HABILITATION À DIRIGER DES RECHERCHES
DE
L'UNIVERSITÉ SORBONNE PARIS CITÉ

Spécialité Informatique

École doctorale n° 386 - Science Mathématiques Paris-Centre

Sémantiques des Calculs Distribués, Différentiels et Probabilistes

Mémoire d'habilitation à diriger des recherches présenté et soutenu publiquement par

Christine TASSON

à l'université Paris Diderot (Paris VII)

le 23 novembre 2018

Rapporteurs

M. Marcelo Fiore,	University of Cambridge
M. Pierre Fraigniaud,	CNRS - Université Paris VII
M. Prakash Panangaden,	McGill University

Jury

Mme Lisbeth Fajstrup,	Aalborg University
M. Marcelo Fiore,	University of Cambridge
Mme Delia Kesner,	Université Paris VII
Mme Alexandra Silva,	University College London
M. Tarmo Uustalu,	Reykjavík University

Président du Jury

M. Achim Jung,	University of Birmingham

Sémantiques des Calculs Distribués, Différentiels et Probabilistes

Résumé

Depuis les années 60, la sémantique s'est avérée très utile pour introduire des langages de haut niveau permettant d'écrire des programmes complexes et de les comprendre d'un point de vue mathématique. Dans les années 80, la logique linéaire a été introduite par Girard, reflétant des propriétés sémantiques liées à l'utilisation des ressources. Cette direction a été poursuivie par Ehrhard dans les années 2000 avec l'introduction du λ-*calcul différentiel*. Dans ses modèles, les programmes sont approximés par des polynômes, dont les monômes représentent les appels d'un programme à ses entrées lors de son exécution. Cette approche analytique a constitué un outil crucial pour l'étude des propriétés quantitatives apparaissant dans les langages de *programmation probabiliste*. En parallèle, depuis les années 90, plusieurs modèles géométriques ont été développés pour représenter des traces d'exécution dans les *systèmes distribués*. Dans cette thèse d'habilitation, nous présentons des modèles que nous avons étudiés dans ces trois domaines : les systèmes distribués, le λ-calcul différentiel, la programmation probabiliste, ainsi que les techniques générales nécessaires et les résultats qu'ils nous ont permis d'obtenir. Celles-ci ont nécessité l'utilisation et le développement d'outils issus de la combinatoire, de topologie dirigée, d'analyse fonctionnelle, de théorie des catégories et de probabilités.

Models of Distributed, Differential and Probabilistic Computation

Abstract

Since the 60s, semantics have proved most useful to introduce high-level languages allowing complex programs to be written and at an accurate mathematical level to be understood. In the 80s, *linear logic* was introduced by Girard following semantical properties reflecting resource usage in computation. This line of research was carried on by Ehrhard in the 2000s with the introduction of *differential* λ-*calculus*. In its models, programs are approximated by polynomials whose monomials represent queries to inputs along computation. This analytic approach has been a key tool to study quantitative properties of programs such as *probabilistic computation*. In parallel, since the 90's, several geometric models was developed to give account to execution traces in *distributed systems*. In this habilitation thesis, we present models that we have studied in three distinct areas: distributed systems, differential lambda calculus, probabilistic programing, as well as general techniques to do so and the results they have allowed us to obtain. Those have required the use and the development of tools coming from combinatorics, directed topology, functional analysis, category theory and probability.

Contents

Introduction

Contents

0.1 Prelude

This thesis presents my recent research journey, often going in collaborations with other searchers. It deals with semantics of programming languages with different paradigms: distributive systems (Chapter 1), differential linear logic (Chapter 2) and probabilistic programming (Chapter 3). The common methodology that drives this thesis is to give a formal description of the meaning of programs independently from their syntax. Programs have to implement algorithms and computations correctly, that is to correspond to their specification. Semantics designs the tools necessary to describe formally this correspondence between a program and the specification of the algorithm that it implements. For this, *operational semantics* is used to describe behavior of programs using rewriting systems and *denotational semantics* is used to describe programs by mathematical functions that act upon spaces representing input and outputs.

There are several goals to this semantical approach: first to prove properties of programs in programming languages using mathematical tools. This is the very first step towards certification of programs. Perhaps a less common purpose is to give computational meaning to mathematical constructions and to derive new programming languages that reflect this constructions. This process led to the introduction of linear logic and of differential linear logic: In the 1980's, Girard realized that in a model he was studying, functions that interpret programs followed all a decomposition into a linear function and an exponential operator; these two key ingredients where used to decompose classical logic and to introduce linear logic Girard [1987]. In the 2000's, semantics of linear logic were built upon topological vector spaces of sequences. In these models functions where smooth (and even formal power series). The differential operator that exists in those semantics was given: a syntactical counterpart by the introduction of differential lambda calculus and differential linear logic; a computational meaning related to the number of queries to the inputs to get a given output Ehrhard and Regnier [2003]; Ehrhard [2018].

Along this manuscript I present different mathematical tools that we have used to denote programs and their executions. The first tools have a *geometric* flavor: in Chapter 1, execution traces of processes in asynchronous distributed systems are represented either as simplices in combinatorial spaces Herlihy et al. [2013] or as continuous directed paths in directed topological spaces Fajstrup et al. [2016]. The second tools come from *functional analysis* and *combinatorics*: in Chapter 2, programs are interpreted as smooth maps between topological spaces of infinite dimension or as formal power series. The last area is probability: in Chapter 3, programs are interpreted as measurable functions between measurable spaces. All the three approaches rely on an approximation theory through finite representations of programs that are well chosen to reason and prove their properties. For instance in Chapter 3, one of the main result (Full Abstraction) relies on the approximation of the interpretation of programs by well chosen polynomials whose degree is related to the number of times the inputs are queried. Presenting programs as formal power series (or approximating them as polynomials) is the subject of differential linear logic (presented in Chapter 2). The differential operator that extends linear logic, can be understood operationally as a way to produce the best linear approximation of programs (the version of the program that query once its input to get its output). The power series approximation can be computed by iterating the differential operator. In Chapter 1, the geometrical structure used by Herlihy

and his coauthors to represent execution traces of distributed protocols gives another finite representation of traces. The project research that I want to follow is to relate this approximate representation of distributed protocols with the approximation theories that was developed in differential linear logic. It will pave the way to extend results in distributed computing to probabilistic distributed computing. I will present this research project in more details at the end of this introduction (see Paragraph **??**).

The three chapters constituting this manuscript can be read independently, they are preceded by an introduction that present the state of the art, my contributions in the area and the main results presented in the chapter. The content of the chapters are of different nature. Chapter 1 is an extract of an article, resulting from a collaboration with Éric Goubault and Samuel Mimram Goubault et al. [2015a]. The operational semantics of asynchronous distributed systems and different denotational semantics based on geometric spaces are presented. Chapter 2 presents an unpublished work in collaboration with Martin Hyland. It deals with categorical theory of linear and non-linear substitution. It is a first step towards a new presentation of the differential operator at play in differential Linear Logic. Chapter 3 gives a panorama of the results that we obtained with Thomas Ehrhard and Michele Pagani about semantics of probabilistic programming languages handling both discrete or continuous probabilities at ground types. My contributions and the results presented in this manuscript are summarized in Paragraph 0.3.

0.2 Scientific Context

A **distributed system** is composed of computing entities, called processes, that *communicate* through *primitives* (e.g. send/receive messages, or update/scan a shared memory) with the purpose of jointly solving a task. Even in the deterministic case, processes have to handle uncertainties for two reasons. First, in an *asynchrounous* setting, processes execute their algorithm locally and at their own pace, and thus cannot rely on the progress of the others. Second, some processes may be faulty, either by *crash failure* (they are delayed or stopped) or by *Byzantine failure* (they are under malicious control). Each process has no information on the state of other processes and thus have to base its decisions on its partial knowledge of the global system (that it infers from the scheduling of the *communication primitives*).

A *task* describes globally the initial and the expected final states of the processes. For instance, one of the very fundamental problem in distributed computing is **the consensus task**: *Processes start with an initial value, for instance their identifier. To reach a consensus, the final values have first to be identical and second to belong to the set of initial values.* Given a communication procedure, *solving a task* consists in the design of one algorithm that each process will run independently. This algorithm precises the execution trace (that is the computation and communication steps) of the process that runs it. Now, considering the system on its whole, scheduling of communication's primitives have to be composed with execution traces. Because the system is asynchronous, this amounts to take into account interleavings of local process traces. *A **solution is correct** when for any set of initial values, for any interleaved execution trace (whatever asynchronicity and failures can be), the set of final values of correct processes satisfies the specification of the task.* Because of their fast growing number with respect to the number of communication rounds, it is not possible to explore every interleaved execution trace, thus the need for formal methods to characterize task solvability.

In the 1990s, two independent lines of research emerged, aiming to understand the limits of distributed systems. The papers in the 1980s were teeming with impossibility results of various kinds like the celebrated *impossibility theorem of distributed consensus* in asynchronous systems if there is even one faulty processor Fischer et al. [1985]. It seemed tempting to think that these numerous impossibility results could be viewed as topological obstructions. To formalize this intuition, Goubault and his co-workers developed a concurrent semantics based on directed algebraic topology Fajstrup et al. [2016, 2006]. Herlihy and coworkers independently showed that there were topological obstructions to the existence of certain protocols Herlihy and Shavit [1999]; Herlihy et al. [2013]. By representing finite execution traces as a combinatorial structure, the so-called **protocol complex** (see Fig. 1 for three processes and one round of communication), he obtained a characterization of task solvability in crash-tolerant (deterministic) distributed systems in terms of connectivity.

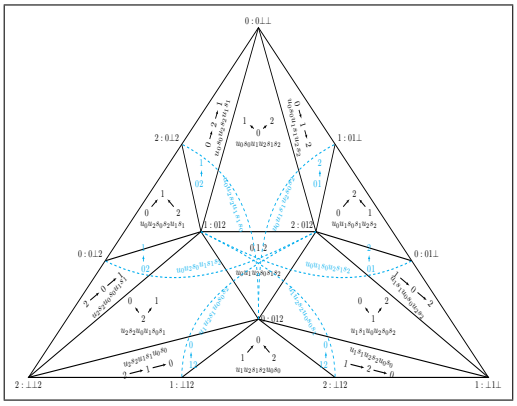


Figure 1 – Protocol complex

A *scheduling* of the communication primitives is formalized in a protocol complex by a *simplex* (such as a triangle on Fig. 1) whose vertices are the number of non-faulty processes. Moreover, two simplices are joined at their border if the processes on this border share the same knowledge of the global system. The protocol complex may be hard to compute from the description of the distributed system and is often produced by hand, hence the need for an adequate formalization of distributed protocols from which the protocol complex would be automatically derived. With Goubault and Mimram, we made a first step towards automatizing the derivation of protocol complexes by introducing distributed computing to operational and denotational semantics Goubault et al. [2018]. This work applied to *deterministic* protocols. It is the subject of Chapter 1.

The purpose of *semantics* in the area of **programming languages** is to describe what programs compute and how. The celebrated Curry-Howard-Lambek correspondence is at the heart of the design of programming languages ensuring correctness *by construction*. The "Curry-Howard" approach relates programs (of λ-calculus, the core of functional languages) and proofs (of natural deduction) via a one-to-one correspondence, the types of programs corresponding to formulas Howard [1980]. Generalizing Milner's slogan that "well-typed programs cannot go wrong", types are seen as invariants of computation. The third part of the correspondence (due to Lambek) relates proofs and programs with structured categories (cartesian closed categories in the λ-calculus case). It stresses the importance of compositionality for the properties of interest. Such categorical axiomatizations often result from the study of semantics for programming languages, which offers "concrete" descriptions of the program invariants, from which the categorical axiomatics can be extracted.

The methodology behind this thesis is the back-and-forth interactions between programs/proofs and semantics, resulting in the design of programming/logical constructions which reflect the properties of the semantics and provide **a computational content to mathematical constructions**.

Semantics originates in the work of Scott and Strachey [1971] whose motivation was the growing difficulty in proving (and even expressing) properties of programs and programming languages based only on their syntax. The idea is to interpret programs as functions, called *denotations*, between mathematical structures, in order to abstract away from the specificity of syntax and to give new, solid tools to reason about programs. There are three main goals: (1) **operational semantics** giving rigorous definitions of the operational behavior of programs that can serve as an implementation-independent reference; (2) providing formal methods to compare the expressiveness of programming languages independently of their syntax; and (3) **denotational semantics** supplying mathematical tools for proving computational properties of programs.

Quantitative semantics is a flourishing research area with deep results and constructions. They have in particular enabled the construction of models of computational cost (time and space consumption) Laird et al. [2013], of higher-order quantum functional programming Pagani et al. [2014], and of probabilistic programming with continuous data types Ehrhard et al. [2018b] (which is the first one combining higher-order, recursion and measurability). **Formal series** have emerged as a useful tools to encode input queries in the λ-calculus, the core of functional languages. More precisely, programs can be seen as a superposition of monomials representing finite approximations of the computation. The degree of those monomials is related to the number of times an argument is used by the program Ehrhard and Regnier [2003]. This approximation theory is called *syntactic Taylor expansion* Ehrhard and Regnier [2006a]. Seen through the Curry-Howard correspondence, the Taylor expansion has a counterpart in intuitionistic logic: proofs can be interpreted as power series and their derivatives provide linear approximations of them. An early version of this observation led Girard to introduce **Linear Logic** (LL) Girard [1987] and later Ehrhard and Regnier to introduce the **differential λ-calculus** Ehrhard and Regnier [2003]. Both these systems emerged after a careful study of denotational models based on power series: one was based on normal functors Girard [1988] (power series whose coefficients are sets) and the other was based on analytic functions over topological vector spaces Blute et al. [2012]; Ehrhard [2005]; Kerjean and Tasson [2018].

