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
Current address: 99 Avenue Jean-Baptiste Clément, 93430 Villetaneuse, FRANCE
e-mail address: seiller@lipn.fr

Abstract. The author introduced models of linear logic known as "Interaction Graphs" which generalise Girard's various geometry of interaction constructions. In this work, we establish how these models essentially rely on a deep connection between zeta functions and the execution of programs, expressed as a cocycle. This is first shown in the simple case of graphs, before begin lifted to dynamical systems. Focussing on probabilistic models, we then explain how the notion of graphings used in Interaction Graphs captures a natural class of sub-Markov processes. We then extend the realisability constructions and the notion of zeta function to provide a realisability model of second-order linear logic over the set of all (discrete-time) sub-Markov processes.

Introduction

We construct a mathematical model (semantics) of second-order linear logic using realisability techniques. The standard approach to semantics consists in starting from a logical system to produce a model, which then turns out to capture (well-behaved) programs in a given programming language through the proofs-as-programs correspondence [AJ94, HO00, EPT18]. In realisability, one starts from a model of computation and exhibits a logical system arising from it: a type system naturally describing the behaviour of the underlying set of programs. Realisability can then be used to study the relationship between the computational principles used to define programs and the logic of types; for instance exhibiting the relationship between bar recursion and the axiom of choice [BBC98].

Realisability models are known for intuitionnistic and classical logic [VO08, Kri01, Kri09, Miq11]. We are here interested in realisability models for linear logic. While a bibliographical search may not return many results, numerous realisability models were defined in the literature, under different names: ludics [Gir01, Gir03, Cur06, Ter11], geometry of interaction [Gir87, Gir89b, Gir89a, Gir88, Gir95, Gir06, Gir11], interaction graphs [Sei12a, Sei16b, Sei17, Sei19, Sei16c], transcendental syntax [Gir17, Gir16, Gir18, ES22]. This may be explained by the fact that realisability models for intuitionnistic or classical

T. Seiller was partially supported by the European Commission Marie Skłodowska-Curie Individual Fellowship (H2020-MSCA-IF-2014) project 659920 - ReACT, the INS2I grants BiGRE and LoBE, the Ile-de-France DIM RFSI Exploratory project CoHOp, and the ANR-22-CE48-0003-01 project DySCo.

logic are based on the lambda-calculus or variations thereof, while models for linear logic are constructed from varying and less standard models of computation. Recently, the author introduced such models with as underlying models of computation classes of generalised dynamical systems. This will be formally explained in Sections 2 and 3, where we also show that his construction provides models based on some restricted class of subprobability kernels – those kernels for which the probability distribution associated to each point is discrete. This raises the question of whether one can extend these ideas to a larger class of kernels, capturing continuous distributions. We will answer positively to this question at the end of this paper.

To construct this model, we build on a newfound fundamental property underlying the author's previous constructions. This property is expressed as a *cocycle condition*[1] relating the execution of programs and linear negation. More precisely, we show that the measurement used in interaction graphs models to define linear negation corresponds to computing the value at 1 of Ruelle's zeta function for dynamical systems [Rue76]. This is then used as a guideline: we define a *zeta kernel* for subprobabilistic kernels which is then used to define a realisability model for second-order linear logic.

Contributions and plan of the paper. 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. This opens the possibility of defining new models of typed lambda-calculus extended with probabilities and specific instructions for sampling discrete and non-discrete distributions. This result is guided by another, more technical, contribution establishing that previous linear realisability models were defined upon a formal geometric connection between program execution and zeta functions (for graphs and dynamical systems). Even though more technically involved and difficult to express in laymen's terms, establishing this fundamental connection between program execution and zeta functions is the second major contribution of this work, one which opens many research directions for future work.

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

- In Section 1, we recall the author's discrete Interaction Graphs (IG) models based on graphs and relate them with the work of Ihara on zeta functions of graphs [Iha66]. This section is mostly introductory: we explain in the simple setting of graphs how the models of linear logic (in this case only the multiplicative fragment MLL) are constructed, and in particular how the notion of type is inferred. We however already state the first contribution of this work: showing how these models essentially rely on a cocycle relation involving the notions of execution and Ihara's graph zeta functions.
- In Sections 2 and 3, we show how this fundamental observation lifts to the more involved models based on *graphings* [Sei17, Sei16c]. We first recall the basic notions and prove that the restriction to *deterministic* graphings boils down to a representation of programs as partial dynamical systems, showing how the interaction graphs constructions provide realisability models for linear logic on dynamical systems. We then prove that in this more general case, the models once again rely on a fundamental cocycle involving execution (here related to the iteration of the dynamical system) and Ruelle's zeta function for

[1]We use this terminology because of the strong resemblance between the condition (Equation 1.4) and the notion of 2-cocycle for a group action.

dynamical systems. Similar results are obtained for the notion of *probabilistic graphings* which we show coincide with a specific class of sub-Markov processes that we describe.
- In Sections 4 and 5, we generalise the realisability constructions from previous section to 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 essential cocycle relation. These results are then used to define realisability models of second-order linear logic over the set of all sub-Markov kernels.

Related work. The results presented here are strongly related to the geometry of interaction program of Girard [Gir87, Gir89b, Gir89a, Gir88, Gir95, Gir01, Gir06, Gir11, Sei12a, Sei16b, Sei17, Sei19, Sei16c, Gir17]. As a consequence, it is close in spirit to game semantics approaches to probabilistic programs [DH02, PW18, CCPW18, CP18, Paq21]. There are strong connections between geometry of interaction semantics and coherence space semantics for linear logic. One might expect some functorial relationship (maybe based upon the antisymmetric tensor algebras – or *Fock* – functor [Sei16a]) between the work presented here and either probabilistic coherence spaces [Gir04, DE11, ETP14, EPT18, Ehr19], or denotational semantics related to staochastic kernels [Gir99, Ker18, DK19, dAKM$^+$21].

1. Interaction Graphs: the discrete case

In this section, we review the models of linear logic introduced by the author under the name "Interaction Graphs". While next section will be devoted to the general case where programs are represented as *graphings*, we here restrict to the more simple specific case of programs represented as graphs.

Graphs provide a minimal but natural mathematical structure to represent programs. Indeed, Turing machines and automata can naturally be abstracted as finite graphs. Obviously, some information is lost by considering discrete graphs: following an edge in a transition graph corresponds to performing some instruction which modifies the state of the machine, something that cannot be accounted for with finite graphs. Restricting to finite structures in some sense limits the approach to models of computation with finite sets of states. To regain expressivity, the author introduced graphings [Sei17], which will be formally introduced in the next section. Graphings are graphs realised on a topological or measured space which represent the space of all possible configurations of the machine. This allows to interpret edges of the transition graph as specific endomorphisms of this space, recovering the expressivity lost by considering only discrete structures.

In order to ease the presentation, we restrict the discussion to finite graphs in this section, stressing that all the intuitions built in this easier setting will carry over in the next sections. In this model for which a program is abstracted as a finite graph, computation is represented as the formation 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 – a sequence of instructions. The dynamic process of computation itself is therefore represented as the operation of *execution* $\mathrm{Ex}(G)$ of a graph G, which is the set of 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. From the point of view of logic, this operation of execution computes the normal form of a proof, i.e. accounts for the cut-elimination procedure.

Types are then defined as sets of graphs (satisfying some properties), i.e. a type $\mathbb{A}$ is identified with the set of all programs typable by $\mathbb{A}$. 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).

One key observation is that a type is not just any set of programs, but one satisfying a closure property. More specifically, a type is a set $\mathbb{A}$ of graphs such that $\mathbb{A} = \mathbb{A}^{\perp\!\!\!\perp\perp\!\!\!\perp}$, where $\perp\!\!\!\perp$ is an *orthogonality relation* accounting for linear negation. Equivalently, a type is a set $\mathbb{A}$ of graphs such that $\mathbb{A} = T_{\mathbb{A}}^{\perp\!\!\!\perp}$ for some set $T_{\mathbb{A}}$ understood as a set of *tests*. For instance, the natural tests for a graph F of type $\mathbb{A} \multimap \mathbb{B}$ consist of pairs (A, B') where A is an element of $\mathbb{A}$ given as input and $B' \in \mathbb{B}^{\perp\!\!\!\perp}$ is used to test the result of the computation can be given the type $\mathbb{B}$. From the point of view of logic, this interactive view of the definition of linear negation extends the notion of *correctness criterion* for MLL proof nets.

After giving this informal overview, we will formally define the models based on graphs. We first review basic definitions, and then explore the relationship with Ihara's zeta function of a graph, which will be extended to the more general setting of graphings in later sections.

1.1. **Interaction Graphs: basic notions.** We briefly recall the basics of Interaction Graphs (IG) model in the discrete case. We work with weighted directed (multi-)graphs; here we will suppose weights are picked in the field of 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 first essential 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 1.1. 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.

