Zeta Functions and the (Linear) Logic of Markov Processes

Thomas Seiller

Zeta Functions and the (Linear) Logic of Markov Processes

Thomas Seiller – seiller@lipn.fr
CNRS & University Sorbonne Paris North, France

February 24, 2020

Abstract

In a series of papers, the author 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 linear logic as a type system over the set of all (discrete-time, time-independent) Markov processes.

1 Introduction

Right after his seminal paper introducing Linear Logic [4], Girard proposed a research programme [5] aimed at obtaining new semantic models, which we chose here to call "dynamic semantics". One could understand the motivations for this programme as 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. In some sense, Girard's programme inspired the introduction of game semantics which provide such models, although they fail to fulfil the second Girard's expectation.

Indeed, while providing a mathematical account of proof normalisation was a major goal, it somehow clouded an even more appealing and ambitious objective, namely to use the Curry-Howard correspondence to provide foundations of mathematical logic based on computer science. As such, the models envisioned where to be constructed around a notion of program abstracted from types, the types being defined as a language to describe the behaviour of the considered programs. While the first such models were constructed around an

abstraction of programs as bounded operators on Hilbert spaces, recent work by the author reformulated the constructions by considering graphs or graph-like structures called *graphings* to account for programs. We will now describe the basic intuitions behind the construction of models in the simple case of graphs.

Graphs provide a minimal but natural mathematical structure to represent programs. Indeed, Turing machines and automata naturally abstract 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. E.g. 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 $G\square A$ – reduces to a graph $\mathrm{Ex}(G\square A)$ of type $\mathbb{B}$.

Obviously, the abstraction is not rich enough to adequately model expressive computational models such as Turing machines. Some information is lost by considering discrete graphs: following an edge corresponds to performing some instruction and therefore modifying the state of the machine. To regain expressivity, the author introduced graphings. Graphings are graphs which are realised on a topological or measure 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 `right`.

As we will show in the first sections, the restriction 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 – as we will show – the types are constructed based on some notion of *zeta function* for dynamical systems, related to Ihara and Ruelle works. The models, and particularly the fact that the notion of type inferred follow the linear logic discipline, then essentially relies on a cocycle relation relating iteration and the zeta function.

Moreover, if one allows to probabilities on edges, then the natural notion of *probabilistic graphings* coincide with a class of Markov processes that we will describe. The goal of the present work, and its main achievement, is to extend the model to the class of all Markov processes. More precisely, we define both a notion of execution and of zeta function for general sub-Markov kernels and show how these satisfy the cocycle relation that ensures that types do form a model of linear logic. This naturally provides models of typed lambda-calculus extended with probabilities and specific instructions for sampling specific distributions.

2 Interaction Graphs: Linear Logic and Zeta functions

In this section, we review the models of linear logic introduced by the author 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.

2.1 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 [25] 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.

Now, we consider a directed graph $G = (V^G, E^G, s^G, t^G)$ and suppose it is *simple*, i.e. that the map $E^G \mapsto V^G \times V^G; e \mapsto (s^G, t^G)$ is injective. We consider the set of closed paths (called circuits by the author in Interaction Graphs (IG) [17]), i.e. words $e_1, \ldots, e_k$ considered up to cyclic permutations. A *prime closed path* (called 1-circuits 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.

The following definition provides a clear parallel with the famous Euler zeta function.

Definition 1. The Bowen-Lanford zeta function associated with the graph G is defined as:

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

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 is done by relating the zeta function with the determinant of the adjacency matrix of G. Notice that this relation was obtained by the author in the special case $z = 1$ [17] and was the motivation behind the definition of orthogonality in Interaction Graphs models.

Lemma 2. *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 3. *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 4. *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 the author's earlier work [17, Lemma 61]. □

In fact, work on zeta functions of graphs showed a stronger result which uses the Perron-Frobenius theorem in case the adjacency matrix M of the graph is aperiodic, i.e. there exists N such that $M^N > 0$.

Theorem 5. *If A is aperiodic then A has simple positive eigenvalue $\lambda_1 > 0$ and all other eigenvalues verify $\max_{2 \leqslant i \leqslant k} |\lambda_i| < \lambda_1$.*

This leads to the theorem of Bowen and Lanford [2].