Quantitative semantics completes the Curry-Howard correspondence by involving linear algebra, computer science and proof theory. Using methods from linear algebra and analysis in these quantitative approaches has allowed to solve problems which were left open in the traditional semantics based on domains.

LL is characterized by an involutive negation and a pair of dual, *exponential* modalities (denoted by ! and ?) which regulate the (explicit) use of structural rules: contraction and weakening. Such a control over contraction and weakening brings an analogue of the notion of resource to proof theory. This allows to consider quantitative properties of proofs. In particular, LL allows the decomposition of the type $A \Rightarrow B$ into more primitive connectives: $A \Rightarrow B = !A \multimap B$. Here, the arrow $A \multimap B$ denotes the space of the linear functions (from a semantic point of view), and the type of the programs using their

resources exactly once (from an operational point of view). The type $!A$ denotes the space of the resources of type A that can be used at will, semantically $!A$ is in some sense analogous to the space of the power series with formal indeterminates over A. The third part of the Curry-Howard-Lambek correspondence, the categorical axiomatization, for LL can be formalized by Linear-Non-Linear categories as presented in Melliès [2009]. In those categorical models, there is a well-behaved adjunction between a symmetric monoidal category $\mathbb{L}$ and a cartesian one $\mathbb{M}$, allowing the exponential modality ! to be interpreted as the comonad.

In order to be able to take quantitative properties into account, it is natural to interpret programs as analytic functions. In particular, these functions are infinitely differentiable. A crucial issue is then to understand whether differentiation is a meaningful syntactic operation. A first positive answer was given with Ehrhard and Regnier's *differential* λ*-calculus* Ehrhard and Regnier [2003] and their graphical calculus counterpart *differential nets* Ehrhard and Regnier [2006b]. Both of these extensions feature two different ways of passing inputs to programs: the usual, *unrestricted* one, and a new, *linear* one. The latter one consists in defining the derivative $\mathtt{D}f \cdot x$ of the program f, seen as a function, on the argument x. The evaluation of $\mathtt{D}f \cdot x$ has a precise operational meaning: it corresponds to passing the input x to f *exactly once*. This imposes non-deterministic choices: if f contains several subroutines each of them demanding for a copy of x, then there are different evaluations of $\mathtt{D}f \cdot x$, depending on which subroutine is fed with the unique available copy of x. We thus need a formal sum, where each summand represents a choice. Such sums express syntactically the addition between vectors in the quantitative semantics, and have a canonical mathematical interpretation — they correspond to the sums that one obtains when computing the derivative of a product of functions. The derivative operator allows to compute syntactically the optimal approximation of a program when applied to linear arguments.

As expected, iterated differentiation yields a natural notion of linear approximation of the ordinary application of a program to an input. This notion relates to ordinary application through the *Taylor formula* Ehrhard and Regnier [2008]. More generally, if one fully develops each application occurring in a program into its corresponding Taylor expansion, the result is an infinite sum of purely "differential programs" that contain only (multi)linear applications and applications to 0.

In **combinatorics**, formal series are used to compute asymptotic properties. Generating functions are one of the most efficient tools to count labelled structures (trees, lists, cycles, ...) by solving the integro-differential equations that define these structures: those are power series such that the coefficient of their n-th monomial is the number of structures with n labels. *Species* have been designed and developed to describe combinatorial structures such as protocol complexes and *operads* to compose these structures. Briefly, an operad is given by an underlying species of operations, together with a unital "multiplication" which describes composition of operations. Each operad thus generates a monad, whose algebras are precisely the desired algebraic structures. In abstract algebra, species have been analysed through higher categories, unifying combinatorics and semantics of LL. These *generalized species* Fiore et al. [2008]; Fiore et al. [2016] are the 2-categorical version of Joyal's combinatorial species generated by the 2-monad of free symmetric monoidal categories. Thus defined, they encode an algebraic theory of linear substitution: composition of generalized species covers the substitution monoidal structure of coloured operads Baez and Dolan [1995] and the linear substitution founding LL. Naturally, generalized species have been used to represent resource terms appearing in the syntactical Taylor expansion Tsukada et al. [2017, 2018]. Even the non-linear substitution of λ-calculus has been formalised as an algebraic theory of cartesian operads Hyland [2017, 2014]. How to combine the linear and the non-linear substitution monads, in the spirit of LL, is the subject of Chapter 2.

Quantitative semantics can be applied to study randomized algorithms. How to implement, formalize and reason on such algorithms is the subject of **probabilistic programming languages** Borgström et al. [2016]; Goodman and Tenenbaum [2014]. This line of research has gained a great interest over the past few years, with the development of probabilistic models used in areas such as artificial intelligence, applied statistics and machine learning. What we call **probabilistic semantics** is the mathematical foundations of the computation of probabilistic programs with an emphasis on composition. In this area, Kozen has developed a measure-theoretic semantics, using *Markov kernels* to describe the *operational semantics* of probabilistic programs Kozen [1981]. Then, building on the work of Giry [1982] and Moggi [1989], Jones and Plotkin [1989] introduced a *denotational semantics* based on probabilistic power domains, which led to rich research programs by Di Gianantonio and Edalat [2013], Jung and Tix [1998], Keimel and Plotkin [2017], among others. Recently, Kozen and Silva used this approach based on domains to model networks with probabilistic features Smolka et al. [2017].

In parallel, Danos and Ehrhard introduced probabilistic coherent spaces, a *denotational semantics*, where *formal series* denote programs Danos and Ehrhard [2011]. Together with Ehrhard and Pagani, we proved that this model is precise enough to statically characterize the computational behavior of

programs handling discrete probabilities Ehrhard et al. [2014]. On a more practical probabilistic side, probabilistic programming languages, including Church, Anglican, Stan, Tabular, Venture, Hakaru, have been introduced for applications in probabilistic models by composing small blocks available in the language releasing searchers from the representation problematic. Basic probabilistic constructors such as sampling and more complex tools for implementing statistical inference are both construction of these languages where continuous distributions constitute the key object. Most of them are *higher-order* functional programming languages: they allow writing programs that can handle programs as arguments. To formalize this paradigm, it is necessary to design a cartesian closed category rich enough for probabilistic semantics. Probabilistic coherent spaces constitute a cartesian closed category, but handling only discrete ground types such as natural numbers. It is only recently that *denotational semantics* of higher-order languages with continuous ground types, such as reals, have been designed. Indeed, there is a major obstacle Panangaden [1999]: Markov kernels and measurable functions are not suitable for interpreting *composition* of functional probabilistic programs. To our knowledge, there are only two models of probabilistic programming handling higher-order and continuous probabilities: quasi-Borel spaces designed by Staton and his collaborators Heunen et al. [2017]; Scibior et al. [2018] and the measurable cones Ehrhard et al. [2018b]. This work is related in Chapter 3. Semantical reasoning on higher-order programs and compositionality has been advocated from an equational viewpoint Bacci et al. [2018] and from a categorical viewpoint Dahlqvist et al. [2016]; Kozen [2016].

0.3 Contributions

Chapter 1 is devoted to my contributions to the semantics of distributed systems following the line of the combinatorial model proposed by Herlihy et al. [2013]. With my coauthors, we studied asynchronous Byzantine systems (where processes can send fake messages) in Mendes et al. [2014] and gave a characterization of distributed solvability in this context. We also gave in Goubault et al. [2015b] a combinatorial proof of the collapsibility of the crash failure protocol complex which is a key feature that implies impossibility results in fault-tolerant distributed computing.

In order to compute protocol complexes appearing in these combinatorial models, we introduced distributed systems to operational semantics in Goubault et al. [2015a] and Goubault et al. [2018]. This semantical approach allows us to give a one-to-one correspondence between several semantical approaches to the representation of traces of executions in distributed systems. More precisely, Theorem 10 describes formally the relation between the protocol complex approach of Herlihy and his coauthors and the directed topology approach of Goubault and his coauthors. The proof reveals the importance of the local view of the processes on the global state of the system that is encoded in the operational semantics as presented in Theorem 18.

Chapter 2 tackles semantics of differential linear logic. In this area, I have followed three directions. First, I have studied and used finiteness spaces introduced in Ehrhard [2005]: in Tasson and Vaux [2018] we studied type fixpoints; in Pagani et al. [2016], we generalized Ehrhard [2010] and used finiteness spaces to characterize strongly normalizing terms of non-deterministic λ-calculus. Second, I studied models of differential linear logic based on topological vector spaces in which the differential operator of the logic is interpreted as the derivation in calculus and proofs (or programs) are interpreted as smooth functions in Blute et al. [2012]. Then, because formal series are a key notion to approximate programs. We refined smooth functions into formal series in Kerjean and Tasson [2018].

In Chapter 2, I present a third contribution that has been developed in collaboration with Martin Hyland. As suggested in Fiore [2006]; Power and Tanaka [2005], we propose to combine the operadic axiomatizations of linear substitution of Fiore et al. [2016] (given by the symmetric monoidal monad) and non-linear substitution of Hyland [2017, 2014] (given by the cartesian monoidal monad). There are many known ways to combine monads, for instance using distributive laws. Unfortunately, none of these constructions is suitable for our setting and we investigate a new one based on colimits. We first show in Theorem 32 that our construction is correct: the colimit of monads produces a monad. Then, in Theorem 35 we give a characterization of the algebras of the produced monad. It is a first step towards an algebraic understanding to linear-non-linear substitution (we still need to understand how to lift the produced monad to profunctors that is needed to interpret terms).

Chapter 3 gives a panorama of the results we had on semantical aspects of probabilistic programming. The key result we proved was the full abstraction for probabilistic coherent spaces and probabilistic PCF Ehrhard et al. [2014, 2018a] and then probabilistic Call-By-Push-Value Ehrhard and Tasson [2018]. Theorems 79 and 88 tell us that the semantics of probabilistic coherent spaces characterizes the behavioral equivalence of probabilistic functional programs. This key property is obtained thanks to the polynomial

approximation of terms. Indeed, probabilistic coherent spaces is a semantics based on linear logic and programs are interpreted as formal series. The full abstraction problem reduces to a problem of real analytic functions. We further studied probabilistic coherent spaces proving in Theorem 64 that the probability that a pure λ-term reduces to a head normal form is equal to its denotation computed on a suitable set of values Ehrhard et al. [2011] ; and proving in Proposition 44 that the exponential comonad of Probabilistic Coherent Spaces is free Crubillé et al. [2017].

In Ehrhard et al. [2018b], we moved from discrete probabilities to continuous ones. We proved that the cartesian closed category of measurable cones and stable, measurable functions is adequate for a version of Probabilistic PCF with the single ground type of reals. Moreover, we gave a bunch of example to prove that this semantics is well-suited to interpret simple randomized algorithms in Paragraph 3.5.6: standard distributions, conditioning, expectation approximation through Monte-Carlo methods and inference algorithms as Metropolis-Hastings (see Example 107). We encoded this algorithms by programs in real probabilistic PCF. We computed their semantics and proved that the encoding are correct by using the Adequacy Theorem 102.