Definition 1.2 (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 $t^{G \cup H}(\pi) = t^{G \cup H}(e_{k-1})$.

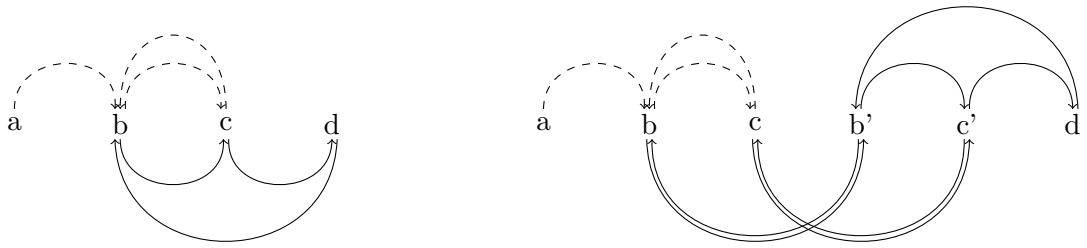


Figure 1: On the left: implicit cut between two graphs (one is plain, the other is dashed). On the right: explicit cut between the same two graphs (the cut is shown below). Both representations lead to the same result, as the cut-elimination is represented by *execution*, an operation defined from alternating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

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 V.

Definition 1.3. Let F and G be two graphs. The *execution* of F and G is the graph $F :: G$ defined by:

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

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 [NPS16] for more details and explanations. Defining orthogonality in IG models is is done by quantifying *closed paths* and *prime closed paths*.

Definition 1.4. Given a graph G, a closed path π (called *circuit* in earlier work [Sei12a]) 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 1.5. 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 is the essential ingredient to define types using realisability techniques. We only recall the measurement here and refer to the first IG paper for more details [Sei12a]. 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 1.6. Let m be a map $\mathbf{C} \to \mathbf{R}_{\geqslant 0}$. For any two graphs F, G we define the measurement

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

Based on these two ingredients (execution and measurement), and two essential properties, namely the associativity of execution [Sei12a] and the *trefoil property* [Sei16b], one can define a myriad of models of Multiplicative Linear Logic (MLL) and Multiplicative-Additive Linear Logic (MALL). As shown by the author, these models capture all the different geometry of interaction models introduced by Girard by choosing carefully the map m used to define the measurement [Sei16b]. We will now explain how there is a similarity between the measurement 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. This connection will then be used to define IG models of multiplicative linear logic.

1.2. **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 [Ter10] 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 non-weighted directed graphs (i.e. there is no weight map ω^G or, equivalently, this map is the constant map equal to 1) and suppose they are *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 1.7. 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 is essential in our work, as it provides an 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 the author in the special case $z = 1$ [Sei12a] and was the initial motivation behind the definition of orthogonality in Interaction Graphs models, since it relates the measurement with the Fuglede-Kadison determinant of operators [FK52] used in Girard's model [Gir11].

Lemma 1.8. *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 1.9. *Let G be a graph and $M(G)$ its* transition matrix*, then* $\mathrm{tr}(M(G)^k) = N(k)$.

The previous lemmas are standard results from the theory of Zeta functions. Together, they yield the following result.

Proposition 1.10. *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 earlier work [Sei12a, 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.

1.3. **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.

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

Similarly to the case of unweighted graphs, we define the *transition matrix* of a *simple* weighted graph 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.

For a general (i.e. non-simple) weighted graph G, we write $G(v, v')$ the set $\{e \in E^G \mid s^G(e) = v, t^G(e) = v'\}$. One can then 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:

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

We here note that the author proved in earlier work that the measurement defined from the function[2] $m := \lambda x.x \mapsto \log(1-x)$ satisfies $[\![F, G]\!]_m = [\![\hat{F}, \hat{G}]\!]_m$.

The zeta function of a weighted graph is defined as follows.

Definition 1.11. 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.

Readers familiar with zeta functions of weighted graphs will notice 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 here multiply expressions $1 - \omega(\pi)z$ instead of $1 - z^{\nu(\pi)}$ in the standard definition.

Adapting the proof of the non-weighted case (Proposition 1.10), one obtains the following general result, which extends the author's combinatorial interpretation of the determinant $\det(1 - M(G))$ [Sei12a, Corollary 61.1].

Proposition 1.12. *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 $z = 1$:

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

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

Orthogonality in IG models is defined by $F \perp G$ as $[\![F, G]\!]_m \neq 0, \infty$, i.e. $-\log(\zeta_{F \bullet G}(1)) \neq 0, \infty$. Through this previous result, this is equivalent to the fact that $\zeta_{F \bullet G}(1) \neq 0, 1$. We will now build on this remark to extend the construction of IG models. This provides a new family of models using zeta functions to define the orthogonality.

1.4. **Zeta, Execution and a Cocycle Property.** As we mentionned earlier, there are two essential properties ensuring that IG realisability models represent (multiplicative additive) linear logic [Sei16b, Sei12b]. The first is the associativity of execution

$$F :: (G :: H) = (F :: G) :: H. \tag{1.1}$$

The second is the so-called *trefoil property* [Sei16b]:

$$[\![F, G :: H]\!]_m + [\![G, H]\!]_m = [\![G, H :: F]\!]_m + [\![H, F]\!]_m. \tag{1.2}$$

[2] In this paper, we use the lambda notation to write down functions, i.e. $\lambda x.\log(1-x)$ denotes the function $x \mapsto \log(1-x)$.

Those properties are satisfied under some mild hypothesis on the graphs (i.e. that $V^F \cap V^G \cap V^H = \emptyset$). Technically speaking, the trefoil property is obtained as a consequence of a geometric identity [Sei12b, Sei16b] establishing that when $V^F \cap V^G \cap V^H = \emptyset$, there is weight-preserving bijection between the following sets of closed paths:

$$\mathcal{C}(F, G :: H) \uplus \mathcal{C}(G, H) \equiv \mathcal{C}(G, H :: F) \uplus \mathcal{C}(H, F). \tag{1.3}$$

This geometric identity can be used to rephrase Equation 1.2 as a special case of a general cocycle condition satisfied by zeta functions. Indeed, using the fact we noticed earlier that $[\![F, G]\!]_{\lambda x.\log(1-x)} = -\log(\zeta_{F \bullet G}(1))$, the trefoil property (Equation 1.2) is a straightforward consequence of the following theorem (taking $z = 1$).

Theorem 1.13. *Suppose $V^F \cap V^G \cap V^H = \emptyset$. Then:*

$$\zeta_{F \bullet (G :: H)}(z).\zeta_{G \bullet H}(z) = \zeta_{G \bullet (H :: F)}(z).\zeta_{H \bullet F}(z). \tag{1.4}$$

Proof. By definition and the geometric trefoil property:

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

which is what we wanted to prove. □

We can then define families of models of linear logic extending the Interaction Graphs approach by considering the following constructs. We change the terminology w.r.t. earlier papers to avoid conflicts. We use the term *proof-object* in place of the term *project*, and we call *types* what was called a *conduct*. We also use the term *antipode* for the set of functions defining the orthogonality relation, as the standard term "pole" might be confused with the notion of pole from complex analysis.

Definition 1.14. A *proof-object* of support V is a pair (g, G) of a function $g : \mathbf{C} \to \mathbf{C}$ and a graph G with $V^G = V$.

Definition 1.15. 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 1.16. 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 $E^{\perp_P} = \{\mathfrak{g} \mid \forall \mathfrak{e} \in E, \mathfrak{e} \perp_P \mathfrak{g}\}$.

We note that many interesting properties of the graph can be used to define orthogonality in this case. Indeed, a number of properties (e.g. connectedness) and invariants (e.g. Euler characteristic) of a graph can be related to analytic properties of the zeta function of

a graph. We also note that previous notions of orthogonality [Sei12a] can be recovered by considering as antipode the set of functions f such that $f(1) \neq 0, 1$.

We now define types and explain how models of MLL can be defined from this. The techniques are standard, and the main results are direct consequences of the above properties. 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 1.17. 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 [Sei16b]. 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 (here expressed as the cocycle Eq. (1.4)).

2. Graphings and dynamical systems

In this section, we review the more general setting of Interaction Graphs based on graphings [Sei17]. We first explain how the notions introduced in the previous section generalise, pinpointing how out the general construction based on zeta functions naturally adapts here. We then explain how deterministic graphings correspond to partial dynamical systems.