Theorem 6. *The zeta function $\zeta_G(z)$ is non-zero and analytic for $|z| < 1/\lambda_1$. Morover, it has a simple pole at $z = \lambda_1$ and a meromorphic extension to the complex plane $\mathbf{C}$ of the form $\zeta(z) = \det(1 - zA)^{-1}$. (In particular, it is a rational function.)*

This can be used to prove the prime graph theorem.

Theorem 7.

$$\mathit{Card}(\{\pi \in \mathcal{C}(G) \mid \lg(\pi) = n\}) \sim \frac{\lambda_1^n}{n} \text{ as } n \to \infty$$

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.

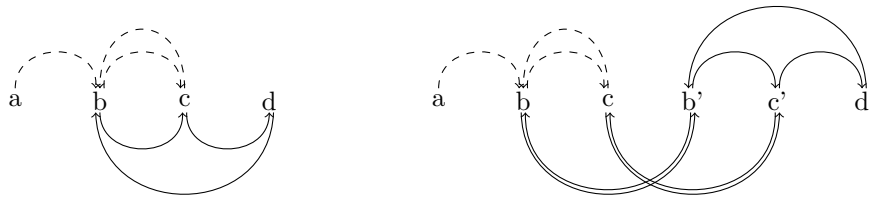


Figure 1: On the left: implicit cut between two graphs (one is plain, the other is dashed). On the right: explicit cut between the same two graphs (the cut is shown below). Both representations lead to the same result, as the cut-elimination is represented by *execution*, an operation defined from alternating paths. It is easily checked that there is a bijective correspondence between alternating paths on the left and alternating paths on the right (though it does not preserve the length).

2.2 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 graphs; here we will suppose weights are complex numbers. 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 implicitely 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.

Definition 8 (Alternating paths)**.** Let G and H be two graphs. We define the *alternating paths* between G and H as the paths (e_i) in $G \cup H$ which satisfy that for all $i = 0, \dots, k-1$, $e_i \in F$ if and only if $e_{i+1} \in G$. 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 in $G \square H$ with source and target in a given set of vertices V.

Definition 9. Let F and G be two graphs. The *execution* of F and G is the graph $F :: G$ defined by:

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

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 correspon-

dence. 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 the author [13] for more details and explanations.

Definition 10. We will call an *alternating closed path* a closed path $(e_i)_{0\leqslant i\leqslant n}$ such that for all $i \in \mathbf{Z}/n\mathbf{Z}$, $e_i \in F$ if and only if $e_{i+1} \in G$. The set of prime alternating closed paths between F and G is 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 [11], and related to Hyland and Schalk's double glueing construction [9]. We only recall the measurement here and refer to the first IG paper for more details [17].

Definition 11. For any two graphs F, G, we define the measurement

$$[\![F, G]\!]_m = \sum_{\pi\in\mathcal{C}(F,G)} -log(1 - \omega^{F\Box G}(\pi)).$$

As the reader might already have noticed, there is a similarity between the measurement defined by the author and the Bowen-Lanford zeta function. To formalise the connection, we will consider zeta functions of weighted graphs.

2.3 Zeta functions of weighted directed graphs

Now, we consider weighted directed graphs, i.e. graphs with weights of the edges. For now, consider weights to be complex numbers. We write ω the weight function, as well as its extension to paths, using the product, i.e. $\omega(\pi) = \prod_{e\in\pi} \omega(e)$.

Now, the zeta function of the weighted graph is defined as follows.

Definition 12. The zeta function associated with the weighted graph G is defined as:

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

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 the author's combinatorial interpretation of the determinant $\det(1 - M(G))$ [17, Corollary 61.1].

Proposition 13. *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 $|z|$.

Notations 14. In this paper, given two sets A, B, we write $A\backslash B$ the set $\{a \in A \mid a \notin B\}$.

Taking the logarithm we obtain:

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

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 with the value of the zeta function at $z = 1$:

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

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

Now, we recall that orthogonality in IG models is defined by $F \perp G$ as $[\![F, G]\!]_{\log(1-x)} \neq 0, \infty$, 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 abstract the construction of IG models and provide an even richer framework based on zeta functions.

2.4 Zeta, Execution and a Cocycle Property

There are two main properties used to define IG models. 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* [20]:

$$[\![F, G :: H]\!]_m + [\![G, H]\!]_m = [\![G, H :: F]\!]_m + [\![H, F]\!]_m.$$

The identity is in fact obtained from the so-called geometric trefoil property [18, 20], satisfied for graphs F, G, H such that $V^F \cap V^G \cap V^H = \emptyset$:

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

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 $[\![F, G]\!]_{\log(1-x)} = -\log(\zeta_{F \bullet G}(1))$, the trefoil property is a straightforward consequence of the following theorem for $z = 1$.