0.4 Bibliography

Giorgio Bacci, Robert Furber, Dexter Kozen, Radu Mardare, Prakash Panangaden, and Dana Scott. Boolean-valued semantics for the stochastic λ-calculus. In *LICS*, pages 669–678. ACM, 2018. 5

J. C. Baez and J. Dolan. Higher-dimensional algebra and topological quantum field theory. *J. Math. Phys.*, 36(11), 1995. 4

Richard Blute, Thomas Ehrhard, and Christine Tasson. A convenient differential category. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 53(3):211–232, 2012. 3, 5

Johannes Borgström, Ugo Dal Lago, Andrew D. Gordon, and Marcin Szymczak. A lambda-calculus foundation for universal probabilistic programming. In Jacques Garrigue, Gabriele Keller, and Eijiro Sumii, editors, *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, ICFP 2016, Nara, Japan, September 18-22, 2016*, pages 33–46. ACM, 2016. 4

Raphaelle Crubillé, Thomas Ehrhard, Michele Pagani, and Christine Tasson. The Free Exponential Modality of Probabilistic Coherence Spaces. In J. Esperanza and A. Murawski, editors, *Proceedings of the 20th International Conference on Foundations of Software Science and Computation Structures, FOSSACS 2017*. ARCoSS, 2017. 6

Fredrik Dahlqvist, Vincent Danos, and Ilias Garnier. Giry and the machine. *Electr. Notes Theor. Comput. Sci.*, 325:85–110, 2016. 5

V. Danos and T. Ehrhard. Probabilistic coherence spaces as a model of higher-order probabilistic computation. *Inform. Comput.*, 2011. doi: 10.1016/j.ic.2011.02.001. 4

P. Di Gianantonio and A. Edalat. A language for differentiable functions. In *Foundations of Software Science and Computation Structures*, pages 337–352, 2013. ISBN 978-3-642-37075-5. 4

T. Ehrhard. Finiteness spaces. *Math. Struct. Comput. Sci.*, 15(4), 2005. 3, 5

T. Ehrhard. A finiteness structure on resource terms. In *LICS*, pages 402–410. IEEE Computer Society, 2010. 5

T. Ehrhard and L. Regnier. The differential lambda-calculus. *Theor. Comput. Sci.*, 309(1), 2003. 1, 3, 4

T. Ehrhard and L. Regnier. Böhm trees, Krivine's Machine and the Taylor Expansion of Lambda-Terms. In *CiE*, 2006a. 3

T. Ehrhard and L. Regnier. Differential interaction nets. *Theoretical Computer Science*, 364(2):166–195, 2006b. 4

T. Ehrhard and L. Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theor. Comput. Sci.*, 403(2-3), 2008. 4

Thomas Ehrhard. An introduction to differential linear logic: proof-nets, models and antiderivatives. *Mathematical Structures in Computer Science*, 28(7):995–1060, 2018. 1

Thomas Ehrhard and Christine Tasson. Probabilistic call by push value. *Logical Methods in Computer Science*, 2018. Accepted for publication, LMCS. 5

Thomas Ehrhard, Michele Pagani, and Christine Tasson. The computational meaning of probabilistic coherence spaces. In *LICS*, pages 87–96. IEEE Computer Society, 2011. 6

Thomas Ehrhard, Michele Pagani, and Christine Tasson. Probabilistic coherence spaces are fully abstract for probabilistic PCF. In *POPL*, pages 309–320. ACM, 2014. 5

Thomas Ehrhard, Michele Pagani, and Christine Tasson. Full abstraction for probabilistic PCF. *Journal of the ACM*, 65(4):23:1–23:44, 2018a. 5

Thomas Ehrhard, Michele Pagani, and Christine Tasson. Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming. In *POPL*, pages 59:1–59:28. ACM, 2018b. 3, 5, 6

L. Fajstrup, É. Goubault, E. Haucourt, S. Mimram, and M. Raussen. *Directed Algebraic Topology and Concurrency*. Springer International Publishing, 2016. 1, 2

Lisbeth Fajstrup, Martin Raußen, and Eric Goubault. Algebraic topology and concurrency. *Theoretical Computer Science*, 357(1):241–278, 2006. 2

M. Fiore. On the structure of substitution. Invited address for MFPSXXII, 2006. 5

M. Fiore, N. Gambino, M. Hyland, and G. Winskel. The cartesian closed bicategory of generalised species of structures. *J. London Math. Soc.*, 77(1), 2008. 4

M. Fiore, N. Gambino, M. Hyland, and G. Winskel. Relative pseudomonads, Kleisli bicategories, and substitution monoidal structures. *ArXiv e-prints*, 2016. 4, 5

Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985. 2

J.-Y. Girard. Normal functors, power series and lambda-calculus. *Annals of Pure and Applied Logic*, 37 (2), 1988. 3

Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987. 1, 3

Michèle Giry. *A categorical approach to probability theory*, pages 68–85. Springer Berlin Heidelberg, Berlin, Heidelberg, 1982. 4

Noah D. Goodman and Joshua B. Tenenbaum. Probabilistic models of cognition. http://probmods.org, 2014. 4

Éric Goubault, Samuel Mimram, and Christine Tasson. From geometric semantics to asynchronous computability. In *Distributed Computing - 29th International Symposium, DISC 2015, Tokyo, Japan, October 7-9, 2015, Proceedings*, volume 9363 of *Lecture Notes in Computer Science*, pages 436–451. Springer, 2015a. 2, 5

Éric Goubault, Samuel Mimram, and Christine Tasson. Iterated chromatic subdivisions are collapsible. *Applied Categorical Structures*, 23(6):777–818, 2015b. 5

Éric Goubault, Samuel Mimram, and Christine Tasson. Geometric and combinatorial views on asynchronous computability. *Distributed Computing*, 31(4):289–316, 2018. 3, 5

Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *Journal of the ACM (JACM)*, 46(6):858–923, 1999. 2

Maurice Herlihy, Dmitry N. Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2013. 1, 2, 5

Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. A convenient category for higher-order probability theory. In *LICS*, pages 1–12. IEEE Computer Society, 2017. 5

W.A. Howard. The formulae-as-types notion of construction. In Jonathan P. Seldin and J. Roger Hindley, editors, *Essays on Combinatory Logic, Lambda Calculus, and Formalism*, volume to H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, pages 479–490. Academic Press, 1980. 3

M. Hyland. Classical lambda calculus in modern dress. *Math. Struct. Comput. Sci.*, 27(5), 2017. 4, 5

Martin Hyland. Elements of a theory of algebraic theories. *Theor. Comput. Sci.*, 546:132–144, 2014. 4, 5

C. Jones and Gordon D. Plotkin. A probabilistic powerdomain of evaluations. In *LICS*, pages 186–195. IEEE Computer Society, 1989. 4

Achim Jung and Regina Tix. The troublesome probabilistic powerdomain. *Electr. Notes Theor. Comput. Sci.*, 13:70–91, 1998. 4

Klaus Keimel and Gordon D. Plotkin. Mixed powerdomains for probability and nondeterminism. *Logical Methods in Computer Science*, 13(1), 2017. 4

Marie Kerjean and Christine Tasson. Mackey-complete spaces and power series - a topological model of differential linear logic. *Mathematical Structures in Computer Science*, 28(4):472–507, 2018. 3, 5

Dexter Kozen. Semantics for probabilistic programs. *Journal of Computer and System Sciences*, 22, 1981. 4

Dexter Kozen. Kolmogorov extension, martingale convergence, and compositionality of processes. In *LICS*, pages 692–699. ACM, 2016. 5

J. Laird, G. Manzonetto, G. McCusker, and M. Pagani. Weighted relational models of typed lambda-calculi. In *LICS*, 2013. 3

P.-A. Melliès. Categorical semantics of linear logic. 2009. 4

Hammurabi Mendes, Christine Tasson, and Maurice Herlihy. Distributed computability in Byzantine asynchronous systems. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 704–713. ACM, 2014. 5

Eugenio Moggi. Computational lambda-calculus and monads. In *LICS*, pages 14–23. IEEE Computer Society, 1989. 4

M. Pagani, P. Selinger, and B. Valiron. Applying quantitative semantics to higher-order quantum computing. In *ACM SIGPLAN Notices*, volume 49. ACM, 2014. 3

Michele Pagani, Christine Tasson, and Lionel Vaux. Strong normalizability as a finiteness structure via the taylor expansion of λ-terms. In *FoSSaCS*, volume 9634 of *Lecture Notes in Computer Science*, pages 408–423, 2016. 5

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. 5

John Power and Miki Tanaka. Binding signatures for generic contexts. In *TLCA*, volume 3461 of *Lecture Notes in Computer Science*, pages 308–323. Springer, 2005. 5

Adam Scibior, Ohad Kammar, Matthijs Vákár, Sam Staton, Hongseok Yang, Yufei Cai, Klaus Ostermann, Sean K. Moss, Chris Heunen, and Zoubin Ghahramani. Denotational validation of higher-order bayesian inference. *PACMPL*, 2(POPL):60:1–60:29, 2018. 5

D. Scott and C. Strachey. Toward A Mathematical Semantics for Computer Languages. In *Proceedings of the Symposium on Computers and Automata*, volume XXI, pages 19–46, 1971. 3

Steffen Smolka, Praveen Kumar, Nate Foster, Dexter Kozen, and Alexandra Silva. Cantor meets scott: semantic foundations for probabilistic networks. In *POPL*, pages 557–571. ACM, 2017. 4

Christine Tasson and Lionel Vaux. Transport of finiteness structures and applications. *Mathematical Structures in Computer Science*, 28(7):1061–1096, 2018. 5

T. Tsukada, K. Asada, and L. Ong. Generalised species of rigid resource terms. In *LICS*, 2017. 4

Takeshi Tsukada, Kazuyuki Asada, and C.-H. Luke Ong. Species, profunctors and taylor expansion weighted by SMCC: A unified framework for modelling nondeterministic, probabilistic and quantum programs. In *LICS*, pages 889–898. ACM, 2018. 4

Chapter 1

Distributed computing

Publications of the author

Éric Goubault, Samuel Mimram, and Christine Tasson. Geometric and combinatorial views on asynchronous computability. *Distributed Computing*, 31(4):289–316, 2018.

Éric Goubault, Samuel Mimram, and Christine Tasson. Iterated chromatic subdivisions are collapsible. *Applied Categorical Structures*, 23(6):777–818, 2015a.

Éric Goubault, Samuel Mimram, and Christine Tasson. From geometric semantics to asynchronous computability. In *Distributed Computing - 29th International Symposium, DISC 2015, Tokyo, Japan, October 7-9, 2015, Proceedings*, volume 9363 of *Lecture Notes in Computer Science*, pages 436–451. Springer, 2015b.

Hammurabi Mendes, Christine Tasson, and Maurice Herlihy. Distributed computability in Byzantine asynchronous systems. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 704–713. ACM, 2014.

Contents

Introduction

In this Chapter 1, we present works, in collaboration with Éric Goubault and Samuel Mimram, in which we study formal methods for fault-tolerant and asynchronous distributed computing, based on a semantical approach.

A distributed system is made of computing entities, called processes, that communicate through a net. They collaborate by passing messages (or through a shared memory) in order to solve a task. However, they have to handle uncertainty for two reasons. First, in an asynchrounous system, processes execute their algorithm at their own pace. Thus, a process cannot take into account the progress of the others. Second, some processes may be faulty, either by crash failure (they are delayed or stopped) or by Byzantine failure (they are under malicious control). Of course, each process does not have any information on the state of other processes. So that, his decisions have to be based on its partial view of the global system, that he infers from the messages he receives.

A task describes globally the initial and the expected final states of the processes. For instance, one of the very fundamental problem in distributed computing is

The consensus task: Processes start with an initial value, for instance their identifier. To reach a consensus, the final values have first to be identical and second to belong to the set of initial ones.

Solving a task consists in the design of one algorithm that each process will run independently. This algorithm precises the execution trace (that is the computation and communication steps) of the process (that runs it). Now, we consider the system on its whole. Because the system is asynchronous, we have to study every global execution trace, that is the interleaving of the (local) process traces.

Correction of a solution: For any set of initial values, for any interleaved execution trace (whatever asynchronicity and failures can be), the set of final values of correct processes satisfies the specification of the task.

Because of their fast growing number with respect to the number of communication rounds, it is not possible to explore every interleaved execution trace. Thus, we need formal methods to characterize task solvability. Herlihy and Shavit [1999] introduced *protocol complexes*, a geometrical formalization of the set of finite traces. A chromatic complex is made of a set of vertices labeled by process identifiers and a set of simplexes (subsets of vertices) that represent interleaved execution traces (see Definition 1.2.1 and Example 12). Moreover, given a distributed system, a protocol complex C is a subdivision of another one C', if C is obtained from traces with more communication rounds than C'. Then, input and output complexes are used to represent the input and the output values specified by the task. Finally, simplicial maps preserve the relations between traces that are encoded in the protocol complex.

Thanks to this geometrical formalization, Herlihy and Shavit [1999] characterized solvability:

Crash Failure Solvability Theorem: A task is solvable in the asynchronous model with crash failures if and only if there is a simplicial map from a subdivision of the input complex to the output complex.

In Mendes et al. [2014], we generalized this theorem to Byzantine failures. This result holds for colorless tasks in which output values depend only on input values and are independent from process identifiers.

Byzantine Solvability Theorem: A colorless task is solvable in the asynchronous model with $n+1$ processes, at most t Byzantine failures and at most I different input values if and only first, $(n+1) > t(I+2)$ and second, there is a continuous function, compatible with the task specification, from the set of initial configurations with at most t values to the set of final configurations.

The proof that the two conditions imply the existence of a protocol relies on the composition of the given continuous function with two asynchronous Byzantine protocols: k-set agreement (see Chaudhuri [1993]) and barycentric agreement (see Herlihy and Shavit [1999]). On the contrary, if there is a protocol, then we prove the first condition by contradiction and the second one by a reduction to the crash failure case. Indeed, any protocol tolerant to Byzantine failures is also tolerant to crash failures.

Collapsibility Theorem: One important concept, for applying the Crash Failure Solvability Theorem, is the *connexity* of chromatic complexes and their subdivisions. Indeed, if the protocol complex is connected, then the Crash Failure Solvability Theorem implies that the consensus cannot be solved. One way of showing connexity is to prove that the complex is collapsible. That means that we can remove free faces (particular simplices), reducing the protocol complex to a point step by step.

We proved in Goubault et al. [2015b], simultaneously with Kozlov [2013], that the chromatic subdivision complex is collapsible and so connected. We present in Figure 1.1 the collapsing procedure for the chromatic subdivision complex for three processes and one round of communication. In blue are the free faces that are removed at each step.