We first recall briefly the notion of graphing. Interested readers can find more detailed presentations in the author's recent work on computational complexity [Sei18b, SPL22]. The definition is parametrised by an *abstract model of computation*: a monoid action $\alpha : M \curvearrowright \mathbf{X}$ on the underlying space $\mathbf{X}$. As an example, Turing machines give rise to a monoid action as follows. One considers 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 $*$. Then the monoid is generated by five maps: $\texttt{left}$, $\texttt{right}$, $\texttt{write}_0$, $\texttt{write}_1$, $\texttt{write}_*$ acting on $\mathbf{X}$ as expected: for instance moving the working head to the right can be represented as the map $\texttt{right} : \mathbf{X} \to \mathbf{X}, (a_i)_{i\in\mathbf{Z}} \mapsto (a_{i+1})_{i\in\mathbf{Z}}$, and the map $\texttt{write}_0$ acts as $(a_i)_{i\in\mathbf{Z}} \mapsto (\hat{a}_i)_{i\in\mathbf{Z}}$ where $\hat{a}_i = 0$ if $i = 0$ and $\hat{a}_i = a_i$ otherwise.

Definition 2.1 (Abstract model of computation)**.** An *abstract model of computation* (AMC) is a monoid action $\alpha : M \curvearrowright \mathbf{X}$.

Given an AMC $\alpha : M \curvearrowright \mathbf{X}$, one then defines α-graphings (or *abstract programs*) through the notion of *α-graphing representative*. A graphing representative is a *geometric realisation* of a graph: it is a collection of pairs (S, m) (called edges) where S is a subspace of $\mathbf{X}$ – the source of the edge – and m is an element of the monoid – a sequence of instructions. For instance, in the Turing machines example above, the instruction "if the head is reading a 0 or a 1, move to the right" is represented as an edge (S, m) of source the subspace $S = \{(a_i)_{i\in\mathbf{Z}} \in \mathbf{X} \mid a_0 \neq *\}$ and realised by the map $\texttt{right}$. We will leave the reader convince herself that any Turing machine can be represented in this way.

Definition 2.2. 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 will identify non-weighted graphings with weighted graphings with constant weight equal to 1. Also, while the notion is quite general, we will restrict our discussion to the case of $\mathbf{X}$ being a measured space.

Remark 2.3. We note that in the general setting, a graphing also possesses a space of control states Q^G. The notions of source and realisers are then adapted: the source is a subset of $\mathbf{X} \times Q^G$, and the realiser is an element of $M \times \mathfrak{S}_{Q^G}$ – where $\mathfrak{S}_{Q^G}$ is the group of permutations on Q^G. While this generalisation is important for defining the models, we will only introduce states in the section on Markov processes. This makes the results in this section (and the next) easier to state.

An α-graphing is then defined as an equivalence class of graphing representatives w.r.t. some notion of *refinement*. The intuition is that an α-graphing represents an *action* on the underlying space, which can be defined by different graphing representatives. The base example is that of 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 $m_{e_1} = m_{e_2} = m_e$. The graphing representatives G and H represent the same action on the underlying space $\mathbf{X}$ as long as[3] $S_e =_{a.e.} S_{e_1} \cup S_{e_2}$ 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 2.4 (Refinement)**.** A graphing representative F is a refinement of a graphing representative G, noted $F \leqslant G$, if there exists a partition[4] $(E_e^F)_{e \in E^G}$ 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 are then equivalent (have the same action on the underlying space) whenever there exists a common refinement H, i.e. such that $H \leqslant F$ and $H \leqslant G$. The fact that this defines an equivalence relation compatible with the essential operations on graphing representatives to define models of linear logic (execution and measurement), is shown in the author's first work on graphings [Sei17].

Definition 2.5. An *α-graphing* (or an *abstract program in the* AMC α 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 earlier papers [Sei17, Sei16c] for the definitions of execution and measurement of graphings [Sei17]. We will write $\mathbb{M}[\Omega, \alpha]$ the obtained realisability model, where Ω is the monoid of weights (as already mentioned, we will only consider the case $\Omega = \mathbf{C}$ in this paper) and α the AMC. By extension, the notation $\mathbb{M}[\Omega, \alpha]$

[3]Since we supposed $\mathbf{X}$ is a measured space, equalities holds up to a null measure set, while those can be exact in other cases, e.g. topological spaces.

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

also denotes the set of all Ω-weighted α-graphings. We will now show how graphings relate to well-established notions in mathematics.

2.1. **Dynamical Systems.**

Definition 2.6. 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 (and *orbits* $\{(f^i(x))_i \mid x \in \mathbf{X}\}$) that are of interest as they describe how the system will evolve.

Dynamical systems represent deterministic systems, such as those described by classical mechanics. If one wants to describe non-deterministic behaviour, one is lead to consider several partial maps. The resulting object coincides with the notion of *graphing* without weights. Describing probabilistic behaviour 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 (where we will show that they correspond to a subclass of subprobabilistic kernels), we now focus on the deterministic case.

Definition 2.7. An α-graphing $G = \{S^G_e, \phi^G_e, \omega^G_e \mid e \in E^G\}$ is *deterministic* if $\forall e \in E^G$, $\omega^G_e = 1$ and the following holds:

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

Theorem 2.8. *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. Since G it is deterministic, we can consider a representative $\bar{G}$ of G such that the set

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

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

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

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 $(\mathbf{X}, \cup_{e \in E^{\bar{G}}} S^{\bar{G}}_e, \Phi)$. 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. □

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[5] $\mathcal{P}(\alpha) = \{(x, y) \in \mathbf{X} \times \mathbf{X} \mid \exists m \in M, \alpha(m)(x) = y\}$.

2.2. **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 defines a realisability model of linear logic, using the restriction of the measurement defined on $\mathbb{M}[\Omega, \alpha]$.

Lemma 2.9. *The execution of two deterministic graphings is a deterministic graphing.*

Proof of Lemma 2.9. A *deterministic graphing* F satisfies that for every edges $e, f \in E^F$, $S^F_e \cap S^F_f$ 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 $B \subset S^{F::G}_e \cap S^{F::G}_f$. 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 $\pi_e = f_0\pi^1_e$ and $\pi_f = f_0\pi^1_f$. Thus the domains of the paths π^1_e and π^1_f coincide on the Borel set $\phi^F_{f_0}(B)$ 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 2.11. 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 [Sei17, 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 [Sei12a, 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.

[5]If α is a group action by measure-preserving transformations, $\mathcal{P}(\alpha)$ is a *Borel equivalence relation*, which can be used to construct von Neumann algebras with a distinguished maximal abelian subalgebra (MASA) [FM77a, FM77b]. This is one of the intuitions behind the author's approach to complexity using graphings, since he established a correspondence between inclusions of MASAs in von Neumann algebras and the expressivity of the logical system arising from realisability techniques [Sei18a].

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

Figure 2: General measurement function on graphings.

As established by the author, this expression simplifies in the measure-preserving case [Sei12a, Proposition 52], and can be expressed as

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

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:

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

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:

$$\begin{aligned}\log(\zeta_{g\circ f,1}(z))&=\sum_{m\geq1}\int_{\mathrm{Fix}((g\circ f)^m)}\frac{z}{m}\\&=\sum_{m\geq1}\sum_{\pi\in\mathrm{AltCycle}(F,G)_m}\int_{S_m^\pi}\frac{z}{m}\end{aligned}$$

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 the author's first paper on Interaction Graphs [Sei12a]):

$$\begin{aligned}\log(\zeta_{g\circ f,1}(z))&=\sum_{m\geq1}\sum_{\pi\in\mathcal{P}[f,g]}\int_{S_m^\pi}\frac{z}{m}\\&=\sum_{\pi\in\mathcal{P}[f,g]}\sum_{m\geq1}\int_{S_m^\pi}\frac{z}{m}\\&=[\![f,g]\!]_m\end{aligned}$$

This is the equality we wanted to prove. □

One can then check that the interpretations of proofs by graphings in earlier papers [Sei17, Sei19, Sei16c] are all deterministic. This gives us the following theorem as a corollary of the previous lemma.

Theorem 2.10 (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.*

This thus defines a realisability 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 2.8 and Theorem 2.10. This is constructed using the measurement defined in earlier work [Sei17] which, similarly to the graph case, we will now show is related to a standard notion of zeta function.

2.3. **Zeta Functions for dynamical systems.** The Ruelle zeta function [Rue76] 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):

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

For $d = 1$ and $\phi = 1$ the constant function equal to 1, this is the Artin-Mazur [AM65] zeta function:

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

Since we work with measured spaces, we consider the following measured variant of Ruelle's zeta function (defined for measure-preserving maps[6]). Suppose 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 $\phi = 1$, this becomes:

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

which we relate to the measurement on graphings defined in earlier work [Sei17].

Proposition 2.11 (Proof in appendix)**.** *Given measure-preserving* NSMP *partial dynamical systems $f, g : \mathbf{X} \to \mathbf{X}$, for all constant c we have:*

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

with $[\![\_, \_]\!]_m$ the standard measurement on graphings [Sei17].

[6]Based on the result of Proposition 2.11, a definition for general NSMP maps could be obtained using the method used by the author [Sei17] to define a generalised measurement between graphings. However, we considered this to be out of the scope of this work.

