}) \\ \quad\times\kappa_{\mathrm{dig}}((z,v),(\dot{x},\dot{e},\dot{f}))\cdot(c,\dot{g})\mathbf{1}(c',\dot{h}) \end{array}\right] \\
&= \int_{(z,v,c,c')} \left[\begin{array}{l} !\kappa_{\mathrm{A}}((x,e);(\dot{z},\dot{v}))\cdot(h,\dot{e})\mathbf{1}(\varphi(f,g),\dot{e'}) \\ \quad\times\mathbf{1}(z,\dot{x})\mathbf{1}(v,\dot{e})\mathbf{1}(c,\varphi(\dot{f},\dot{g}))\mathbf{1}(c',\dot{h}) \end{array}\right] \\
&= !\kappa_{\mathrm{A}}((x,e);(\dot{x},\dot{e}))\cdot(h,\dot{h})\mathbf{1}(\varphi(f,g),\varphi(\dot{f},\dot{g})) \\
&= !\kappa_{\mathrm{A}}((x,e);(\dot{x},\dot{e}))\cdot(h,\dot{h})\mathbf{1}(f,\dot{f})\mathbf{1}(g,\dot{g}) \\
&= \kappa_{\mathrm{A}}(x;\dot{x})\cdot(e;\dot{e})\mathbf{1}(h,\dot{h})\mathbf{1}(f,\dot{f})\mathbf{1}(g,\dot{g}) \\
&= \kappa_{\mathrm{A}}(x;\dot{x})\cdot(e;\dot{e})\mathbf{1}(f,\dot{f})\mathbf{1}(\varphi(h,g),\varphi(\dot{h},\dot{g}))
\end{aligned}
$$

Which proves that $!\mathfrak{a} :: \mathfrak{dig}_{\mathbf{X}\to\mathbf{Y}} = !!\mathfrak{a}$ (up to the isomorphism φ between $[0,1]^2$ and $[0,1]$ on the stateset).

This ends the proof: since $!\mathbb{A}$ is generated by the elements of the form $!\mathfrak{a}$, the fact that $\mathfrak{dig}_{\mathbf{X}\to\mathbf{Y}}$ maps every element of the form $!\mathfrak{a}$ to an element of $!!\mathbb{A}$ suffices to establish that it belongs to $!\mathbb{A} \multimap !!\mathbb{A}$, by Claim 69. □

Proof of Proposition 71. Now, dereliction is a map $!A \multimap A$. Suppose $\mathbb{A}$ is of support $\mathbf{X} \to \mathbf{Y}$. This map is easily implemented as a project $\mathfrak{der} = (\mathbf{1}, \kappa_{\mathrm{der}_{\mathbf{X}}}^{*} + \kappa_{\mathrm{der}_{\mathbf{Y}}})$ with:

$\kappa_{\mathrm{der}_{\mathbf{X}}}^{*}$:
$(\mathbf{X})\cdot[0,1]\times\mathbf{X}\times[0,1]\cdot[0,1] \to [0,1]$
$x\cdot e,(x',f)\cdot f' \mapsto \mathbf{1}(x,x')\mathbf{1}(e,\varphi(f,f))$

$$\begin{array}{l}\kappa_{\mathrm{der}_{\mathbf{Y}}}:\\ (\mathbf{Y}\times[0,1])\cdot[0,1]\times\mathbf{Y}\cdot[0,1]\to[0,1]\\ (x,e)\cdot e',x'\cdot f\mapsto\mathbf{1}(x,x')\mathbf{1}(\varphi(e,e'),f)\end{array}$$

where φ is a fixed bijection between $[0,1]$ and $[0,1]^2$.