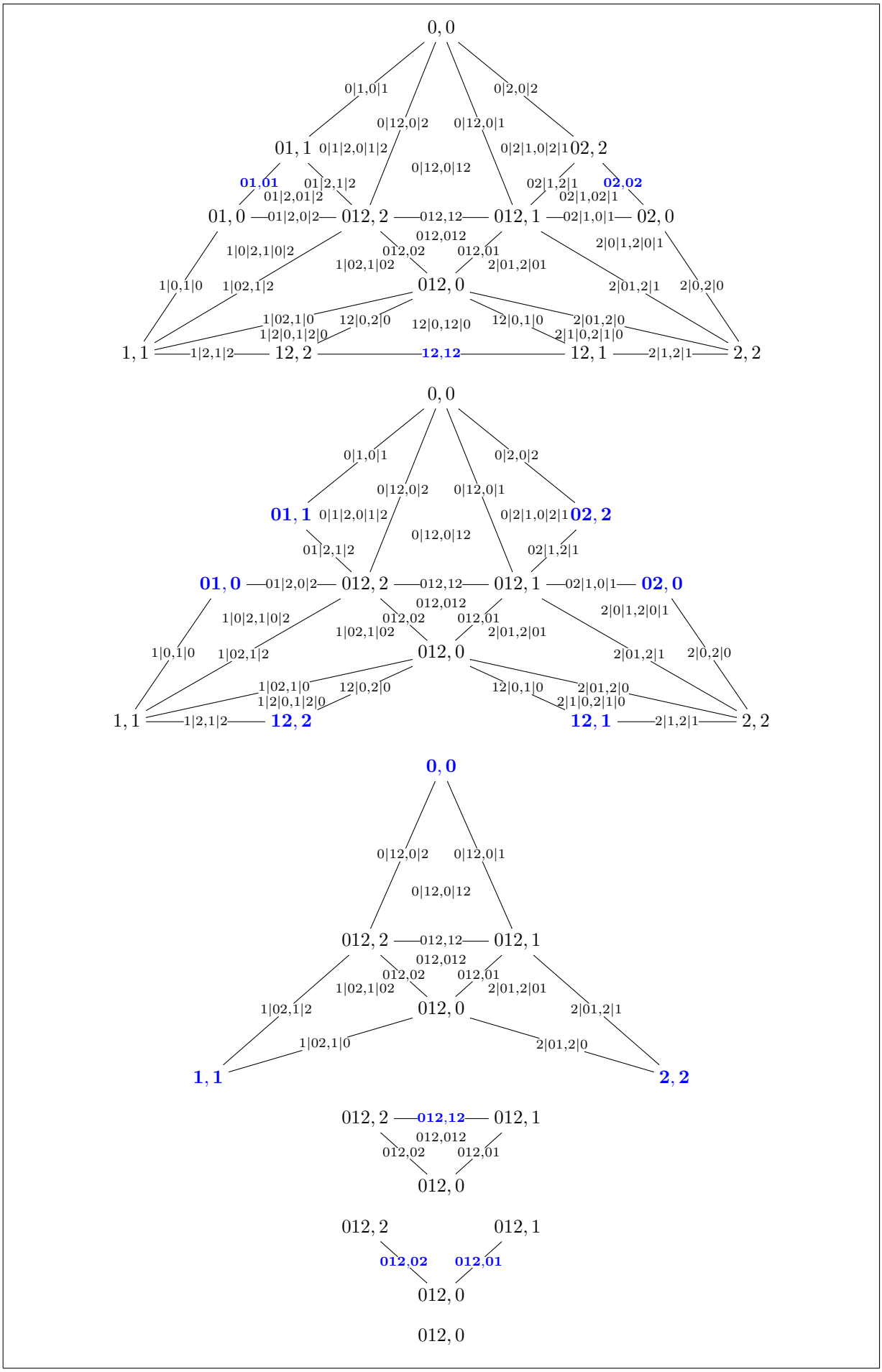


Figure 1.1 – The collapse of the chromatic subdivision complex for three processes.

Although this geometrical approach to problems in fault-tolerant distributed computing has been very successful, there is a potential limitation. Indeed, for some intricate distributed systems, it might be difficult to produce their corresponding protocol complex. To bypass this issue, in Goubault et al. [2015b], we have provided a framework that builds protocol complexes from the *operational semantics* of communication primitives. Although this framework is designed to be general, we present it for a well-known and simple case in fault-tolerant distributed computing: *atomic snapshot protocols* (see Afek et al. [1993]; Anderson [1993]; Lynch [1996]).

This allows to associate to each interleaved execution trace the global final state of the system and to characterize traces that are observationally equivalent (they generate the same final state). We have proved formally the correspondence between traces (up to observational equivalence) and simplexes in the protocol complex.

We have also formalized the correspondence with another successful geometric model of distributed systems where time evolution is taken into account. Introduced in the 1990s, *directed algebraic topology* uses topological models to give a semantics to concurrent and distributed systems.

Similar representations of execution traces and protocol complexes. In this first Chapter 1, we draw a correspondence between

- execution traces (up to observational equivalence),
- simplexes in chromatic protocol complexes,
- directed paths in topological models (up to directed homotopy),
- interval orders (that are partially ordered sets of actions).

Noteworthy, we provided new constructions of the protocols complex, based on traces, dipaths, and interval orders.

Towards a geometric comprehension of probabilistic algorithms. If we allow probabilistic algorithms (processes can toss a coin), then more tasks are solvable. This is the case for consensus, solved by Ben-Or [1983] who designed a randomized algorithm (when at most half of the processes are faulty). Recall that the consensus have been proved to be unsolvable in the deterministic setting (even for 1 faulty process). We would like to give a geometric comprehension to the probabilistic solvability. The idea is that the protocol complexes become almost surely disconnected gradually with the growing number of communication rounds. We will thus need to understand the asymptotic structure of traces that have non zero-probabilities. A first step will be to give a combinatorial recursive description of traces which is easy to work with. And our alternative description of protocol complexes will be our starting point.

Context

The seminal result in this field was established by Fisher, Lynch and Paterson in 1985, who proved that the consensus task that cannot be solved in a message-passing system (or in shared memory Loui and Abu-Amara [1987]) with at most one potential crash Fischer et al. [1985]. Later on, Biran, Moran and Zaks developed a characterization of the decision tasks that can be solved by a (simple) message-passing system in the presence of one failure Biran et al. [1988]. The argument uses a *similarity chain*, which can be seen as a connectedness result of a representation of the space of all reachable states, called the *view complex* Kozlov [2012] or the *protocol complex* Herlihy and Shavit [1999]. Of course, this argument turned out to be difficult to extend to models with more failures, as higher-connectedness properties of the protocol complex matter in these cases. This technical difficulty was first tackled, using homological considerations, by Herlihy and Shavit [1993] (and independently Borowsky and Gafni [1993]; Saks and Zaharoglou [1993]): there are simple decision tasks, such as k-set agreement, a weaker form of consensus, that cannot be solved for $k < n$ in the wait-free asynchronous model, i.e. shared-memory distributed protocols on n processes, with up to $n - 1$ crash failures. Then, the full characterization of wait-free asynchronous decision tasks with atomic reads and writes (or equivalently, with atomic snapshots) was described by Herlihy and Shavit [1999]: this relies on the central notion of chromatic (or colored) simplicial complexes, and their subdivisions. All these results stem from the contractibility of the "standard" chromatic subdivision, which was completely formalized in Kozlov [2012, 2013] (and even for *iterated* models Goubault et al. [2015b]) and which corresponds to the *protocol complex* of distributed algorithms solving layered immediate snapshot protocols. Over the years, the geometric

approach to problems in fault-tolerant distributed computing has been very successful (see Herlihy et al. [2014] for a fairly complete up-to-date treatment).

Another successful theory of concurrent and distributed computations is based on *directed algebraic topology*. Actually, the semantics of concurrent and distributed systems can be given by topological models, as pushed forward in a series of seminal papers in concurrency, in the early 1990s. These papers have first explored the use of precubical sets and *Higher-Dimensional Automata* (which are labeled precubical sets equipped with a distinguished beginning vertex) Pratt [1991]; van Glabbeek [1991]. Then, they have begun to suggest possible homology theories in Goubault and Jensen [1992] and Goubault [1995]. Finally, they have pushed the development of a specific homotopy theory which is a part of a general *directed algebraic topology* Grandis [2009]. On the practical side, directed topological models have found applications to deadlock and unreachable state detection Fajstrup et al. [1998], validation and static analysis Bonichon et al. [2011]; Fajstrup et al. [2012]; Goubault and Haucourt [2005], state-space reduction (as in e.g. model-checking) Goubault et al. [2013], serializability and correctness of databases Gunawardena [1994] (see also Fajstrup et al. [2006]; Goubault [2003] for a panorama of applications).

Contributions

In Goubault et al. [2018] and Goubault et al. [2015a], we show that the protocol complex can be directly derived from an operational semantics of the underlying communication primitives. We furthermore described a correspondance between protocol complexes and directed algebraic topology. This correspondance is the subject of this Chapter 1.

In Goubault et al. [2015b], we have given a purely combinatorial proof that the standard and the iterated chromatic subdivision complex are collapsible. This property is central in theoretical distributed computing as it implies impossibility results in fault-tolerant distributed computing.

In Mendes et al. [2014], we have extended the application of the combinatorial model used to characterize solvability in crash failure systems to colorless tasks in asynchronous Byzantine systems.

Organization of the Chapter

This chapter is devoted to the geometric description of execution traces. It is extracted from Goubault et al. [2015a] which is an extended abstract of Goubault et al. [2018].

Section 1.1 begins in §1.1.1 with the definition of the *standard operational semantics* (or interleaving semantics) of atomic read/write protocols, and more precisely of atomic snapshot protocols where read and write primitives are replaced by update and (global) scan ones. In §1.1.2, we give an alternative *geometric* semantics, which encodes also independence of actions, as a form of *homotopy* in a geometric model. The very basics of *directed algebraic topology* have been introduced for this purpose, but we refer the reader to Fajstrup et al. [2016]; Grandis [2009] for more details. Yet, we prove the fact that (directed) homotopy encodes commutation of actions, in the form of an equivalence between the standard semantics and the geometric semantics. It is shown in Section 1.1.3, Proposition 4 that two traces in the interleaving semantics modulo commutation of actions induce dihomotopic (directed) paths in the geometric model. The converse is shown in Section 1.1.3, Proposition 9, using the combinatorial notion of *interval order* (see Fishburn [1970]). We then combine these results with the semantic equivalence of Proposition 6.

In Section 1.2, we turn to the other geometric model of distributed systems: protocol complexes. The second main contribution of the paper is developed in Section 1.2.2: the protocol complex for atomic snapshot protocols (possibly iterated) is derived from the geometric semantics of Section 1.1.2, through interval orders. We specify this construction in Section 1.2.3 to the case of *layered immediate snapshot* which is generally studied by most authors, since it is much simpler to study, and is enough to prove the classical impossibility theorems, as for instance Herlihy and Shavit [1993]. Our explicit description of the protocol complex in the latter case is the same as the one of Goubault et al. [2015b] (linked as well to the equivalent presentation of Kozlov [2012]). Combined with the result of Goubault et al. [2015b] it proves that the layered immediate snapshot protocols produce collapsible protocol complexes, for any number of rounds. It then implies the asynchronous computability theorem of Herlihy and Shavit [1993] all the way from the semantics of the communication primitives.

1.1 Concurrent semantics of asynchronous read/write protocols

1.1.1 Interleaving semantics of atomic read/write protocols

In *atomic snapshot* protocols, n processes communicate through shared memory using two primitives: `update` and `scan`. Informally, the shared memory is partitioned in n parts, each one corresponding to one of the n processes. The part of the memory associated with process P_i, with $i \in \{0, \dots, n-1\}$, is the one on which process P_i can write, by calling `update`. This primitive writes onto that part of memory, a value computed from the value stored in a local register of P_i. Note that as the memory is partitioned, there are never any write conflicts on memory. Conversely, all processes can read the entire memory through the `scan` primitive. Note also that there are never any read conflicts on memory. Still, it is well known that atomic snapshot protocols are equivalent Lynch [1996] with respect to their expressiveness in terms of fault-tolerant decision tasks they can solve, to the protocols based on atomic registers with atomic reads and writes. Generic snapshot protocols are such that all processes loop, any number of times, on the three successive actions: locally compute a *decision value*, `update` then `scan`. It is also known Herlihy and Shavit [1993, 1999] that, as far as fault-tolerant properties are concerned, an equivalent model of computation can be considered: the full-information protocol where, for each process, decisions are only taken at the end of the protocol, i.e. after rounds of `update` then `scan`, only remembering the history of communications.

Interleaving semantics and trace equivalence.

Formally, we consider a fixed set $\mathcal{V}$ of *values*, together with two distinguished subsets $\mathcal{I}$ and $\mathcal{O}$ of *input* and *output values*, the elements of $\mathcal{V} \setminus (\mathcal{I} \cup \mathcal{O})$ being called *intermediate values*, and an element $\perp \in \mathcal{I} \cap \mathcal{O}$ standing for an *unknown value*. We suppose that the sets of values and intermediate values are infinite countable, so that pairs $\langle x, y \rangle$ of values $x, y \in \mathcal{V}$ can be encoded as intermediate values, and similarly for tuples. We suppose fixed a number $n \in \mathbb{N}$ of processes. We also write $[n]$ as a shortcut for the set $\{0, \dots, n-1\}$, and $\mathcal{V}^n$ for the set of n-tuples of elements of $\mathcal{V}$, whose elements are called *memories*. Given $v \in \mathcal{V}^n$ and $i \in [n]$, we write v_i for the i-th component of v. We write $\perp^n$ for the memory l such that $l_i = \perp$ for any $u \in [n]$.