This shows that the author's realisability (sub)models of deterministic graphings – or equivalently of partial measured dynamical systems – can be constructed using zeta functions to define the orthogonality, similarly to the restricted graph setting we considered in the previous section.

3. Probabilities and kernels

As mentioned above, one could also consider a notion of *probabilistic graphings* to represent probabilistic processes. Recall that we consider here $\Omega = \mathbf{C}$ so it makes sense to talk about weights taken in the interval $[0,1]$. We will show how this notion is closed under composition – hence defines a *sub-probabilistic model* –, and how the corresponding objets capture specific subprobabilistic kernels.

3.1. **A probabilistic model.**

Definition 3.1. 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 (Remark 4.4), probabilistic graphings are *not* closed under composition.

Theorem 3.2. *The execution of two sub-probabilistic graphings is a sub-probabilistic graphing.*

Proof of Theorem 3.2. 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. □

As a corollary, we get an equivalent of Theorem 2.10.

Theorem 3.3 (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.*

We now explain how these models can be understood as realisability models over a subclass of (sub-)Markov processes.

3.2. **Discrete-image sub-Markov processes.** We are now considering probabilistic systems. More specifically, we consider systems for which evolution is still time-independent, but which obey the principle of probabilistic choices: given a state, it 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 3.4. 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 $\mathbf{X}$.

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. This action of deterministic graphings (equivalently, dynamical systems) on the set of (sub-)probability distributions $\mathbb{P}(\mathbf{X})$ can be naturally extended to an action of sub-probabilistic graphings on $\mathbb{P}(\mathbf{X})$. In fact, we show that sub-probabilistic graphings define sub-Markov kernels. We recall 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) [Pan99].

Notations 3.5. In this section and the following, we write measured spaces $\mathbf{X}$, $\mathbf{Y}$, etc. in boldface fonts. We will use the same letter in normal fonts, e.g. X, Y, etc. to denote the underlying set and the same letter in calligraphic fonts, e.g. $\mathcal{X}$, $\mathcal{Y}$, etc. to denote the associated σ-algebra. We do not assume generic notation for the measures and, should the need to talk about them arise, we would explicitly name them.

Definition 3.6. Let $\mathbf{X}$, $\mathbf{Y}$ be measured spaces. A sub-Markov kernel on $\mathbf{X}\times\mathbf{Y}$ is a measurable map $\kappa : X\times\mathcal{Y}\to[0,1]$ such that $\forall x\in X$ and $\forall B\in\mathcal{Y}$, $\kappa(x,\_)$ 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.

Definition 3.7. 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 3.8. 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 3.9. *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 $B \in \mathcal{X}$. □

As a consequence of the results in this section, the author's work [Sei16c] give rise – when restricting to subprobabilistic graphings – to a realisability model of linear logic over discrete-image sub-Markov kernels. We now have the needed context to address to the main question answered (positively) in this work: can one construct a realisability model of linear logic on the set of (unconstrained) sub-Markov processes.

4. A sub-Markov processes cocycle

Based on the previous sections, we will extend the realisability constructions to general Markov sub-processes. The need to consider sub-Markov kernels and not only Markov kernels is explained by technical reasons we illustrate below (Remark 4.4).

Notations 4.1. In the following we write $\mathbf{1}$ the *identity kernel* on $\mathbf{X} \to \mathbf{X}$, i.e. the Dirac delta function $\mathbf{1}(x, \dot{x}) = \delta(x, \dot{x})$ s.t. $\int_A \mathbf{1}(x, \dot{x}) = 1$ if $x \in A$ 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 will then proceed to prove the cocycle property which will ensure the realisability model obtained captures the linear logic discipline.

4.1. **Execution and Zeta.**

Definition 4.2 (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:

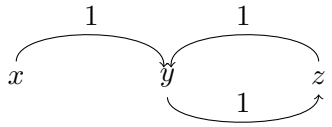
$$\kappa^{(k)}(x_0, \dot{x}_k) = \iint_{(x_1,\ldots,x_{k-1}) \in (\mathbf{X} \cap \mathbf{Y})^{k-1}} \prod_{i=0}^{k-1} \kappa(x_i, \dot{x}_{i+1}).$$

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

Definition 4.3 (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):

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

Remark 4.4. One could wonder why this is not defined on the whole space $X \times \mathcal{Y}$. The restriction is needed to define a sub-Markov kernel, something that 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.

Lemma 4.5. *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 $\int_{a \in A} \sum_{i=1}^{k} \kappa^{(i)}(x, \dot{a})$ 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 using the additivity of κ (we recall that A and $\mathbf{X}\cap\mathbf{Y}$ do not intersect):

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

which is the required bound. □

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 call the "zeta kernel".

Definition 4.6 (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:

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

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 in Remark 4.4), 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 4.7 (Zeta function)**.** We now define the Zeta function associated with a sub-Markov kernel κ on $\mathbf{X} \times \mathbf{Y}$:

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

4.2. **Execution and the Cocycle Property.**

Definition 4.8. 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* [JSV96, Has97, HS06] should not be surprised of this definition and the following properties[7].

Definition 4.9. 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*[8] 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 4.10. *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:*

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

Proof. 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 $x_1, x_2, \ldots, x_k$ 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. Using the geometric identity relating alternating paths and cycles (Equation 1.3 on page 9, established in [Sei16b]), this is the same as integrating over all alternating paths between κ_0 and alternating paths between κ_1 and κ_2. □

Lemma 4.11 (Proof in appendix)**.** *Given two sub-Markov kernels κ on $\mathbf{X} \times \mathbf{X}'$ and κ' on $\mathbf{Y} \times \mathbf{Y}'$ such that $\mathbf{X} \cap \mathbf{Y} = \mathbf{X}' \cap \mathbf{Y}' = \emptyset$:*

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

[7] In fact, the execution kernel should define a trace in the categorical sense.

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

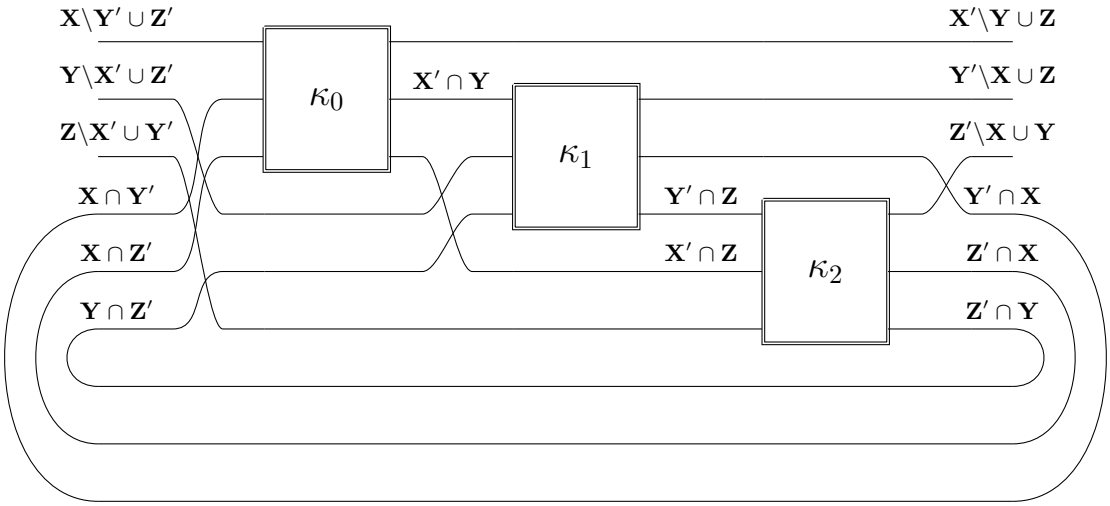


Figure 3: Proof of Lemma 4.10, first figure

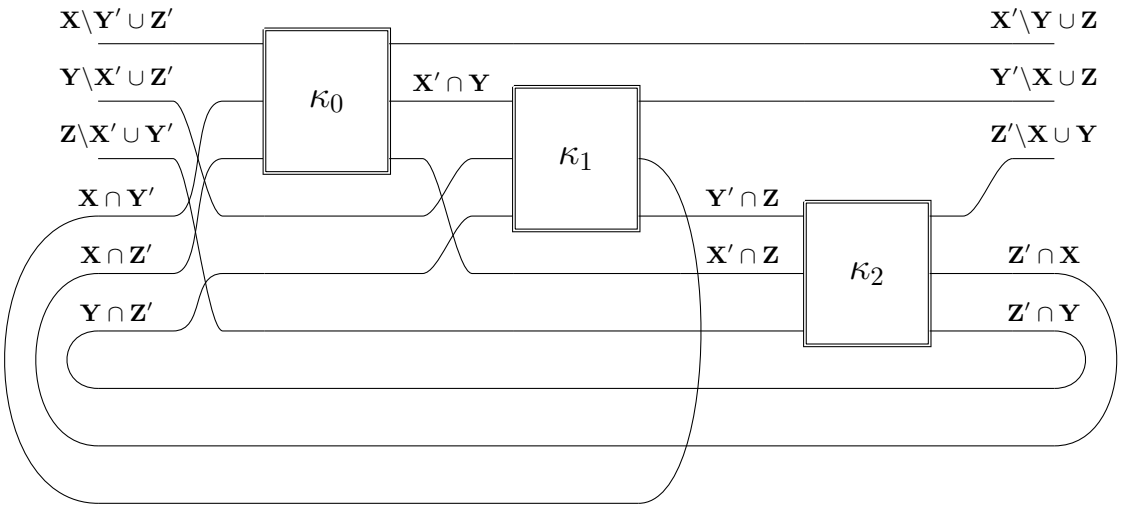


Figure 4: Proof of Lemma 4.10, second figure

Proof. The assumption on the spaces implies that one can picture the composition as shown in Figure 5.