Theorem 15. *Suppose $V^F \cap V^G \cap V^H = \emptyset$. Then:*

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

Proof. By definition and the geometric trefoil property:

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

□

We can then define families of models of linear logic extending the author's previous 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 16. 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 17. Given two proof objects $\mathfrak{g} = (g, G)$ and $\mathfrak{h} = (h, H)$ we define the *zeta-measurement* as the complex function:

$$\zeta_{\mathfrak{g},\mathfrak{h}} : z \mapsto g(z).h(z).\zeta_{G,H}(z),$$

where the dots denotes pointwise multiplication of functions.

Definition 18. 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 $\zeta_{\mathfrak{g},\mathfrak{h}} \in P$.

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 [17] 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 19. 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 [17]. 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 [20]. 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.

3 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 the notion of graphing. Graphings are equivalence classes of graph-like objects called graphing representatives. The definition is usually parametrised by a monoid action $M \curvearrowright \mathbf{X}$ on the underlying space $\mathbf{X}$, but the choice of the monoid action will not be important in this paper.

Definition 20. 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.

Graphings 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 action will be *continuous*. If $\mathbf{X}$ is a measure space, which will be the case considered here, the action will be *measurable*. More precisely, for technical purposes, the action is be by NSMP maps, i.e. maps which are non-singular and *measure-preserving* [22].

Notations 21. Given two sets A, B, we write $A \Delta B$ their symmetric difference.

Definition 22 (Refinement)**.** A graphing representative F is a refinement of a graphing representative G, noted $F \leqslant G$, if there exists a partition[1] $(E_e^F)_{e \in E^G}$ of E^F such that $\forall e \in E^G$:

$$\mu\left(\left(\cup_{f \in E_e^F} S_f^F\right) \triangle S_e^G\right) = 0; \qquad \forall f \neq f' \in E_e^F,\ \mu(S_f^F \triangle S_{f'}^F) = 0;$$

$$\forall f \in E_e^F,\ m_f^F = m_e^G$$

This notion defines an equivalence relation defined by $F \sim G$ if and only if there exists H with $H \leqslant F$ and $H \leqslant G$.

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

Definition 23. Aa *α-graphing* is an equivalence class of α-graphing representatives w.r.t. the equivalence relation generated by refinements.

We will now show how graphings relate to well-established notions in mathematics.

3.1 Dynamical Systems

Definition 24. 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 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 is 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 25. A graphing $G = \{S_e^G, \phi_e^G, \omega_e^G \mid e \in E^G\}$ is *deterministic* if 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 26. *There is a one-to-one correspondence between deterministic graphings and partial non-singular measurable-preserving dynamical systems (up to a.e. equality).*

Proof. 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 as follows. 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. □

3.2 A submodel

We now prove that the set of deterministic graphings is closed under composition, 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 and non-deterministic graphings define submodels of $\mathbb{M}[\Omega, \mathfrak{m}]$.

Lemma 27. *The set of deterministic graphings is closed under execution.*

Proof. 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 $F :: G$). □

One can then check that the interpretations of proofs by graphings in earlier papers [22, 24, 21] are all deterministic. This gives us the following theorem as a corollary of the previous lemma.

Theorem 28 (Deterministic model)**.** *Let Ω be a monoid and $\mathfrak{m}$ a microcosm. The set of Ω-weighted* deterministic *graphings in $\mathfrak{m}$ yields a model, denoted by $\mathbb{M}^{\mathtt{det}}[\Omega, m]$, of multiplicative-additive linear logic.*

3.3 Zeta Functions for dynamical systems

The Ruelle zeta function [16] 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 (all that counts is the trace so we may use a type II_1 factor here). 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 $\phi = 1$, this is the Artin-Mazur [1] 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 measure spaces, so it is natural to consider the following measured variant of the Ruelle zeta function (defined for measure-preserving maps[2]). Suppose that we work with a measure 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 $\phi = 1$, 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 [22].

Proposition 29. *Given measure-preserving* NSMP *partial dynamical systems $f, g : \mathbf{X} \to \mathbf{X}$, we have the following equality:*

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

where $[\![\_, \_]\!]_m$ denotes the measurement between graphings defined in earlier work [22].