There are two families of memories, each one containing one memory cell for each process P_i: the *local memories* $l = (l_i)_{i \in [n]} \in \mathcal{V}^n$, and the *global (shared) memory*: $m = (m_i)_{i \in [n]} \in \mathcal{V}^n$. A *state* of a program is a pair $(l, m) \in \mathcal{V}^n \times \mathcal{V}^n$ of such memories. Processes can communicate by performing actions which consist in updating and scanning the global memory, using their local memory: we denote by u_i any `update` by the i-th process and s_i any of its `scan`. We write $\mathcal{A}_i = \{u_i, s_i\}$ and $\mathcal{A} = \bigcup_{i \in [n]} \mathcal{A}_i$ for the set of *actions*.

Formally, the effect of the actions on the state is defined by a *protocol* π which consists of two families of functions $\pi_{u_i} : \mathcal{V} \to \mathcal{V}$ and $\pi_{s_i} : \mathcal{V} \times \mathcal{V}^n \to \mathcal{V}$ indexed by $i \in [n]$ such that $\pi_{s_i}(x, m) = x$ for $x \in \mathcal{O}$. Starting from a state (l, m), the effect of actions is as follows: u_i means "replace the contents of m_i by $\pi_{u_i}(l_i)$", and s_i means "replace the contents of l_i by $\pi_{s_i}(l_i, m)$".

A protocol is *full-information* when $\pi_{u_i}(x) = x$ for every $x \in \mathcal{V}$, i.e. each process fully discloses its local state in the global memory. A sequence of actions $T \in \mathcal{A}^*$ is called an *interleaving trace*, and we write $[\![T]\!]_\pi(l, m)$ for the state reached by the protocol π after executing the actions in T, starting from the state (l, m). A sequence of actions $T \in \mathcal{A}^*$ is *well-bracketed* or *well-formed* (giving some form of generic protocol) when for every $i \in [n]$ we have $\mathrm{proj}_i(T) \in (u_i s_i)^*$, where $\mathrm{proj}_i : \mathcal{A}^* \to \mathcal{A}_i^*$ is the obvious projection which only keeps the letters in $\mathcal{A}_i$ in a word over $\mathcal{A}$. We denote by $\mathcal{A}^\omega$ the set of countably infinite sequences of actions; such a sequence is well-bracketed when every finite prefix is.

It can be noticed that different interleaving traces may induce the same final local view for any process. Indeed, if $i \neq j$, then u_i and u_j modify different parts of the global memory, as we already noted informally, and thus $u_i u_j$ and $u_j u_i$ induce the same action on a given state. Similarly, s_i and s_j change different parts of the local memory, and thus $s_i s_j$ and $s_j s_i$ induce the same action on a given state. On the contrary, $u_i s_j$ and $s_j u_i$ may induce different traces as u_i may modify the global memory that is scanned by s_j. We thus define an *equivalence* $\approx$ on interleaving traces, as the smallest congruence such that $u_j u_i \approx u_i u_j$ and $s_j s_i \approx s_i s_j$ for every indices i and j. Therefore :

Proposition 1. *The equivalence $\approx$ of traces induces an operational equivalence: two equivalent interleaving traces starting from the same initial state lead to the same final state.*

This justifies that we consider traces *up to equivalence* in the following. We use the usual notions on such operational semantics: *execution traces*, *interleaving traces* will denote any finite sequences of actions u_i and s_i in $\mathcal{A}^*$, *maximal* execution traces are traces that cannot be further extended. We also use the classical notions of *length* and *concatenation* of execution traces.

Decision tasks.

We are going to consider the possibility of solving a particular task with an asynchronous protocol. Formally, those tasks are specified as follows:

Definition 2. A *wait-free task specification* Θ is a relation $\Theta \subseteq \mathcal{I}^n \times \mathcal{O}^n$ such that for all $(l, l') \in \Theta$, $i \in [n]$ such that $l_i = \bot$, and $x \in \mathcal{I}$, we also have $(l_i^x, l') \in \Theta$ where l_i^x is the memory obtained from l by replacing the i-th value by x. We note $\operatorname{dom}\Theta = \{l \in \mathcal{I}^n \mid \exists l' \in \mathcal{O}^n, (l, l') \in \Theta\}$ for the *domain* of a wait-free task specification Θ and $\operatorname{codom}\Theta = \{l' \in \mathcal{O}^n \mid \exists l \in \mathcal{I}^n, (l, l') \in \Theta\}$ for its *codomain*.

Notice that $\operatorname{dom}\Theta$ induces a simplicial complex, with $[n] \times (\mathcal{I} \setminus \{\bot\})$ as vertices, and simplices are of the form $\{(i, x) \in [n] \times \mathcal{V} \mid l_i = x \neq \bot\}$, for any $l \in \operatorname{dom}\Theta$. This simplicial complex is called the *input complex*; the *output complex* is defined similarly from $\operatorname{codom}\Theta$. We say that a protocol π *solves* a task specification Θ when for every $l \in \operatorname{dom}\Theta$, and well-bracketed infinite sequence of actions $T \in \mathcal{A}^\omega$, there exists a finite prefix T' of T such that $(l, l') \in \Theta$ where l' is the local memory after executing T', i.e. $(l', m') = [\![T']\!]_\pi(l, \bot^n)$. It can be shown Herlihy and Shavit [1999] that, w.r.t. task solvability, we can assume that $\operatorname{dom}\Theta$ contains only the memory l such that $l_i = i$, for all i, and its faces; for simplicity we will do so in Section 1.2.

Of particular interest is the *view protocol* (sometimes identified with the full-information protocol in the literature) $\pi^\lhd$ such that $\pi^\lhd_{u_i}(x) = x$ for $x \in \mathcal{V}$, i.e. the protocol is full-information, and $\pi^\lhd_{s_i}(x, m) = \langle x, \langle m \rangle\rangle$ for $x \in \mathcal{V}$ and $m \in \mathcal{V}^n$: when reading the global memory, the protocol stores (an encoding of) the pair constituted of its current local memory x and (an encoding as a value of) the global memory m it has read. This is akin to the use of *generic protocols in normal form* Herlihy and Shavit [1999], where protocols only exchange their full history of communication for a fixed given number of rounds, and then apply a local decision function. It can be shown that the view protocol is the "most general one" (i.e. initial in a suitable category). Thus, we will be satisfied with describing the potential sets of histories of communication between processes, without having to encode the decision values: this is the basis of the geometric semantics of Section 1.1.2. As a direct consequence, we recover the usual definition of the solvability of a task as a simplicial map from some iterated protocol complex to the output complex Herlihy and Shavit [1999]; Herlihy et al. [2014].

1.1.2 Directed geometric semantics

In this section, we give an alternative semantics to atomic snapshot protocols, using a geometric encoding of the state space, together with a notion of "time direction". One of the most simple settings in which this can be performed is the one of pospaces Gierz [1980]; Nachbin [1965]: a *pospace* is a topological space $\mathbb{X}$ endowed with a partial order $\leq$ such that the graph of the partial order is closed in $\mathbb{X} \times \mathbb{X}$ with the product topology. The intuition is that, given two points $x, y \in \mathbb{X}$ such that $x \leq y$, y cannot be reached before x. The encoding, or semantics of a concurrent or distributed protocol in terms of directed topological spaces of some sort can be done in a more general manner Fajstrup et al. [2016, 2012]. Here, we simply define, directly, the pospace that gives the semantics we are looking for. It is rather intuitive and we will check this is correct with respect to the interleaving semantics, in Section 1.1.3.

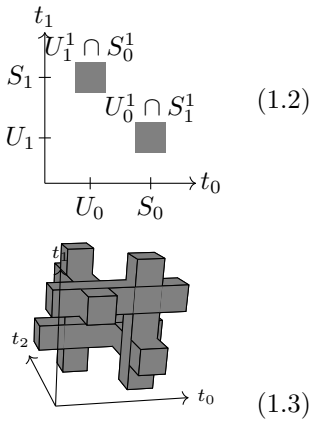
Consider the pospace $\mathbb{X}^n_{(r)}$ below, indexed by the number n of processes and the vector of number of rounds $(r) = (r_0, \ldots, r_{n-1})$ (each $r_i \in \mathbb{N}$, with $i \in [n]$, is the number of times process P_i performs `update` followed by `scan`). Here, we use a vector to represent the number of rounds: this is because we do not want to treat only the layered immediate snapshot protocols, but more general atomic snapshot protocols. We claim now that the geometric semantics of the generic protocol, for n processes and (r) rounds, is represented by the pospace

$$\mathbb{X}^n_{(r)} \quad = \quad \prod_{i \in [n]} [0, r_i] \setminus \bigcup_{i,j \in [n], k \in [r_i],\ l \in [r_j]} U_i^k \cap S_j^l \tag{1.1}$$

endowed with the product topology and product order induced by $\mathbb{R}^n$, where

- $n, r_i \in \mathbb{N}$ and u, s are any reals such that $0 < u < s < 1$: u (resp. v) is representing the local time at which an update (resp. scan) takes place in a round, and their precise values will not matter,
- $U_i^k = \left\{ x \in \prod_{i \in [n]} [0, r_i] \;\middle|\; x_i = k + u \right\}$ stands for the region where the i-th process updates the global memory with its local memory for the k-th time,
- $S_j^l = \left\{ x \in \prod_{i \in [n]} [0, r_i] \;\middle|\; x_j = l + s \right\}$ stands for the region where the j-th process scans the global memory into its local memory for the l-th time.

The meaning of (1.1) is that a state $(x_0, \dots, x_{n-1}) \in \prod_{i\in[n]}[0, r_i]$, i.e. a state in which each process P_i is at local time x_i, is allowed except when it is in $U_i^k \cap S_j^l$ (for $i, j \in [n]$ and $k \in [r_i]$, $l \in [r_j]$): these forbidden states are precisely the states for which there is a `scan` and `update` conflict. Namely, states in $U_i^k \cap S_j^l$ are states for which process P_i updates (for the k-th time) while process P_j scans (for the l-th time), which is forbidden in the semantics. Indeed, the memory has to serialize the accesses since shared locations are concurrently read and written, and either the `scan` operation will come before the `update` one, or the contrary, but the two operations cannot occur at the same time. This is reflected in the geometric semantics by a hole in the state space, as pictured on (1.2) for two processes with one round each, and in (1.4) for two processes with several rounds each. Notice that the holes are depicted as squares instead of points to improve the visibility on the diagram. In higher-dimensions, the holes exhibit a complicated combinatorics.



(1.2)

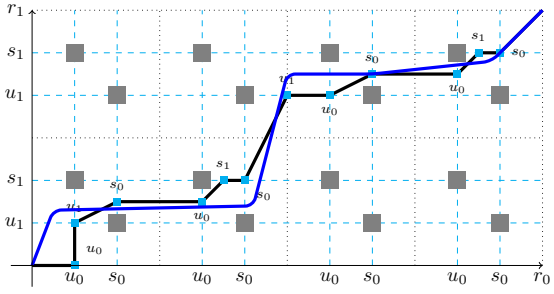
(1.3)

For instance, for three processes, and one round each, as in (1.3) shows forbidden regions that intersect one another. What happens in dimension 3 is that for all 3 pairs of processes (P,Q), we have to produce a forbidden region which has a projection, on the two axes corresponding to P and Q, similar to the one on (1.2). Hence for all three pairs of processes, we have two cylinders with square section punching entirely the set of global states of the system. Each of these 6 cylinders correspond to a pair (P,Q) of processes, and a hole created either by a scan of P and an update of Q, or a scan of Q and an update of P. Consider the cylinder created by the conflict between the scan of P with the update of Q: it intersects exactly two cylinders (parallel to the other axes), the one created by the scan of the third processor R and the update of Q, and the one created by the update of R and the scan of P, see (1.3).

1.1.3 Equivalence of the standard and geometric semantics

In the geometric semantics of Section 1.1.2, we can define notions analogous to equivalence of traces as for the standard interleaving semantics of Section 1.1.1 (Proposition 1). A *dipath* (or *directed path*) in a pospace $(\mathbb{X}, \leq)$ is a continuous map $\alpha : [0, 1] \to \mathbb{X}$ which is continuous and non decreasing when $[0, 1]$ is endowed with the order and topology induced by the real line. A dipath is the continuous counterpart (as we will make clear later) of a trace in the interleaving semantics, or an execution. A dipath $\alpha : [0, 1] \to \mathbb{X}$ is called *inextendible*, if there is no dipath $\beta : [0, 1] \to \mathbb{X}$ such that $\alpha([0, 1]) \subsetneq \beta([0, 1])$. This is the analogous, in our geometric setting, to maximal execution traces. The *concatenation* of two dipaths $\alpha, \alpha' : [0, 1] \to \mathbb{X}$ with compatible ends, i.e. $\alpha(1) = \alpha'(0)$ is the dipath $\alpha \cdot \alpha'$ such that $\alpha \cdot \alpha'(x)$ is $\alpha(x)$ (resp. $\alpha'(2x - 1)$) when $x \leqslant 0.5$ (resp. $x \geq 0.5$).