□

This establishes the existence of a well-defined associative execution, the first ingredient for constructing linear realisability models. Following what was exposed in the first sections, we now define a zeta function associated to pairs of general sub-Markov processes, and show it satisfies the required cocycle property w.r.t. execution.

Definition 4.12. Given two kernels κ, κ', we define their *zeta-measurement* $\zeta_{\kappa,\kappa'}$ as the function $\zeta_{\kappa\bullet\kappa'}(z)$.

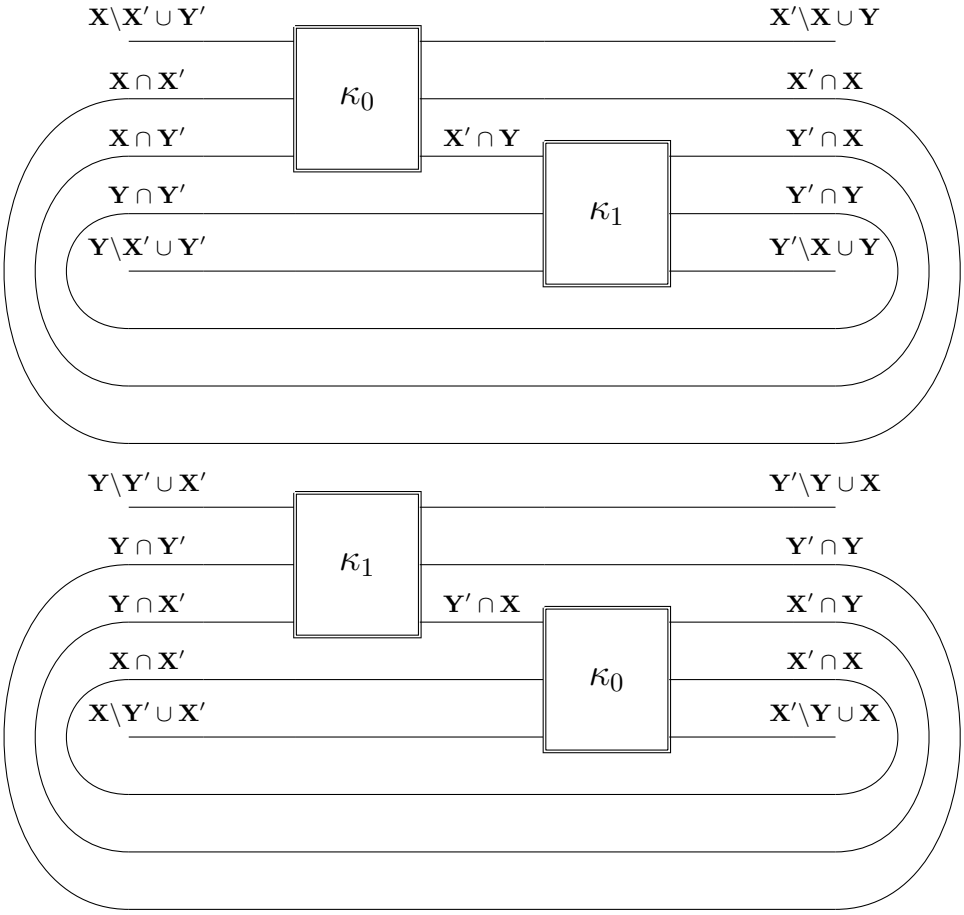


Figure 5: Figure for proof of Lemma 4.11

Proposition 4.13 (Cocycle, proof in appendix)**.** *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:*

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

Proof of Proposition 4.13. The proof consists in heavy computations, but without any technical difficulties. The main ingredient is again the geometric adjunction (Equation 1.3 on page 9). The pictures shown in the proof of **??** 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 [Sei12a].

4.3. **A first model of Linear Logic.** To obtain a model of full linear logic, one has to consider sub-Markov kernels with a set of states. Following a previous construction of a model of second-order linear logic [Sei16c], we will represent the set of states by the segment $[0,1]$.

Definition 4.14. 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: 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 $(\mathbf{X} \times [0,1] \times [0,1]) \times (\mathbf{X} \times [0,1] \times [0,1]) \to [0,1]$:

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

Definition 4.15. Given two proof objects $\mathfrak{f} = (f, \kappa_{\mathrm{F}})$ and $\mathfrak{g} = (g, \kappa_{\mathrm{G}})$ we define the *zeta-measurement* as the 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 4.16. 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 $[0,1]^2$.

Based on **??** and Lemma 4.11 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. This follows the construction on graphs in Section 1.

Definition 4.17. 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. We now explain how to construct a model of second order linear logic. We will omit the description of the construction of additive connectives: it follows from earlier work [Sei16b, Sei19] in a straightforward manner.

We now try to provide intuitions on exponentials.

Definition 4.18. 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 4.19. For $\mathbb{A}$, $\mathbb{B}$ types of disjoint supports, we define:

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

A direct consequence of the cocycle property and these definitions is the following property: 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 replaces the program $\mathfrak{a}$ by a single-state program $!\mathfrak{a}$, encoding the states of $\mathfrak{a}$ in the configuration space to avoid information loss. This encoding requires the underlying space $\mathbf{X}$ to be large enough, i.e. contain the space $[0,1]^{\mathbf{N}}$. Exponentiation, represented in this way, is therefore defined as long as the underlying space $\mathbf{X}$ contains $[0,1]^{\mathbf{N}}$.

We thus restrict in this section to spaces of the form $\mathbf{X} = \mathbf{Y} \times [0,1]^{\mathbf{N}}$, but we will show in the next section how to bypass this restriction.

Definition 4.20. A proof-object (f, κ_{F}) is *balanced* if $f = 1$, the constant function equal to 1. If E is a set of proof-objects, we write $\mathrm{bal}(E)$ the subset of balanced proof-objects in E.

Following an earlier model [Sei16c], 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[9] $B_{\mathbf{X}}^{-1} \circ \kappa \circ B_{\mathbf{X}}$, 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: $!\kappa : (x, e, \dot{x}, \dot{e}) \mapsto B_{\mathbf{X}}^{-1} \circ \kappa \circ B_{\mathbf{X}}(x, \dot{x}) \mathbf{1}(e, \dot{e})$. 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 4.21 (Perennisation)**.** Let $\mathfrak{f} = (0, \kappa_{\mathrm{F}})$ be a balanced proof-object. We define its *perennisation* $!\mathfrak{f} = (0, !\kappa_{\mathrm{F}})$.

Definition 4.22 (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 $\sharp\mathbb{A} = \{!\mathfrak{a} \mid \mathfrak{a} \in \mathrm{bal}(\mathbb{A})\}$.

This defines a model of second-order linear logic (LL^2) using the constructions from the author's work on graphings [Sei16c].

Theorem 4.23. *Restricting to spaces $\mathbf{X} = \mathbf{Y} \times [0,1]^{\mathbf{N}}$, proof-objects and types define a sound model of* LL^2.

[9] 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}.

Proof. In the IG model for full linear logic [Sei16c], linear logic proofs are interpreted by *deterministic graphings*. As such, they are in fact interpreted by dynamical systems by Theorem 2.8, which in turn define sub-Markov kernels. □

The model just sketched restricts the type of spaces considered. We will therefore devote the next section to bypass this issue, explaining how to model LL^2 in the unrestricted setting of sub-Markov kernels by defining a new interpretation of exponential connectives.

5. Sub-Markov processes and linear logic

Notations 5.1. 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 $\kappa(x,e,\dot{y},\dot{f})$.

To avoid restricting to spaces of the form $\mathbf{X} = \mathbf{Y} \times [0,1]^{\mathbf{N}}$, we will consider that κ and $!\kappa$ need not act on the same space: while κ is defined on $\mathbf{X} \times \mathbf{Y}$, $!\kappa$ will be defined on $(\mathbf{X} \times [0,1]) \times (\mathbf{Y} \times [0,1])$. This implies that the we need to generalise the framework to define proof-objects with an underlying sub-Markov kernel on $\mathbf{X} \times \mathbf{Y}$ and not necessarily on $\mathbf{X} \times \mathbf{X}$.