3.4 The Probabilistic Model

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.

Definition 30. A graphing $G = \{S_e^G, \phi_e^G, \omega_e^G \mid e \in E^G\}$ is *sub-probabilistic* if the following holds:

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

[2]Based on the result of Proposition 29, a definition for general NSMP maps could be obtained using the method used by the author [22] to define a generalised measurement between graphings. However, we considered this to be out of the scope of this work.

Definition 31. A graphing $G = \{S_e^G, \phi_e^G, \omega_e^G \mid e \in E^G\}$ of support V is *probabilistic* (on V) if the following holds:

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

It turns out that this notion of graphing also behaves well under composition, i.e. there exists a *sub-probabilistic* submodel of $\mathbb{M}[\Omega, \mathfrak{m}]$, 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 32. *The set of sub-probabilistic graphings is closed under execution.*

Proof. 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 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. □

Theorem 33 (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.*

3.5 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 34. 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) [14].

Definition 35. A sub-Markov kernel is a measurable map $\kappa : X \times Y \to [0, 1]$ with the properties that for all $x \in X$ and $B \subset Y$, $\kappa(x, \_)$ 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.

We restrict in this paper to time-independent Markov processes. Within this section, we furthermore restrict to what we call *discrete-image kernels*.

Definition 36. A *discrete-image kernel* is a sub-Markov kernel $\kappa : X \times Y \to [0, 1]$ such that for all $x \in \mathbf{X}$, $\kappa(x, \_)$ is a discrete probability distribution.

Notations 37. 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 kernels $\kappa : X \times Y \to [0, 1]$ and $\kappa' : Y \times Z \to [0, 1]$ is computed as follows:

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

Theorem 38. *There is a one-to-one correspondence between sub-probabilistic graphings on* $\mathbf{X}$ *and discrete-image sub-Markov kernels* $\mathbf{X} \times \mathbf{X} \to [0, 1]$.

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:

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

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 $\{y \in \mathbf{X} \mid \kappa(x, y) > 0\}$ for all $x \in \mathbf{X}$ (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 $B \subset \mathbf{X}$. □

4 Markov processes and linear logic

Based on the previous sections, we want to extend the constructions to general Markov processes. For technical reasons, the model will be defined on sub-Markov kernels, but we discuss later the possible restrictions to Markov kernels.

Notations 39. In the following we write $\mathrm{Id} : \mathbf{X} \to \mathbf{X}$ the *identity kernel*, i.e. the Dirac delta funuction $\mathrm{Id}(x, \dot{x}) = \delta(x, \dot{x})$ satisfying $\int_A \mathrm{Id}(x, \dot{a}) = 1$ whenever $x \in A$ and $\int_A \mathrm{Id}(x, \dot{a}) = 0$ 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 40 (Iterated kernel)**.** Let κ be a sub-Markov kernel $\mathbf{X} \times \mathbf{Y} \to [0, 1]$. For $k > 1$, we define the k-th iterated kernel $\kappa^{(k)}$ as

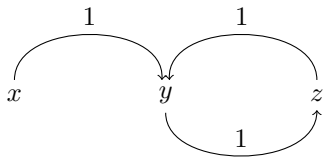
$$\kappa^{(k)}(x_0, \dot{x}_k) = \iint_{(x_1,\dots,x_{k-1}) \in (\mathbf{X} \cap \mathbf{Y})^{k-1}} \prod_{i=0}^{k-1} \kappa(x_i, \dot{x}_{i+1}).$$

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

Definition 41 (Maximal paths – Execution kernel)**.** Let κ be a sub-Markov kernel $\mathbf{X} \times \mathbf{Y} \to [0, 1]$. We define the *execution kernel* of κ as the map (in the formula, x_{n+1} is used as a notation for y):

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

While this maps could be defined on the whole space $\mathbf{X} \times \mathbf{Y}$, 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, an additional assumption on κ is required. We will discuss these issues in **??**. For the moment, we prove that $\mathrm{tr}^A(\kappa)$ is indeed a sub Markov kernel.