The continuous setting allows us to use the classical concepts of (di)homotopy, which is the natural notion of equivalence between paths, and to use some tools from algebraic topology to derive properties of protocols (and more generally programs Goubault [2003]). A *dihomotopy* is a continuous map $H : [0, 1] \times [0, 1] \to \mathbb{X}$ such that for all $t \in [0, 1]$, the map $H(-, t)$ is a dipath. Two dipaths α, β such that $\alpha(0) = \beta(0)$ and $\alpha(1) = \beta(1)$ are *dihomotopic*, if there is a dihomotopy $H : [0, 1] \times [0, 1] \to \mathbb{X}$ with $H(-, 0) = \alpha$ and $H(-, 1) = \beta$. We denote by $[\alpha]$ the set of inextendible dipaths dihomotopic to α and $\mathbf{dPath}(\mathbb{X})$ the set of dipaths up to dihomotopy. For instance, two dipaths that are dihomotopic in the geometric semantics $X^2_{(4,2)}$ can be pictured as in Figure (1.4).



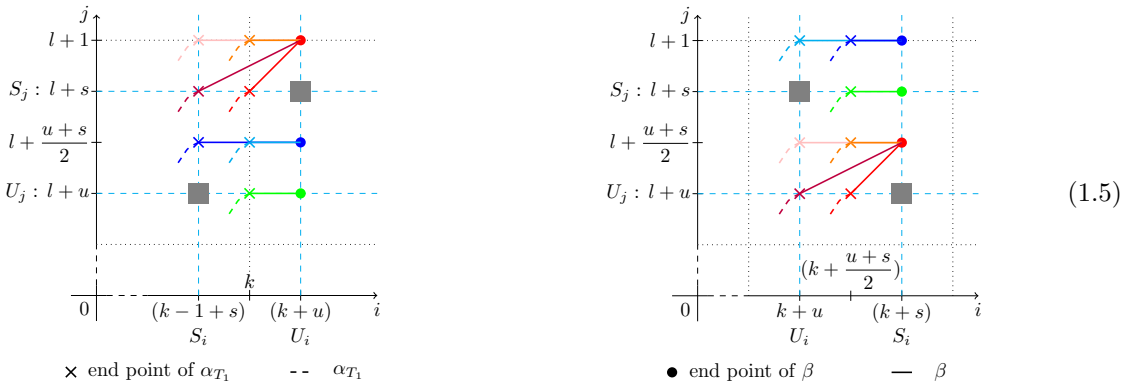
(1.4)

From equivalence classes of interleaving traces to dipaths modulo dihomotopy.

To any interleaving trace T with n processes and (r) rounds, we associate a dipath α_T in $\mathbb{X}^n_{(r)}$. This dipath accurately reflects the whole computation of T, e.g. if T' extends T, then $\alpha_{T'}$ also extends α_T. For example, the black path of (1.4) is the dipath associated to the trace $u_0u_1s_0u_0s_1s_0u_1u_0s_0u_0s_1s_0$: the points along it correspond to actions and the path consists of a linear interpolation between those. The dipath α_T is built by induction on the length of trace T: when T is of length 0, α_T is the constant dipath staying at the origin; when T is the concatenation of a trace T_1 with an action A, we concatenate the dipath α_{T_1} and a dipath β which is defined according to the previous actions in T_1:

Lemma 3. *There exists a (not necessarily inextendible) dipath α_T in $\mathbb{X}^n_{(r)}$ such that $\alpha_T(0)_i = 0$, for every $i \in [n]$, and satisfying the following. For any $i \in [n]$, if the last action of process i in T is its k-th update, then $\alpha_T(1)_i \in \{k+u, k+\frac{u+s}{2}\}$. If it is its k-th scan, then $\alpha_T(1)_i \in \{k+s, k+1\}$. If the last action in T is the k-th update of process i, then $\alpha_T(1)_i = k+u$. If it is the k-th scan of process i, then $\alpha_T(1)_i = k+s$.*

Proof. First, when T is of length 0, α_T is the constant dipath staying at the origin 0. Otherwise, let $T = T_1 \cdot A$ be the concatenation of a trace T_1 with action A (being either update u_i or scan s_i). By induction, we have a dipath α_{T_1} starting at 0 and ending at $\alpha_{T_1}(1)$, associated to T_1, that satisfies Lemma 3. Now, construct a dipath β, which is a line, as pictured below,



(1.5)

starting at $\beta(0) = \alpha_{T_1}(1)$, and ending at $\beta(1)$ and such that:

- Let us assume that the action A is an update, say the k-th update of process i. As partly represented on the left part of (1.5), by Lemma 3, since the previous action was a scan or nothing, $\alpha_{T_1}(1)_i \in \{0, k-1+s, k\}$ and we set $\beta(1)_i = k+u$.
  For any other process $j \neq i$, if the last action of j is its say l-th scan, then $\alpha_{T_1}(1)_j \in \{l+s, l+1\}$ and we set $\beta(1)_j = l+1$ (in red tones), otherwise we set $\beta(1)_j = \alpha_{T_1}(1)_j$ (in blue tones).

- If A is a scan, say the k-th scan of i then, see the right part of (1.5). Since the action of i before was the k-th update, we have $\alpha_{T_1}(1)_i \in \{k+u, k+\frac{u+s}{2}\}$ and we set $\beta(1)_i = k+s$.
  For any other process j, if the last action of j is its l-th update, then $\alpha_{T_1}(1)_j = \{l+u, l+\frac{u+s}{2}\}$ and set $\beta(1)_j = l+\frac{u+s}{2}$ (in red tones), otherwise we set $\beta(1)_j = \alpha_{T_1}(1)_j$ (in blue tones).

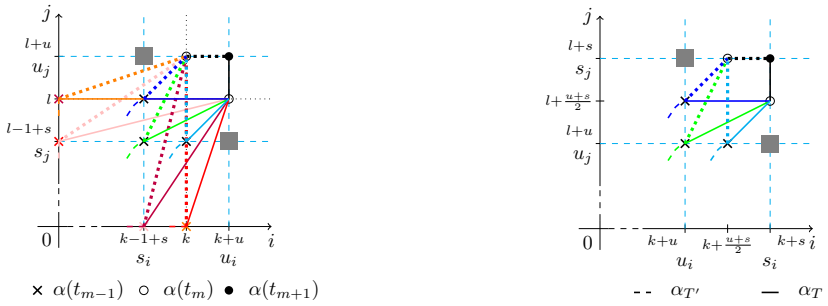
We then define the dipath $\alpha_{T_1 \cdot A} = \alpha_{T_1} \cdot \beta$. □

To a maximal interleaving trace T, we associate an inextendible dipath α'_T by further extending α_T: we define α'_T to be $\alpha_T \cdot \gamma$ where γ is the dipath given by (any parameterization of) the line from $\gamma(0) = \alpha_T(1)$ to $\gamma(1) = (r_i)_{i\in[n]}$, the point $\gamma(1)$ being the end of all inextendible dipaths in $\mathbb{X}^n_{(r)}$. We shall not distinguish in the sequel α'_T from α_T since we will only consider maximal interleaving traces and their inextendible counterparts.

Proposition 4. *Two equivalent interleaving traces induce dihomotopic dipaths.*

Proof. Recall from Proposition 1 that the equivalence of traces is generated by $u_iu_j \approx u_ju_i$ and $s_is_j \approx s_js_i$. Consider two traces T and T' and their associated dipaths α_T and $\alpha_{T'}$. Assume that T and T' are identical until the $(m-1)$-th action and only differ by the ordering of their m-th and $(m+1)$-th actions. Up to reparametrization, we can assume that these actions occur at the same time in α_T and $\alpha_{T'}$,

respectively t_{m-1}, t_m, and t_{m+1}.



(1.6)

First assume that in T, the m-th action is the k-th update of process i and the $(m+1)$-th action is the l-th update of process j. On the left part of (1.6), the possible paths are drawn, one color being associated to one possible point at t_{m-1}. Notice that from t_m, the paths are identical and are colored in black. Indeed, by Lemma 3,

$$\alpha_T(t_m)_i = k+u \quad \text{and} \quad \alpha_T(t_{m+1})_j = l+u.$$

These actions are in the reverse order in T', so

$$\alpha_{T'}(t_m)_j = l+u \quad \text{and} \quad \alpha_{T'}(t_{m+1})_i = k+u.$$

The action of i and j before t_m in T are respectively the $(k-1)$-th scan and the $(l-1)$-th scan or nothing. Hence,

$$\alpha_T(t_{m-1})_i = \alpha_{T'}(t_{m-1})_i \in \{0, (k-1)+s, k\},$$
$$\alpha_T(t_{m-1})_j = \alpha_{T'}(t_{m-1})_j \in \{0, (l-1)+s, l\}.$$

Besides, by construction (the scan and update region is forbidden),

$$\alpha_{T'}(t_m)_i = k \quad \text{and} \quad \alpha_T(t_m)_j = l,$$
$$\alpha_T(t_{m+1})_i = k+u \quad \text{and} \quad \alpha_{T'}(t_{m+1})_j = l+u.$$

Then $t \mapsto t\alpha_T + (1-t)\alpha_{T'}$ is a dihomotopy in $\vec{\mathbb{X}}^n_{(r)}$ between α_T and $\alpha_{T'}$.

Now, assume that in T, the m-th action is the k-th scan of process i and the $(m+1)$-th action is the l-th scan of process j. The possible paths are drawn on the right part of (1.6). Again by Lemma 3,

$$\alpha_T(t_m)_i = k+s \quad \text{and} \quad \alpha_T(t_{m+1})_j = l+s.$$

These action are in the reverse order in T', so

$$\alpha_{T'}(t_m)_j = l+s \quad \text{and} \quad \alpha_{T'}(t_{m+1})_i = k+s.$$

The action of i and j before t_m in T are respectively the k-th update and the l-th update. Hence,

$$\alpha_T(t_{m-1})_i = \alpha_{T'}(t_{m-1})_i \in \{k+u, k+(u+s)/2\},$$
$$\alpha_T(t_{m-1})_j = \alpha_{T'}(t_{m-1})_j \in \{l+u, l+(u+s)/2\}.$$

Besides, by construction (the scan and update region is avoided),

$$\alpha_{T'}(t_m)_i = k+(u+s)/2 \quad \text{and} \quad \alpha_T(t_m)_j = l+(u+s)/2,$$
$$\alpha_T(t_{m+1})_i = k+s \quad \text{and} \quad \alpha_{T'}(t_{m+1})_j = l+s.$$

Then $t \mapsto t\alpha + (1-t)\alpha'$ is a dihomotopy between α_T and $\alpha_{T'}$. □

Equivalence between equivalence classes of interleaving traces and (colored) interval orders.

In order to prove that dipaths modulo dihomotopy are in bijection with interleaving traces modulo equivalence, we introduce a combinatorial tool encoding the history of events observable on both an equivalence class of interleaving traces, and a dihomotopy class of dipaths in our continuous models.

Definition 5. Let $(I_x)_{x\in X}$ be a family of intervals on the real line $(\mathbb{R}, \leqslant)$. This family induces a poset $(X, \preccurlyeq)$, where $\prec$ is defined as $x \prec y$ if and only if for every $s \in I_x$ and $t \in I_y$ we have $s < t$. Such a poset is called an *interval order* Fishburn [1970]. We denote as $x \| y$ the *independence* relation.
An $[n]$-*colored interval order* is given by an interval order $(X, \preccurlyeq)$ and a *labeling* function $\ell : X \to [n]$ such that two elements with the same label are comparable. Then for any $i \in [n]$, the restriction of the interval order to intervals labeled by i is a total order. We denote as $\mathbf{cIO}(X)$ the set of colored interval orders on a set X.

Proposition 6. *There is a bijection between $[n]$-colored interval orders and traces up to equivalence.*

Proof. We first associate a colored interval order to an interleaving trace T. For any $i \in [n]$. Let r_i be the number of occurrences of u_i in T. Let v_i^k and σ_i^k be the respective k-th occurrence of u_i and s_i. Let $X = \{(i,k) \mid k \in [r_i],\ i \in [n]\}$. Any embedding of T in the real line induces an interval order by setting $I_{(i,k)} = [v_i^k, \sigma_i^k]$. More precisely, X is then endowed with the partial order:

$$(i,k) \prec (i',k') \quad \text{iff} \quad \sigma_i^k < v_{i'}^{k'} \tag{1.7}$$

that is σ_i^k occurs before $v_{i'}^{k'}$. We can label this interval order $(X, \preccurlyeq)$ by $\ell : (i,k) \mapsto i$, and hence produce an $[n]$-colored interval order since T is well-bracketed.

Conversely, we associate an interleaving trace T_I to an $[n]$-colored interval order $I = (X, \preccurlyeq)$ labeled by ℓ. For any $i \in [n]$, the set $\{x \in X \mid l(x) = i\}$ is totally ordered of cardinal $[r_i]$. Then, we can assume w.l.o.g. that $X = \{(i,k) \mid k \in [r_i],\ i \in [n]\}$ and that $(i,k) \prec (i,k')$ whenever $k < k'$. Let us choose w.l.o.g. an interval representation I of $(X, \preccurlyeq)$ such that endpoints are pairwise disjoints. For any $k \in [r_i],\ i \in [n]$, let v_i^k and σ_i^k be the left and right endpoint of the interval $I_{(i,k)}$ of the real line. The real line order induces a linear ordering of the endpoints such that the equivalence (1.7) is satisfied. Then T_I is obtained by substituting u_i to v_i^k and s_i to σ_i^k in the given sequence of endpoints.