To show that this implements dereliction is a simple computation. Taking $\mathfrak{a}$ a balanced project, we consider $!\mathfrak{a}$, and show that $!\mathfrak{a}::\mathfrak{der}_{\mathbf{X}\to\mathbf{Y}}=\mathfrak{a}$ up to some bijection on the stateset. Now, we compute $!\mathfrak{a}::\mathfrak{der}$ with $!\mathfrak{a}=(\mathbf{1},!\kappa)$. Again, given the definition of $\mathfrak{der}$, this consists in computing paths of length 3, i.e.

$$\begin{array}{rl}&\mathrm{tr}(!\kappa_{\mathrm{A}}^{\dagger}\bullet(\kappa_{\mathrm{der}_{\mathbf{X}}}^{*}+\kappa_{\mathrm{der}_{\mathbf{Y}}})^{\ddagger})(x;\dot{x})\cdot(e,f;\dot{e};\dot{f})\\ =&\displaystyle\int_{(y,a,b,c)}\int_{(z,u,v,w)}\\ &\left[\begin{array}{r}\kappa_{\mathrm{der}_{\mathbf{X}}}^{*}(x;(\dot{y},\dot{a}))\cdot(e;\dot{b})\mathbf{1}(f,\dot{c})\\ \times!\kappa_{\mathrm{A}}((y,a);(\dot{z},\dot{u}))\cdot(c;\dot{w})\mathbf{1}(b;\dot{v})\\ \times\kappa_{\mathrm{der}_{\mathbf{Y}}}((z,u);\dot{x})\cdot(v;\dot{e})\mathbf{1}(w,\dot{f})\end{array}\right]\\ =&\displaystyle\int_{(y,a,b,c)}\mathbf{1}(x;\dot{y})\mathbf{1}(e;\varphi(\dot{a},\dot{b}))\mathbf{1}(f,\dot{c})\\ &\displaystyle\int_{(z,u,v,w)}\left[\begin{array}{r}!\kappa_{\mathrm{A}}((y,a);(\dot{z},\dot{u}))\cdot(c;\dot{w})\mathbf{1}(b;\dot{v})\\ \times\kappa_{\mathrm{der}_{\mathbf{Y}}}((z,u);\dot{x})\cdot(v;\dot{e})\mathbf{1}(w,\dot{f})\end{array}\right]\\ =&\displaystyle\int_{(z,u,v,w)}\left[\begin{array}{r}!\kappa_{\mathrm{A}}((x,\varphi_0^{-1}(e));(\dot{z},\dot{u}))\cdot(f;\dot{w})\mathbf{1}(\varphi_1^{-1}(e);\dot{v})\\ \times\kappa_{\mathrm{der}_{\mathbf{Y}}}((z,u);\dot{x})\cdot(v;\dot{e})\mathbf{1}(w,\dot{f})\end{array}\right]\\ =&\displaystyle\int_{(z,u,v,w)}\left[\begin{array}{r}!\kappa_{\mathrm{A}}((x,\varphi_0^{-1}(e));(\dot{z},\dot{u}))\cdot(f;\dot{w})\mathbf{1}(\varphi_1^{-1}(e);\dot{v})\\ \times\mathbf{1}(z;\dot{x})\mathbf{1}(\varphi(u,v);\dot{e})\mathbf{1}(w,\dot{f})\end{array}\right]\\ =&!\kappa_{\mathrm{A}}((x,\varphi_0^{-1}(e));(\dot{x},\varphi_0^{-1}(\dot{u})))\cdot(f;\dot{f})\mathbf{1}(\varphi_1^{-1}(e);\varphi_1^{-1}(\dot{v}))\\ =&\kappa_{\mathrm{A}}(x;\dot{x})\cdot(\varphi_0^{-1}(e);\varphi_0^{-1}(\dot{e}))\mathbf{1}(f,\dot{f})\mathbf{1}(\varphi_1^{-1}(e);\varphi_1^{-1}(\dot{v}))\end{array}$$

This shows that $!\mathfrak{a}::\mathfrak{der}_{\mathbf{X}\to\mathbf{Y}}=\mathfrak{a}$ up to the isomorphism between $[0,1]^3$ and $[0,1]^2$ defined by $(a,b,c)\mapsto(\varphi(a,c),b)$.