Notations 5.2. 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 constructions, which require to redefine parts of the interpretations of linear logic proofs [Sei16c].

5.1. Multiplicatives.

The definition of orthogonality, types, and multiplicative connectives follow the constructions exposed in previous sections.

Definition 5.3. A *general proof-object* of support $\mathbf{X} \to \mathbf{Y}$ is a pair $\mathfrak{f} = (f, \kappa_{\text{F}})$ of a complex function f and a sub-Markov kernel κ_{F} from $\mathbf{X}$ to $\mathbf{Y}$.

The zeta-measurement and the notion of antipode are defined as above (Definition 4.15 and Definition 4.17).

Definition 5.4. 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 $\zeta_{\mathfrak{f},\mathfrak{g}} \in P$.

From now on, we fix an antipode and omit subscripts.

Definition 5.5. A *type* of support $\mathbf{V}$ is a set $\mathbb{A}$ of general proof-objects of support $\mathbf{V}$ such that $\mathbb{A} = \mathbb{A}^{\perp\perp}$.

Definition 5.6. 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 $(\mathbf{X} \cup \mathbf{X}') \setminus (\mathbf{Y} \cup \mathbf{Y}') \to (\mathbf{Y} \cup \mathbf{Y}') \setminus (\mathbf{X} \cup \mathbf{X}')$ defined as $\mathfrak{f} :: \mathfrak{g} = (\zeta_{\mathfrak{f},\mathfrak{g}}, \kappa_{\text{F}} :: \kappa_{\text{G}})$.

Remark 5.7. 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 5.8. 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.

Notations 5.9. We say that two types $\mathbb{A}, \mathbb{B}$ of support $\mathbf{X} \to \mathbf{Y}$ and $\mathbf{X}' \to \mathbf{Y}'$ are of *disjoint support* when $\mathbf{X}, \mathbf{X}', \mathbf{Y}, \mathbf{Y}'$ are pairwise disjoint.

Definition 5.10. Let $\mathbb{A}, \mathbb{B}$ be types of disjoint supports $\mathbf{X} \to \mathbf{Y}$ and $\mathbf{X}' \to \mathbf{Y}'$. We define

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

of respective supports $\mathbf{X} \times \mathbf{X}' \to \mathbf{Y} \times \mathbf{Y}$ and $\mathbf{Y} \times \mathbf{X}' \to \mathbf{X} \times \mathbf{Y}$.

The following theorem then establishes that multiplicative connectives are adequately interpreted. We omit the proof which is standard [Sei12a, Sei16b, Sei17].

Theorem 5.11. *Let $\mathbb{A}$ and $\mathbb{B}$ be types of disjoint supports $\mathbf{X} \to \mathbf{Y}$ and $\mathbf{X}' \to \mathbf{Y}'$. We have:*

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

5.2. **Additives and Quantifiers.** To represent additives, one uses the notion of state: the additive conjunction & superposes two proof-objects of the same support whose sets of states S and S' by creating a proof-object with set of states $S + S'$.

Notations 5.12. We write $\mathfrak{o}_{\mathbf{X} \to \mathbf{Y}}$ the proof-object $(1, \mathbf{0}_{\mathbf{X} \to \mathbf{Y}})$ where 1 is a the constant function equal to 1 and $\mathbf{0}_{\mathbf{X} \to \mathbf{Y}}$ is the zero sub-Markov kernel from $\mathbf{X}$ to $\mathbf{Y}$, i.e. $\mathbf{0}(x, \_)$ is the constant 0 subprobability distribution.

Notations 5.13. Let A, B be kernels respectively from $\mathbf{X}$ to $\mathbf{Y}$ and from $\mathbf{X}$ to $\mathbf{Y}$. We write $A \,\&\, B$ the kernel κ from $\mathbf{X}$ to $\mathbf{Y}$ defined as $\kappa(x, y) \cdot (e, \dot{f}) = A(x, y) \cdot (2e, \dot{f})$ if $0 \leqslant e \leqslant 1/2$ and $\kappa(x, y) \cdot (e, \dot{f}) = B(x, y) \cdot (2e - 1, \dot{f})$ otherwise.

Definition 5.14. If $\mathfrak{a} = (a, A)$ and $\mathfrak{b} = (b, B)$ are proof-objects of support $\mathbf{X} \to \mathbf{Y}$, we define $\mathfrak{a} \,\&\, \mathfrak{b} = (a + b, A \,\&\, B)$ of support $\mathbf{X} \to \mathbf{Y}$.

Additives are defined on a subset of types that never allow for weakening called *behaviour* in earlier work [Sei19]; we here use the terminology *purely linear types*.

Definition 5.15. A type $\mathbb{A}$ has the expansion property when $\forall \mathfrak{a} \in \mathbb{A}, \mathfrak{a} \,\&\, \mathfrak{o}_{\mathbf{X} \to \mathbf{Y}} \in \mathbb{A}$. A type $\mathbb{A}$ is *purely linear* if both $\mathbb{A}$ and $\mathbb{A}^{\perp}$ have the expansion property.

Definition 5.16. Let $\mathbb{A}, \mathbb{B}$ be purely linear types of disjoint supports $\mathbf{X} \to \mathbf{Y}$ and $\mathbf{X}' \to \mathbf{Y}'$. We define the following purely linear types of support $\mathbf{X} \times \mathbf{X}' \to \mathbf{Y} \times \mathbf{Y}'$:

$$\mathbb{A} \oplus \mathbb{B} = (\{\mathfrak{a} \otimes \mathfrak{o}_{\mathbf{X}' \to \mathbf{Y}'} \mid \mathfrak{a} \in \mathbb{A}\} \cup \{\mathfrak{o}_{\mathbf{X} \to \mathbf{Y}} \otimes \mathfrak{b} \mid \mathfrak{b} \in \mathbb{B}\})^{\perp\perp},$$
$$\mathbb{A} \,\&\, \mathbb{B} = \{(\mathfrak{a} \otimes \mathfrak{o}_{\mathbf{X}' \to \mathbf{Y}'}) \,\&\, (\mathfrak{o}_{\mathbf{X} \to \mathbf{Y}} \otimes \mathfrak{b}) \mid \mathfrak{a} \in \mathbb{A}, \mathfrak{b} \in \mathbb{B}\}^{\perp\perp}.$$

Definition 5.17. We define (support-wise) second-order quantification as the following operations on types (not necessarily purely linear):

$$\forall_{\mathbf{X} \to \mathbf{Y}} \mathbb{X}\ \mathbb{F}(\mathbb{X}) = \bigcap_{\mathbb{A} \text{ of support } \mathbf{X} \to \mathbf{Y}} \mathbb{F}(\mathbb{A})$$
$$\exists_{\mathbf{X} \to \mathbf{Y}} \mathbb{X}\ \mathbb{F}(\mathbb{X}) = \left(\bigcup_{\mathbb{A} \text{ of support } \mathbf{X} \to \mathbf{Y}} \mathbb{F}(\mathbb{A}) \right)^{\perp\perp}$$

These definitions of additives and quantifiers, together with the interpretation of multiplicatives explained in the previous section, allow to interpret second-order multiplicative additive linear logic (MALL^2); proofs follow closely the author's previous work [Sei17].

5.3. **Exponentials.** We redefine exponentials for balanced proof-objects.

Definition 5.18. Let $\mathfrak{f} = (1, \kappa)$ be a balanced general proof-object of support $\mathbf{X} \to \mathbf{Y}$. We define $!\mathfrak{f}$ as the general proof-object $(1, !\kappa)$ 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 5.19 (Exponentiation)**.** Let $\mathfrak{f} = (1, \kappa_{\mathrm{F}})$ be a balanced proof-object. We define its *exponential* $!\mathfrak{f} = (1, !\kappa_{\mathrm{F}})$.

We will now show that the exponential principles of linear logic can be interpreted faithfully.

Notations 5.20. 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 $(\mathbf{X} \times [0,1]) \times (\mathbf{Y} \times [0,1])$ (i.e. the kernel *from* $\mathbf{X}$ *to* $\mathbf{Y}$) defined as $\bar{\kappa}(x; \dot{y}) \cdot (e; \dot{f}) = \kappa(x, \dot{y}) \mathbf{1}(e, \dot{f})$.

The following lemma, established by the author [Sei19, 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 $\mathbb{A}$.