Let us finally prove that two interval representations $I = (I_{k,i})$ and $J = (J_{k,i})$, indexed by $k \in [r_i]$ and $i \in [n]$, induce equivalent traces $T_I \approx T_J$. From the equivalence (1.7), we deduce that if $(i,k) \prec (i',k')$ then $\sigma_i^k < v_{i'}^{k'}$ and if $(i,k) \| (i',k')$ then $\sigma_i^k \not< v_{i'}^{k'}$, that is $\sigma_i^k > v_{i'}^{k'}$. Thus, the only freedom is on the ordering of the u_i's on the one side, and of the s_i's on the other side, which corresponds precisely to the equivalence of traces. □

From Propositions 4 and 6, we can associate to any interval order a class of dipaths modulo dihomotopy. Let $i : \mathbf{cIO}(X_n) \to \mathbf{dPath}(\mathbb{X}^n_{(r)})$ be mapping an interval order to a dipath up to dihomotopy.

From dipaths modulo dihomotopy to equivalence classes of interleaving traces.

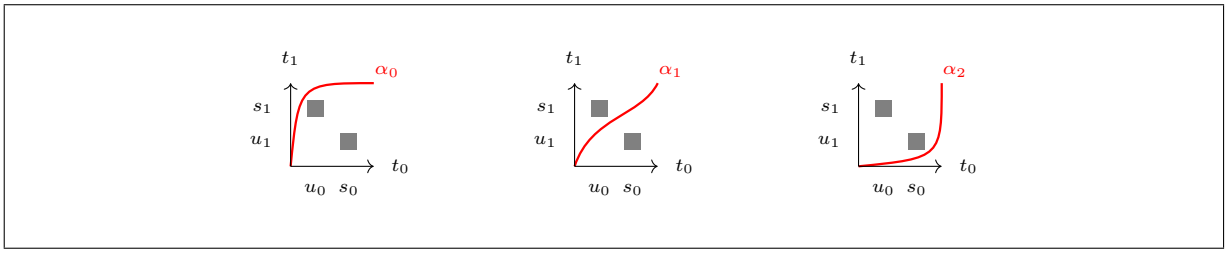
As already mentioned, dipaths geometrically represent execution traces, keeping in mind that dipaths which can be deformed through a continuous family of executions are operationally equivalent. This argument can be made concrete for the asynchronous model we are working on, by giving the explicit relation between dipaths and colored interval orders (Definition 5), because of Proposition 6.

To any inextendible dipath $\alpha : [0,1] \to \mathbb{X}^n_{(r)}$, we associate an interval order $\preccurlyeq_\alpha$ on the set $X^n_{(r)} = \{(i,k) \mid i \in [n], k \in [r_i]\}$ through the interval collection for $i \in [n]$, $I_{(i,k)} = [u_i^k, s_i^k]$ colored by i where u_i^k or s_i^k respectively correspond to the event "α enters an update or scan hyperplane":

$$u_i^k = \inf\left\{t \in [0,1] \;\middle|\; \alpha(t)_i \in U_i^k\right\}, \qquad s_i^k = \inf\left\{t \in [0,1] \;\middle|\; \alpha(t)_i \in S_i^k\right\}. \tag{1.8}$$

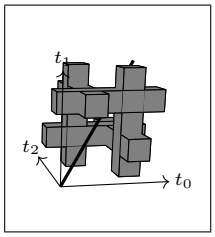
For any $i \in [n]$, the restriction of this order to the intervals labeled by i is a total order. Indeed, dipaths α are non decreasing, $u < s$ and $\alpha(u_i^k)_i = k + u$, $\alpha(s_i^k)_i = k + s$, hence for all $k \in [r_i]$, $u_i^k < s_i^k$ and if $k \neq 0$, $s_i^{k-1} < u_i^k$.

Let us give simple examples of this in dimension 2 and 3. In dimension 2, and for one round, consider the three inextendible dipaths in $\mathbb{X}^2_{(1,1)}$ pictured on the below (we are not writing the round number



as upper index since we are considering here only one round). Those are representatives of the three dihomotopy classes of dipaths in this pospace. The dipath α_0, on the left-most figure, corresponds to an execution in which process 1 does its update and scan before process 0 even starts updating. Hence, the interval of local times at which process 1 updates and scans is less than the interval of local times at which process 0 updates and scans: this is reflected by the corresponding interval order $[u_1, s_1] \prec_{\alpha_0} [u_0, s_0]$. The one on the right-most figure, α_2 is symmetric: the corresponding interval order is $[u_0, s_0] \prec_{\alpha_2} [u_1, s_1]$. The dipath on the middle corresponds to an execution in which the two processes are running synchronously, updating at the same time, and scanning at the same time: the corresponding interval order is $[u_0, s_0] \| [u_1, s_1]$.

In dimension 3, there are more dipaths that one can draw. Consider, for instance, the synchronous execution of the three processes (i.e. the pospace $\mathbb{X}^3_{(1,1,1)}$), shown on the right. It corresponds to the interval order where the intervals $[u_0, s_0]$, $[u_1, s_1]$ and $[u_2, s_2]$ are not comparable. The path figured corresponds to a synchronous execution.



We then have the following simple facts first :

Lemma 7. *Two inextendible dipaths α and β, which intersect the update and scan hyperplanes in the same order, are dihomotopic.*

Proof. Since α and β intersect the update and scan hyperplanes in the same order, we can reparametrize β such that the times at which u_i^k and s_j^l intersect are the same for α and β. Then, the function defined by $H : x, t \mapsto t\,\alpha(x) + (1-t)\beta(x)$ is a dihomotopy. Let us prove that H takes its value in $\mathbb{X}^n_{(r)}$, that is, for all $x, t \in [0,1]$, $H(x,t) \notin U_i^k \cap S_j^l$. Assume for instance that $u_i^k > s_j^l$. If $H(x,t) \in U_i^k$, then $H(x,t)_i = k + u$ and, since $\alpha, \beta \in \mathbb{X}^n_{(r)}$,

- either $\alpha(x)_i > k + u$ and $\beta(x) < k + u$, then, as α and β are non decreasing, $x > u_i^k$ and $x < u_i^k$ and we get a contradiction,
- either $\alpha(x)_i < k + u$ and $\beta(x) > k + u$, this case is impossible for the same reason,
- or $\alpha(x)_i = k + u$ and $\beta(x) = k + u$, then, as α and β are non decreasing,

$$\alpha(x)_j \geq \alpha(u_i^k)_j > \alpha(s_j^l)_j = l + s$$

and $\beta(x)_j > l + s$, thus $H(x,t) \notin S_j^l$.

If $u_i^k < s_j^l$, consider $H(x,t) \in S_j^l$ to show $H(x,t) \notin U_i^k$. □

Remark. One should keep in mind that a dipath α satisfies:

- $\alpha(u_i^k)_i = k + u$ and $\alpha(s_i^k)_i = k + s$,
- if $u_i^k \leq t < s_i^k$, then $k + u \leq \alpha(t)_i < k + s$,
- if $s_i^k \leq t < u_i^{k+1}$, then $k + s \leq \alpha(t)_i < (k+1) + u$.

Moreover, notice that the trace T_α induced by the intersection of α with the update and scan hyperplanes is associated to the interval order $(X^n_{(r)}, \preccurlyeq_\alpha)$.

We write $\alpha \leftrightsquigarrow \beta$ when the two dipaths are dihomotopic.

Proposition 8. *A dipath α is dihomotopic to the dipath associated to the interval order induced by α, that is, $i \circ r(\alpha) \leftrightsquigarrow \alpha$.*

Proof. Let T be a trace representing the interval order $(X^n_{(r)}, \preccurlyeq_\alpha)$ induced by α. Let T_α be the trace induced by the sequence of intersection of α with the update and scan hyperplanes. By Remark 1.1.3, T_α is also representing the interval order $(X^n_{(r)}, \preccurlyeq_\alpha)$, so that T_α and T are equivalent interleaving traces. Thus, $\alpha_T \leftrightsquigarrow \alpha_{T_\alpha}$ by Proposition 4. Now, by construction, the dipath α_{T_α} intersects the update and scan hyperplanes in the order given by T_α, that is in the same order as α (see Remark 1.1.3). Therefore, by Lemma 7, $\alpha \leftrightsquigarrow \alpha_{T_\alpha}$. Finally, we get $\alpha \leftrightsquigarrow \alpha_{T_\alpha} \leftrightsquigarrow \alpha_T = i(r(\alpha))$. □

This implies the following, among the main results of this article :

Proposition 9. *Two dihomotopic inextendible dipaths on $\mathbb{X}^n_{(r)}$ induce the same interval order.*

Theorem 10. *There is a bijective correspondence between traces up to equivalence and dipaths up to dihomotopy over $\mathbb{X}^n_{(r)}$, that is: $r \circ i = \mathrm{id}_{\mathbf{cIO}(X_n)}$, where $r : \mathbf{dPath}(\mathbb{X}^n_{(r)}) \to \mathbf{cIO}(X_n)$ maps a dipath up to dihomotopy to an interval order.*

1.2 Protocol complexes, derived from the concurrent semantics

We will see that two executions modulo dihomotopy correspond to higher-dimensional simplices in protocol complexes (Proposition 13). In the case of update/scan protocols, these executions modulo dihomotopy are characterized by the nice combinatorial notion of interval order, which makes the construction of the protocol complex (Definition 17) from the geometric semantics immediate.

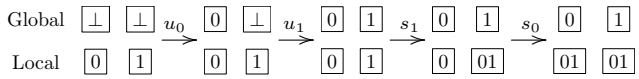
1.2.1 Protocol complex

The protocol complex has been designed Herlihy and Shavit [1999] to represent the possible reachable states, at some given round, of the generic protocol in normal form, i.e. it is going to encode all possible histories of communication between processes, and as we will prove later on, all interleaving traces up to equivalence (or equivalently the dipaths up to dihomotopy), by maximal simplices:

Definition 11. The *protocol complex* for atomic snapshot protocols is the abstract simplicial complex constructed from the generic protocol in normal form, and whose

- vertices are pairs (i, l_i) where $i \in [n]$ represents the name of a process and l_i its local memory,
- maximal simplices are $\{(0, l_0), \dots, (n, l_n)\}$ where l_i is the local view by process i at the end of the execution represented by this simplex.

Example 12. The local views in each vertex are determined by the operational semantics of Section 1.1.1, as in the following example:



leading to the local view $l = \langle\langle 0, \langle 0, 1\rangle\rangle, \langle 1, \langle 0, 1\rangle\rangle\rangle$. Similarly, the trace $u_0 s_0 u_1 s_1$ leads to the local view $l = \langle\langle 0, \langle 0, \bot\rangle\rangle, \langle 1, \langle 0, 1\rangle\rangle\rangle$, and there is a third potential outcome of the computation, symmetric to this last case, in which process 1 updates and scans before process 0 does. Putting this together, according to Definition 11, we get the protocol complex for one round and two processes Herlihy and Shavit [1999]:

$$0, (0\bot) \text{ —— } 1, (01) \text{ —— } 0, (01) \text{ —— } 1, (\bot 1)$$

The encoding of the local states, i.e. vertices in the graph above, is as follows. The identifier of the process whose local view is the number before the comma, e.g. the state $0, (0\bot)$ above is the local view of processor 0. The group of numbers or $\bot$ within parentheses, e.g. $(0\bot)$ in the state above, is a condensed notation for the local state where $l_0 = \langle 0, \langle 0, \bot\rangle\rangle$, see Section 1.1.1. Similarly, state $1, (01)$ denotes the local view of processor 1, with local state such that $l_1 = \langle 1, \langle 0, 1\rangle\rangle$.

1.2.2 Construction of the protocol complex from the directed geometric semantics

We can now link protocol complexes with interval orders, i.e. traces up to equivalence or dipaths up to dihomotopy: a colored interval order represents indeed an execution and we can deduce the local view of the i-th process by restricting the interval order to the last scan of i. We encode local views restricting to the full information generic protocol in normal form with initial local state $l_i = i$ for $i \in [n]$ (this only changes the naming of local states, and not the structure of the protocol complex).

Proposition 13. *Let $(X^n_{(r)}, \preccurlyeq)$ be an $[n]$-colored interval order. Then the local memory of the i-th process at round k of its corresponding execution[1]. is given by its restriction $\mathcal{V}_i^k$ to the k-th scan S_i^k of the i-th process, i.e.*

$$\mathcal{V}_i^k \quad = \quad \{(j,l) \mid (i,k) \| (j,l) \text{ or } (j,l) \prec (i,k)\}$$

meaning that it is the value of the local state l_i under the semantics of Section 1.1.1 for the interleaving path corresponding to the interval order $\mathcal{V}_i^k$ under the equivalence of Proposition 6.