Lemma 42. *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):

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

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):

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

□

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 43 (Finite orbits – Zeta kernel)**.** Let κ be a sub-Markov kernel $\mathbf{X} \times \mathbf{Y} \to [0, 1]$. The *zeta kernel*, or kernel of finite orbits of κ is a kernel $\mathbf{X} \times \mathbf{N} \to [0, 1]$ – where $\mathbf{N}$ denotes the set of natural numbers – defined as:

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

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

Definition 44 (Zeta function)**.** We now define the Zeta function associated with a sub-Markov kernel $\kappa : \mathbf{X} \times \mathbf{Y} \to [0, 1]$:

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

4.1 Execution and the Cocycle Property

Definition 45. Given two sub-Markov kernels $\kappa : \mathbf{X} \times \mathbf{X}' \to [0, 1]$ and $\kappa' : \mathbf{Y} \times \mathbf{Y}' \to [0, 1]$, we define their execution $\kappa :: \kappa'$ as the kernel $\mathrm{tr}(\kappa \bullet \kappa')$ where:

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

The reader with notions from *traced monoidal categories* [10, 8, 7] should not be surprised of this definition and the following properties.

Definition 46. Three sub-Markov kernels $\kappa : \mathbf{X} \times \mathbf{X}' \to [0, 1]$, $\kappa' : \mathbf{Y} \times \mathbf{Y}' \to [0, 1]$, $\kappa'' : \mathbf{Z} \times \mathbf{Z}' \to [0, 1]$ are said to be *in general position*[3] 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 47. *Given three sub-Markov kernels $\kappa_0 : \mathbf{X} \times \mathbf{X}' \to [0, 1]$, $\kappa_1 : \mathbf{Y} \times \mathbf{Y}' \to [0, 1]$, $\kappa_2 : \mathbf{Z} \times \mathbf{Z}' \to [0, 1]$ in general position:*

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

Proof. The fact that this is true is a consequence of the fact that kernels are in general position. As a consequence, 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 $x_1, x_2, \ldots, x_k$ of the alternating product of

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

$\kappa(x_i, \dot{x}_{i+1})$ 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 [17], 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. □

Lemma 48. *Given two sub-Markov kernels $\kappa : \mathbf{X} \times \mathbf{X}' \to [0,1]$ and $\kappa' : \mathbf{Y} \times \mathbf{Y}' \to [0,1]$ 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 49. Given two kernels κ, κ', we define their *zeta-measurement* $\zeta_{\kappa,\kappa'}$ as the function $\zeta_{\kappa \bullet \kappa'}(z)$.

Proposition 50 (Cocycle)**.** *Given three sub-Markov kernels $\kappa : \mathbf{X} \times \mathbf{X}' \to [0,1]$, $\kappa' : \mathbf{Y} \times \mathbf{Y}' \to [0,1]$, $\kappa'' : \mathbf{Z} \times \mathbf{Z}' \to [0,1]$ in general position:*

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

Proof. 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 47 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. □

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 the author in earlier work [17], and is essential to obtain Theorem 57.

4.2 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 the author's model for second-order linear logic [21], 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 [24], 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 51. 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 $F : (\mathbf{X} \times [0,1]) \times (\mathbf{X} \times [0,1]) \to [0,1]$.

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 $(\mathbf{X} \times [0,1] \times [0,1]) \times (\mathbf{X} \times [0,1] \times [0,1]) \to [0,1]$:

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

Definition 52. 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 53. 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 47 and Lemma 48 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 54. 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 55. 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 56. For $\mathbb{A}$ and $\mathbb{B}$ two types of disjoint supports, we define:

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

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 [17, 20, 22].

Theorem 57. *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. However, this issue is dealt with in the models by exponentiation, as only exponentiated processes can be used multiple times. To ensure they end their computation in the same state as they started, the principle is that 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 58. 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 the author's model [21], 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[4] $B_{\mathbf{X}}^{-1} \circ \kappa \circ B_{\mathbf{X}}$, 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:

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

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

Definition 60 (Perrenisation)**.** 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

$$\sharp\mathbb{A} = \{!\mathfrak{a} \mid \mathfrak{a} \in \mathrm{bal}(\mathbb{A})\}.$$

[4]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}.

We are now ready to state the main theorem.

Theorem 61. *Restricting to spaces of the form* $\mathbf{X} = \mathbf{Y} \times [0,1]^{\mathbf{N}}$*, proof-objects and types define a sound model of linear logic.*

Proof. All the work has been done already. Indeed, the interpretations of linear logic proofs are defined in IG for full linear logic [21], 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 26, 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 [21]).