Lemma 5.21. *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 5.22. *The digging rule can be interpreted.*

Proof of Proposition 5.22. Suppose $\mathbb{A}$ is of support $\mathbf{X} \to \mathbf{Y}$. This map is easily implemented as a project $\mathfrak{dig}_{\mathbf{X} \to \mathbf{Y}} = (1, \kappa_{\mathrm{dig}_{\mathbf{Y}}} + \kappa^*_{\mathrm{dig}_{\mathbf{X}}})$ with

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

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

$$
\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}{r} \kappa^{*}_{\mathrm{dig}_{\mathbf{X}}}((x,e,f),(\dot{y},\dot{u})) \cdot (g,\dot{d}) \mathbf{1}(h,\dot{d'}) \\ \times !\kappa_{\mathrm{A}}((y,u);(\dot{z},\dot{v})) \cdot (d',\dot{c'}) \mathbf{1}(d,\dot{c}) \\ \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}{r} !\kappa_{\mathrm{A}}((y,u);(\dot{z},\dot{v})) \cdot (d',\dot{c'}) \mathbf{1}(d,\dot{c}) \\ \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')} !\kappa_{\mathrm{A}}((x,e);(\dot{z},\dot{v})) \cdot (h,\dot{c'}) \mathbf{1}(\varphi(f,g),\dot{c}) \times \kappa_{\mathrm{dig}}((z,v),(\dot{x},\dot{e},\dot{f})) \cdot (c,\dot{g}) \mathbf{1}(c',\dot{h}) \\
&= \int_{(z,v,c,c')} !\kappa_{\mathrm{A}}((x,e);(\dot{z},\dot{v})) \cdot (h,\dot{e}) \mathbf{1}(\varphi(f,g),\dot{e'}) \times \mathbf{1}(z,\dot{x}) \mathbf{1}(v,\dot{e}) \mathbf{1}(c,\varphi(\dot{f},\dot{g})) \mathbf{1}(c',\dot{h}) \\
&= !\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})) \\
&= !\kappa_{\mathrm{A}}((x,e);(\dot{x},\dot{e})) \cdot (f,\dot{f}) \mathbf{1}(\varphi(h,g),\varphi(\dot{h},\dot{g})) \\
&= !!\kappa_{\mathrm{A}}((x,e,f);(\dot{x},\dot{e},\dot{f})) \cdot (\varphi(h,g),\varphi(\dot{h},\dot{g}))
\end{aligned}
$$

Figure 6: Computation from the proof of Proposition 5.22.

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 3 paths". The computation in Figure 6 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 5.21. □

Proposition 5.23. *The dereliction rule can be interpreted.*

Proof of Proposition 5.23. 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 $\mathfrak{der} = (1, \kappa^{*}_{\mathrm{der}_{\mathbf{X}}} + \kappa_{\mathrm{der}_{\mathbf{Y}}})$ with:

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

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

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

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

Figure 7: Computation from the proof of Proposition 5.23.

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 $!\mathfrak{a} = (1, !\kappa)$. Again, given the definition of $\mathfrak{der}$, this consists in computing paths of length 3. The computation shown in Figure 7 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 5.21 this is enough to establish that $\mathfrak{der}_{\mathbf{X}\to\mathbf{Y}}$ belongs to $!\mathbb{A} \multimap \mathbb{A}$. □

Proposition 5.24. *Functorial promotion holds.*

Proof of Proposition 5.24. 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 $\mathfrak{twist} = (1, \kappa_{\mathrm{twist}})$ with:

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

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

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

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

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

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

We now consider projects $!\mathfrak{a} = (1, !\kappa_{\mathrm{A}})$ and $!\mathfrak{f} = (1, !\kappa_{\mathrm{F}})$ of support $\mathbf{X}$ and $\mathbf{X} \times \mathbf{Y}$ respectively. A computation similar to those from the proofs of Proposition 5.22 and 5.23 (but more involved, as paths are not limited to length 3 here) show that $((!\kappa_{\mathrm{A}} :: \kappa_{\mathtt{l}}) :: \kappa_{\mathrm{twist}} ::(!\kappa_{\mathrm{F}} :: \kappa_{\mathtt{r}})) :: \kappa_{\mathtt{c}}$ is equal to $!\kappa_{\mathrm{A} :: \mathrm{F}}$. □

We now have stated all the key results needed to establish our main theorem, adapting previous interpretation of second-order linear logic sequent calculus [Sei16c].

Theorem 5.25. *General proof-objects and types define a sound model of* LL^2.

6. Perspectives

We established that sub-Markov processes provide a model of second-order linear logic. Probabilistic languages with sampling instructions should be interpretable in this model: 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 be considered. We expect strong connections with game semantics models dealing with such languages [CP19], although our approach differs from the start by its intention. In particular, the realisability approach provides a very rich notion of types arising from the behaviour of processes. This can incorporate dependent types [Gir11], and could be used to consider new type constructions adapted to probabilistic computation [NPS].

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 [Sei15, Sei18b]. 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 Seiller and Pellissier established using the framework of graphings that strong algebraic lower bounds can be obtained using *topological entropy* [SPL22]. The current 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.

References

[AJ94] Samson Abramsky and Radha Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59(2):543–574, 1994. doi:10.2307/2275407.

[AM65] Michael Artin and Barry Mazur. On periodic points. *Annals of Mathematics*, pages 82–99, 1965.

[BBC98] Stefano Berardi, Marc Bezem, and Thierry Coquand. On the computational content of the axiom of choice. *The Journal of Symbolic Logic*, 63(2):600–622, 1998. doi:10.2307/2586854.

[CCPW18] Simon Castellan, Pierre Clairambault, Hugo Paquet, and Glynn Winskel. The concurrent game semantics of probabilistic PCF. In Dawar and Grädel [DG18], pages 215–224. doi:10.1145/3209108.3209187.

[CP18] Pierre Clairambault and Hugo Paquet. Fully abstract models of the probabilistic lambda-calculus. In Dan R. Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018, Birmingham, UK*, volume 119 of *LIPIcs*, pages 16:1–16:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.CSL.2018.16.

[CP19] Simon Castellan and Hugo Paquet. Probabilistic programming inference via intensional semantics. In Luís 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. doi:10.1007/978-3-030-17184-1_12.

[Cur06] Pierre-Louis Curien. Introduction to linear logic and ludics, part II. *Advances in Mathematics (China)*, 35(1):1–44, 2006. doi:10.2307/1969645.

[dAKM+21] Pedro H. Azevedo de Amorim, Dexter Kozen, Radu Mardare, Prakash Panangaden, and Michael Roberts. Universal semantics for the stochastic λ-calculus. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–12. IEEE, 2021. doi:10.1109/LICS52264.2021.9470747.

[DE11] Vincent Danos and Thomas Ehrhard. Probabilistic coherence spaces as a model of higher-order probabilistic computation. *Inf. Comput.*, 209(6):966–991, 2011. doi:10.1016/j.ic.2011.02.001.

[DG18] Anuj Dawar and Erich Grädel, editors. *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*. ACM, 2018. doi:10.1145/3209108.

[DH02] Vincent Danos and Russell S. Harmer. Probabilistic game semantics. *ACM Trans. Comput. Logic*, 3(3):359–382, jul 2002. doi:10.1145/507382.507385.

[DK19] Yoann Dabrowski and Marie Kerjean. Models of linear logic based on the schwartz ε-product. *Theory and Applications of Cateogries*, 34:1440–1525, 2019.

[Ehr19] Thomas Ehrhard. Differentials and distances in probabilistic coherence spaces. In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany*, volume 131 of *LIPIcs*, pages 17:1–17:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.FSCD.2019.17.

[EPT18] Thomas Ehrhard, Michele Pagani, and Christine Tasson. Full abstraction for probabilistic PCF. *J. ACM*, 65(4):23:1–23:44, 2018. doi:10.1145/3164540.

[ES22] Boris Eng and Thomas Seiller. Multiplicative linear logic from a resolution-based tile system. 2022. arXiv:2207.08465, doi:10.48550/arXiv.2207.08465.

[ETP14] Thomas Ehrhard, Christine Tasson, and Michele Pagani. Probabilistic coherence spaces are fully abstract for probabilistic PCF. In Suresh Jagannathan and Peter Sewell, editors, *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 309–320. ACM, 2014. doi:10.1145/2535838.2535865.

[FK52] Bent Fuglede and Richard V. Kadison. Determinant theory in finite factors. *Annals of Mathematics*, 56(2), 1952.

[FM77a] Jacob Feldman and Calvin C Moore. Ergodic equivalence relations, cohomology, and von neumann algebras. I. *Transactions of the American mathematical society*, 234(2):289–324, 1977. doi:10.2307/1997924.

[FM77b] Jacob Feldman and Calvin C Moore. Ergodic equivalence relations, cohomology, and von neumann algebras. II. *Transactions of the American Mathematical Society*, 234(2):325–359, 1977. doi:10.2307/1997925.

[Gir87] Jean-Yves Girard. Multiplicatives. In Lolli, editor, *Logic and Computer Science : New Trends and Applications*, pages 11–34, Torino, 1987. Università di Torino. Rendiconti del seminario matematico dell'università e politecnico di Torino, special issue 1987.