Again, by Claim 69 this is enough to establish that $\mathfrak{der}_{\mathbf{X}\to\mathbf{Y}}$ belongs to $!\mathbb{A}\multimap\mathbb{A}$. □

Proof of Proposition 72. The proof is a tad more involved than the preceding ones. Here we will implement the rule in three steps. The principle is easy to understand: given $!\mathfrak{a}\in!\mathbb{A}$ and $!\mathfrak{f}\in!(\mathbb{A}\multimap\mathbb{B})$, we will first compute the executions $!\mathfrak{a}::\mathfrak{left}$ and $!\mathfrak{f}::\mathfrak{right}$ in order to ensure disjointness of the spaces used to encode the statesets of $\mathfrak{a}$ and $\mathfrak{f}$ respectively. Once this is done, the execution $(!\mathfrak{a}::\mathfrak{left})::(!\mathfrak{f}::\mathfrak{right})$ morally computes the same as $!\mathfrak{f}::\mathfrak{a}$ up to some transformation $\mathfrak{fit}$ that internalises a stateset isomorphism.

Let $\mathfrak{a}\in\mathbb{A}$ and $\mathfrak{f}\in\mathbb{A}\multimap\mathbb{B}$ be balanced proof-objects, of respective supports $\mathbf{X}\to\mathbf{Y}$ and $\mathbf{X}'\to\mathbf{Y}'$. We consider the proof-object $\mathfrak{twist}=(\mathbf{1},\kappa_{\mathrm{twist}})$ with:

$$\begin{array}{l}\kappa_{\mathrm{twist}}:\\ (\mathbf{X}\cup\mathbf{X}')\times[0,1]\cdot[0,1](\mathbf{Y}\cup\mathbf{Y}')\times[0,1]\cdot[0,1]\to[0,1]\\ (x,e)\cdot f(\dot{y},\dot{e})\cdot\dot{f}\mapsto\mathbf{1}(x,\dot{y})\mathbf{1}(e,\varphi_0^{-1}(\dot{f}))\mathbf{1}(f,\varphi(\dot{e},\varphi^{-1}(\dot{f}))\end{array}$$

for $x,\dot{y}\in\mathbf{Y}\cap\mathbf{X}$. We then use the kernels:

$$\begin{array}{l}\kappa_{\mathrm{l}}:\\ (\mathbf{X}\times[0,1])\cdot[0,1]\times(\mathbf{X}\times[0,1]\times[0,1])\cdot[0,1]\to[0,1]\\ (x,e)\cdot e',(x',f,f')\cdot f''\mapsto\mathbf{1}(x,x')\mathbf{1}(e,f)\mathbf{1}(e',\varphi(f',f''))\end{array}$$

$$\begin{array}{l}\kappa_{\mathrm{r}}:\\ (\mathbf{X}\times[0,1])\cdot[0,1]\times(\mathbf{X}\times[0,1]\times[0,1])\cdot[0,1]\to[0,1]\\ (x,e)\cdot e',(x',f,f')\cdot f''\mapsto\mathbf{1}(x,x')\mathbf{1}(e,f')\mathbf{1}(e',\varphi(f,f''))\end{array}$$

$$\begin{array}{l}\kappa_{\mathrm{c}}:\\ (\mathbf{X}\times[0,1]\times[0,1])\cdot[0,1]\times(\mathbf{X}\times[0,1])\cdot[0,1]\to[0,1]\\ (x,e,e')\cdot e'',(x',f)\cdot f'\mapsto\mathbf{1}(x,x')\mathbf{1}(\varphi(e,e'),f)\mathbf{1}(e',f')\end{array}$$

where φ is a fixed bijection between $[0,1]$ and $[0,1]^2$.

Consider projects $!\mathfrak{a}=(\mathbf{1},!\kappa_{\mathrm{A}})$ and $!\mathfrak{f}=(\mathbf{1},!\kappa_{\mathrm{F}})$ of support $\mathbf{X}$ and $\mathbf{X}\times\mathbf{Y}$ respectively. We show that $((!\kappa_{\mathrm{A}}::\kappa_{\mathrm{l}})::\kappa_{\mathrm{twist}}::(!\kappa_{\mathrm{F}}::\kappa_{\mathrm{r}}))::\kappa_{\mathrm{c}}$ is equal to $!\kappa_{\mathrm{A}::\mathrm{F}}$. □