Remark 62. We restricted the model to the spaces of the form $\mathbf{X} = \mathbf{Y} \times [0,1]^{\mathbf{N}}$ to ease the presentation and proofs. However, it is clear 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 $\mathbf{X}$, $!\kappa$ is defined on $\mathbf{X} \times [0,1]$. In that case, no assumptions need to be made on $\mathbf{X}$, and $!\kappa$ is simply defined as the kernel

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

where the second copy of $[0,1]$ represent the set of states and

$$!\kappa(x, e; \dot{x}, \dot{e})(f; \dot{f}) = \kappa(x, e; \dot{x}, \dot{e})\mathrm{Id}(f, \dot{f}).$$

This construction however requires to redefine the interpretations of linear logic proofs, which are obtained as easy adaptations of the interpretation used by Seiller [21]. Nonetheless, we had to leave them out for lack of space but may include those in an extended version of this work.

5 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. We expect strong connections with game semantics models dealing with such languages [3], although our model 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 and quantifiers [22, 21, 6], and could be used to consider new type constructions adapted to probabilistic computation [12].

Lastly, 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 [19, 23]. 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* [15]. 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.

It is important to stress here 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 $\zeta_\kappa^\infty(z)$ 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 $\zeta_{\kappa \bullet \kappa'}^\infty(z) = 0$ 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] M. Artin and B. Mazur. On periodic points. *Annals of Mathematics*, pages 82–99, 1965.

[2] R. Bowen and O. Lanford. Zeta functions of restrictions of the shift transformation. *Proceedings of Symposia in Pure Mathematics*, 14:43–50, 1970.

[3] S. Castellan and H. Paquet. Probabilistic programming inference via intensional semantics. In L. Caires, editor, *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*, volume 11423 of *Lecture Notes in Computer Science*, pages 322–349. Springer, 2019.

[4] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987.

[5] J.-Y. Girard. Towards a geometry of interaction. In *Proceedings of the AMS Conference on Categories, Logic and Computer Science*, 1989.

[6] J.-Y. Girard. Geometry of interaction V: Logic in the hyperfinite factor. *Theoretical Computer Science*, 412:1860–1883, 2011.

[7] E. Haghverdi and P. Scott. A categorical model for the geometry of interaction. *Theoretical Computer Science*, 350(2):252–274, 2006.

[8] M. Hasegawa. Recursion from cyclic sharing: Traced monoidal categories and models of cyclic lambda calculi. pages 196–213. Springer Verlag, 1997.

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

[10] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3):447–468, 1996.

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

[12] A. Naibo, M. Petrolo, and T. Seiller. Logical constants from a computational point of view. *in preparation*.

[13] A. Naibo, M. Petrolo, and T. Seiller. On the computational meaning of axioms. In *Epistemology, Knowledge and the Impact of Interaction*, pages 141–184. Springer, 2016.

[14] P. Panangaden. The category of markov kernels. *Electronic Notes in Theoretical Computer Science*, 22:171 – 187, 1999. PROBMIV'98, First International Workshop on Probabilistic Methods in Verification.

[15] L. Pellissier and T. Seiller. Prams over integers do not compute maxflow efficiently. submitted, 2018.

[16] D. Ruelle. Zeta-functions for expanding maps and anosov flows. *Inventiones mathematicae*, 34(3):231–242, 1976.

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

[18] T. Seiller. *Logique dans le facteur hyperfini : géometrie de l'interaction et complexité*. PhD thesis, Université Aix-Marseille, 2012.

[19] T. Seiller. Towards a *Complexity-through-Realizability* theory. http://arxiv.org/pdf/1502.01257, 2015.

[20] T. Seiller. Interaction graphs: Additives. *Annals of Pure and Applied Logic*, 167:95 – 154, 2016.

[21] T. Seiller. Interaction graphs: Full linear logic. In *IEEE/ACM Logic in Computer Science (LICS)*, 2016.

[22] T. Seiller. Interaction graphs: Graphings. *Annals of Pure and Applied Logic*, 168(2):278–320, 2017.

[23] T. Seiller. Interaction graphs: Nondeterministic automata. *ACM Transaction in Computational Logic*, 19(3), 2018.

[24] T. Seiller. Interaction Graphs: Exponentials. *Logical Methods in Computer Science*, Volume 15, Issue 3, Aug. 2019.

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