[Gir88] Jean-Yves Girard. Geometry of interaction II: Deadlock-free algorithms. In *Proceedings of COLOG*, number 417 in Lecture Notes in Computer Science, pages 76–93. Springer, 1988. doi:10.1007/3-540-52335-9_49.

[Gir89a] Jean-Yves Girard. Geometry of interaction I: Interpretation of system F. In *In Proc. Logic Colloquium 88*, 1989. doi:10.1016/S0049-237X(08)70271-4.

[Gir89b] Jean-Yves Girard. Towards a geometry of interaction. In *Proceedings of the AMS Conference on Categories, Logic and Computer Science*, 1989.

[Gir95] Jean-Yves Girard. Geometry of interaction III: Accommodating the additives. In *Advances in Linear Logic*, number 222 in Lecture Notes Series, pages 329–389. Cambridge University Press, 1995. doi:10.1017/CBO9780511629150.017.

[Gir99] Jean-Yves Girard. Coherent banach spaces: A continuous denotational semantics. *Theor. Comput. Sci.*, 227(1-2):275–297, 1999. doi:10.1016/S0304-3975(99)00056-0.

[Gir01] Jean-Yves Girard. Locus solum: From the rules of logic to the logic of rules. *Mathematical Structures in Computer Science*, 11(3), 2001. doi:10.1007/3-540-44802-0_3.

[Gir03] Jean-Yves Girard. From foundations to ludics. *Bulletin of Symbolic Logic*, 9(2):131–168, Jun 2003. doi:10.2178/bsl/1052669286.

[Gir04] Jean-Yves Girard. *Between logic and quantic : a tract*, pages 346–381. Number 316 in London Mathematical Society Lecture Note Series. Cambridge University Press, 2004. doi:10.1017/CBO9780511550850.011.

[Gir06] Jean-Yves Girard. Geometry of interaction IV: the feedback equation. In Stoltenberg-Hansen and Väänänen, editors, *Logic Colloquium '03*, pages 76–117, 2006. doi:10.1017/9781316755785.006.

[Gir11] Jean-Yves Girard. Geometry of interaction V: Logic in the hyperfinite factor. *Theoretical Computer Science*, 412:1860–1883, 2011. doi:10.1016/j.tcs.2010.12.016.

[Gir16] Jean-Yves Girard. Transcendental syntax ii: non deterministic case. Logical Methods in Computer Science (to appear), 2016.

[Gir17] Jean-Yves Girard. Transcendental syntax i: deterministic case. *Mathematical Structures in Computer Science*, 27(5):827–849, 2017. doi:10.1017/S0960129515000407.

[Gir18] Jean-Yves Girard. Transcendental syntax iii: equality. preprint, 2018.

[Has97] Masahito Hasegawa. Recursion from cyclic sharing: Traced monoidal categories and models of cyclic lambda calculi. pages 196–213. Springer Verlag, 1997. doi:10.1007/978-1-4471-0865-8_7.

[HO00] John Martin Elliott Hyland and C.-H. Luke Ong. On full abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285–408, 2000. doi:10.1006/inco.2000.2917.

[HS06] Esfandiar Haghverdi and Philip Scott. A categorical model for the geometry of interaction. *Theoretical Computer Science*, 350(2):252–274, 2006. doi:10.1007/978-3-540-27836-8_60.

[Iha66] Yasutaka Ihara. On discrete subgroups of the two by two projective linear group over $\mathfrak{p}$ -adic fields. *J. Math. Soc. Japan*, 18(3):219–235, 07 1966. doi:10.2969/jmsj/01830219.

[JSV96] André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3):447–468, 1996. doi:10.1017/S0305004100074338.

[Ker18] Marie Kerjean. A logical account for linear partial differential equations. In Dawar and Grädel [DG18], pages 589–598. doi:10.1145/3209108.3209192.

[Kri01] Jean-Louis Krivine. Typed lambda-calculus in classical zermelo-fraenkel set theory. *Archive for Mathematical Logic*, 40(3):189–205, 2001.

[Kri09] Jean-Louis Krivine. Realizability in classical logic. *Panoramas et synthèses*, 27:197–229, 2009.

[Miq11] Alexandre Miquel. A survey of classical realizability. In C.-H. Luke Ong, editor, *Typed Lambda Calculi and Applications - 10th International Conference, TLCA 2011, Novi Sad, Serbia, June 1-3, 2011. Proceedings*, volume 6690 of *Lecture Notes in Computer Science*, pages 1–2. Springer, 2011. URL: http://dx.doi.org/10.1007/978-3-642-21691-6, doi:10.1007/978-3-642-21691-6_1.

[NPS] Alberto Naibo, Mattia Petrolo, and Thomas Seiller. Logical constants from a computational point of view. *In preparation.*

[NPS16] Alberto Naibo, Mattia Petrolo, and Thomas Seiller. On the computational meaning of axioms. In *Epistemology, Knowledge and the Impact of Interaction*, pages 141–184. Springer, 2016. doi:10.1007/978-3-319-26506-3_5.

[Pan99] Prakash 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. doi:https://doi.org/10.1016/S1571-0661(05)80602-4.

[Paq21] Hugo Paquet. Bayesian strategies: probabilistic programs as generalised graphical models. In Nobuko Yoshida, editor, *Programming Languages and Systems - 30th European Symposium on Programming, ESOP 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings*, volume 12648 of *Lecture Notes in Computer Science*, pages 519–547. Springer, 2021. doi:10.1007/978-3-030-72019-3_19.

[PW18] Hugo Paquet and Glynn Winskel. Continuous probability distributions in concurrent games. In Sam Staton, editor, *Proceedings of the Thirty-Fourth Conference on the Mathematical Foundations of Programming Semantics, MFPS 2018, Dalhousie University, Halifax, Canada, June 6-9, 2018*, volume 341 of *Electronic Notes in Theoretical Computer Science*, pages 321–344. Elsevier, 2018. doi:10.1016/j.entcs.2018.11.016.

[Rue76] David Ruelle. Zeta-functions for expanding maps and anosov flows. *Inventiones mathematicae*, 34(3):231–242, 1976. doi:10.1007/BF01403069.

[Sei12a] Thomas Seiller. Interaction graphs: Multiplicatives. *Annals of Pure and Applied Logic*, 163:1808–1837, December 2012. doi:10.1016/j.apal.2012.04.005.

[Sei12b] Thomas Seiller. *Logique dans le facteur hyperfini : géometrie de l'interaction et complexité*. PhD thesis, Université Aix-Marseille, 2012. URL: http://tel.archives-ouvertes.fr/tel-00768403/.

[Sei15] Thomas Seiller. Towards a complexity-through-realizability theory. 2015. URL: http://arxiv.org/abs/1502.01257, arXiv:1502.01257.

[Sei16a] Thomas Seiller. From dynamic to static semantics, quantitatively. 2016. URL: http://arxiv.org/abs/1604.05047, arXiv:1604.05047.

[Sei16b] Thomas Seiller. Interaction graphs: Additives. *Annals of Pure and Applied Logic*, 167:95 – 154, 2016. doi:10.1016/j.apal.2015.10.001.

[Sei16c] Thomas Seiller. Interaction graphs: Full linear logic. In *IEEE/ACM Logic in Computer Science (LICS)*, 2016. URL: http://arxiv.org/pdf/1504.04152.

[Sei17] Thomas Seiller. Interaction graphs: Graphings. *Annals of Pure and Applied Logic*, 168(2):278–320, 2017. doi:10.1016/j.apal.2016.10.007.

[Sei18a] Thomas Seiller. A correspondence between maximal abelian sub-algebras and linear logic fragments. *Mathematical Structures in Computer Science*, 28(1):77–139, 2018. doi:10.1017/S0960129516000062.

[Sei18b] Thomas Seiller. Interaction graphs: Non-deterministic automata. *ACM Trans. Comput. Log.*, 19(3):21:1–21:24, 2018. doi:10.1145/3226594.

[Sei19] Thomas Seiller. Interaction Graphs: Exponentials. *Logical Methods in Computer Science*, Volume 15, Issue 3, August 2019. doi:10.23638/LMCS-15(3:25)2019.

[SPL22] Thomas Seiller, Luc Pellissier, and Ulysse Léchine. On the power of euclidean division. lower bounds for algebraic machines, semantically. https://hal.archives-ouvertes.fr/hal-01921942, 2022.

[Ter10] Audrey Terras. *Zeta functions of graphs: a stroll through the garden*, volume 128. Cambridge University Press, 2010. doi:10.1017/CBO9780511760426.

[Ter11] Kazushige Terui. Computational ludics. *Theor. Comput. Sci.*, 412(20):2048–2071, 2011. doi:10.1016/j.tcs.2010.12.026.

[VO08] Jaap Van Oosten. *Realizability: an introduction to its categorical side*. Elsevier, 2008.