Proof. Remember that $(i,k) \prec (j,l)$ iff S_i^k happens before U_j^l, see (1.7). By contradiction, $(i,k) \| (j,l)$ or $(j,l) \prec (i,k)$ iff S_i^k happens after U_j^l. We conclude, noticing that the i-th local memory only depends on the updates preceding the last scan of process i. □

1. In the full-information generic protocol in normal form, i.e. its view (see Proposition 6 and the following example).

Example 14. Consider again the one round, two processes case. We have represented below the protocol complex already depicted in Example 12, and decorated its maximal simplices, i.e. edges, with the corresponding dipaths modulo dihomotopy above, and the corresponding interval order, below:

$$0,(0\bot) \xrightarrow[0\prec 1]{} 1,(01) \xrightarrow[0\ \ 1]{} 0,(01) \xrightarrow[0\succ 1]{} 1,(\bot 1)$$

The local view of process 0 which is $0,(0\bot)$ comes from the restriction of the interval order ${}_{0\prec 1}$, subscript of the leftmost edge in the graph above, to 0: an interleaving trace corresponding to this interval order, under Proposition 6 is u_0s_0 leading to local state $(0\bot)$ on process 0. Similarly, $1,(01)$ corresponds to the local state $l_1 = (01)$ for process 1, both for the restriction ${}_{0\prec 1}$ of ${}_{0\prec 1}$ to $\mathcal{V}_1^1$ (corresponding to a trace $u_0s_0u_1s_1$, as in the trace $\ulcorner\cdot\cdot\lrcorner$ superscript of the edge on the left of the graph above) and for the restriction ${}_{0\ \ 1}$ of ${}_{0\ \ 1}$ to $\mathcal{V}_1^1$ (corresponding to a trace $u_0u_1s_0s_1$ for instance, as in the trace $\nearrow\!\!\!\cdot\!\!\!\cdot\!\!\!\swarrow$ superscript of the middle edge of the graph above).

We are now in a position to give a combinatorial description of the protocol complex of Definition 11, using interval orders. We call the resulting equivalent complex, the *interval order complex*:

Definition 15. The *interval order complex* is the simplicial complex whose

- vertices are $((i,k),V_i^k)$ where i stands for the i-th process, k for the round number and V_i^k for an interval order such that for all $(j,l) \in V_i^k$, either $(i,k)\|(j,l)$ or $(j,l) \prec (i,k)$,
- maximal simplices are $\{((0,r_0),V_0^{r_0}),\dots,((n,r_n),V_n^{r_n})\}$ such that there is an interval order $(X^n_{(r)},\prec)$ whose restriction to (i,r_i) is $V_i^{r_i}$.

In that case we say that it is the interval order complex on (r) rounds and for $n+1$ processes.

Example 16. An example of interval order complex with the traces corresponding to the execution for 2 processes, 2 rounds is depicted at Figure 1.2. Note that this is not the classical iterated subdivision in three parts at each round, i.e. a 9 edges complex, that is depicted for atomic snapshot protocols Herlihy et al. [2014]. This is because we are considering more executions than the classical *layered immediate snapshot protocols* Herlihy et al. [2014]: we allow round 2 of process 0 to begin while process 1 is still in round 1 for instance. Consider the interval order $\begin{smallmatrix}0\succ 1\\ \uparrow\!\times\!\uparrow\\ 0\succ 1\end{smallmatrix}$ labeling the upper left edge of the protocol complex in Figure 1.2, where an arrow $x \succ y$ means $x \prec y$. As shown in the same figure, it corresponds to the execution $\ulcorner\cdot\cdot\cdot\cdot\lrcorner$ precisely where process 0 is executing its 2 rounds before process 1 even starts its first round. The local view of process 0 at (its) round 2 corresponds to the interval order $\begin{smallmatrix}0\\ \uparrow\\ 0\end{smallmatrix}$, restriction of $\begin{smallmatrix}0\succ 1\\ \uparrow\!\times\!\uparrow\\ 0\succ 1\end{smallmatrix}$ to $\mathcal{V}_0^{(2,0)}$. An interleaving trace corresponding to this is e.g. $u_0s_0u_0s_0$, which, by the semantics of Section 1.1.1, leads to the local state of process 0: $\langle 0,\langle 0,\langle 0,\bot\rangle\rangle\bot\rangle$ written in condensed form as the upper left local state $0,((0\_)\_)$ in Figure 1.2.

In Figure 1.3, we show the interval order complex for 3 processes and 1 round. Note again that we do not have exactly the same picture as in Herlihy et al. [2014]: to the 13 triangles of Herlihy et al. [2014], we have to add the 6 extra blue triangles that make the complex not faithfully representable as a planar shape and which correspond to non immediate snapshot executions. For instance, the upper left blue triangle is labeled with the interval order where 0 is not comparable to both 1 and 2, and 2 is less than 1. An interleaving trace (up to equivalence) corresponding to this interval order is given on the same figure: $u_0u_2s_2u_1s_1s_0$.

1.2.3 Particular case of 1-round immediate snapshot protocols

We have not quite finished with describing the connections between directed algebraic topology and the protocol complex approach : the combinatorial description of the protocol complex in the case of layered immediate snapshot protocols seems, at first glance, of a different nature than the one using interval order complexes of Definition 15. We recall that an (layered, for multi-round protocols) *immediate snapshot* protocol Herlihy et al. [2014] is a protocol where the snapshot of a given process comes "right after" its update, meaning that the allowed traces (within one round), up to equivalence, should be, of the form $u_{i_1}\dots u_{i_k}s_{i_1}\dots s_{i_k}$. Of course, there is some difference in that interval order

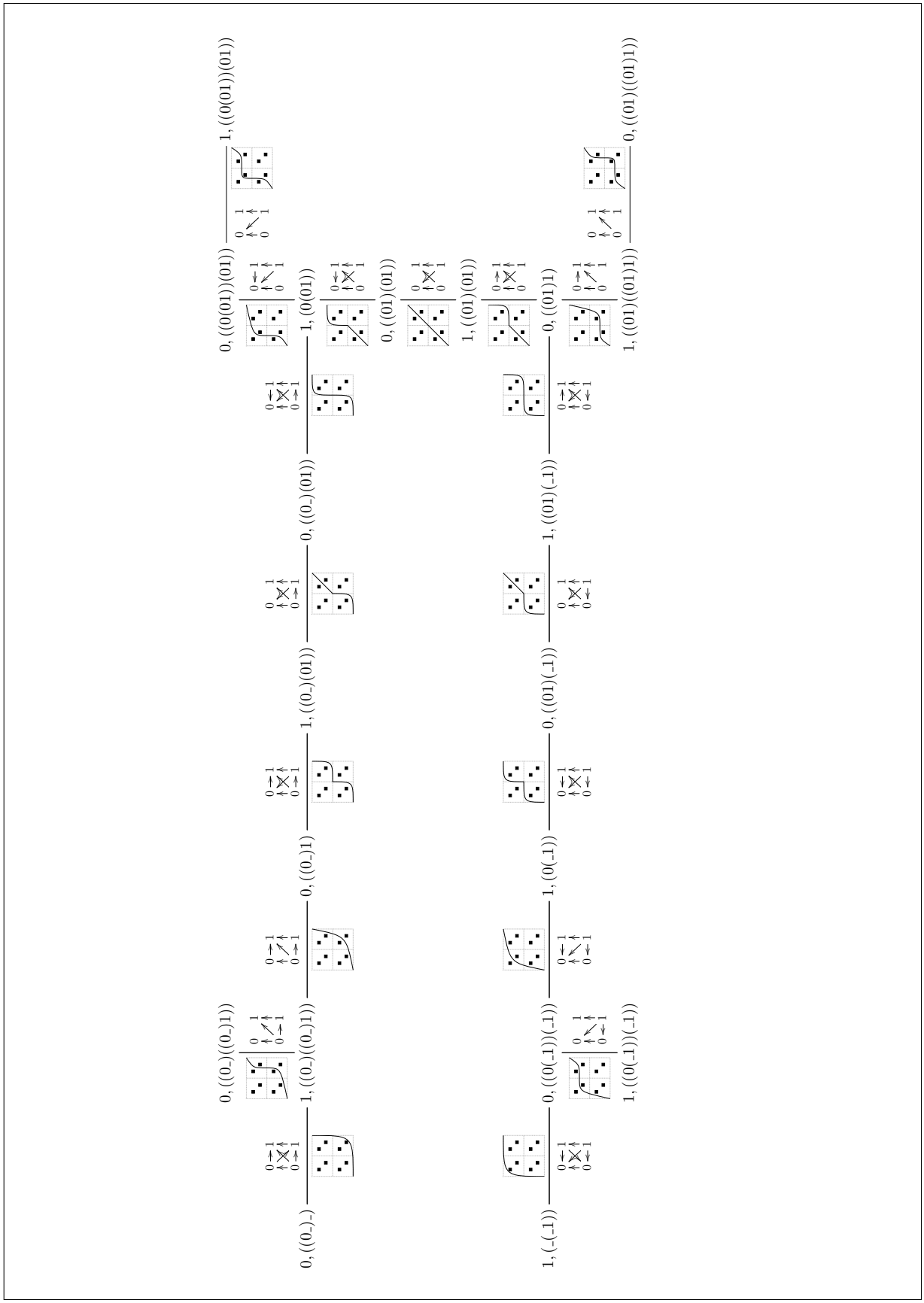


Figure 1.2 – The protocol complex, decorated with corresponding traces and interval orders, of 2 processes, 2 rounds.

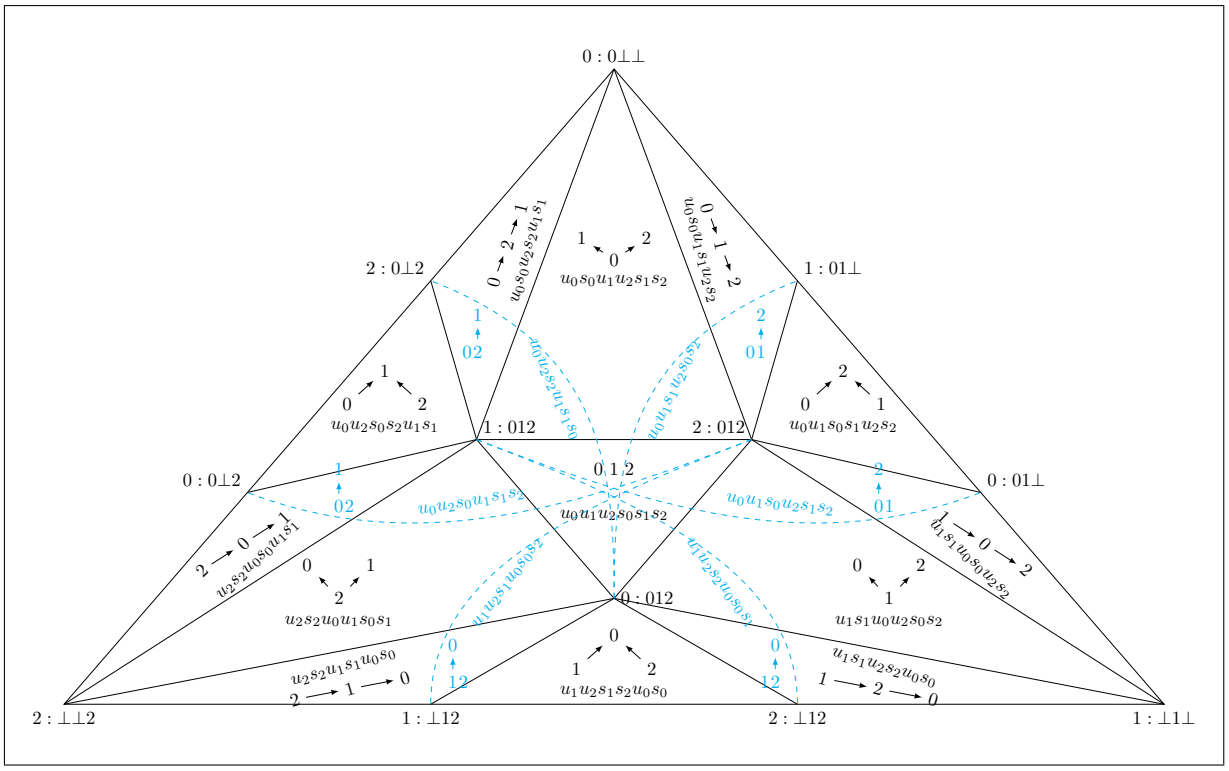


Figure 1.3 – Interval order complex with traces of 3 processes, 1 round.

complexes account for non necessarily layered nor "immediate" protocols. It is the aim of this section to make the connection between the subcomplex of interval order complexes describing layered immediate snapshot protocols, and the equivalent two definitions of chromatic barycentric subdivision Goubault et al. [2015b]; Kozlov [2012] that describe combinatorially the protocol complex in that case.

The standard chromatic subdivision $\chi(\Delta^{[n]})$ of the standard $[n]$-colored simplicial complex $\Delta^{[n]}$ is defined as follows (see Goubault et al. [2015b], where an equivalence with the Definition in Kozlov [2012] is also shown):

Definition 17. The *standard chromatic subdivision* $\chi(\Delta^{[n]})$ of $\Delta^{[n]}$ is the $[n]$-colored simplicial complex whose vertices are pairs (V, i) with $V \subseteq [n]$ and $i \in V$ and simplices are sets of the form $\sigma = \{(V_0, i_0), \ldots, (V_d, i_d)\}$ with $d \geqslant -1$ ($\sigma = \emptyset$ when $d = -1$) which are

1. well-colored: for every $k, l \in [d]$, $i_k = i_l$ implies $k = l$,
2. ordered: for every $k, l \in [d]$, $V_k \subseteq V_l$ or $V_l \subseteq V_k$,
3. transitive: for every $k, l \in [d]$, $i_l \in V_k$ implies $V_l \subseteq V_k$.

This complex is colored via the second projection: $\ell(V, i) = i$.

Remark. The transitivity (property 3) of Definition 17 is equivalent to looking only at immediate snapshot executions. Observe the left upper blue triangle of Figure 1.2, which is composed of vertices $(0 : 012)$, $(1 : 012)$ and $(2 : 0\bot2)$ (respectively meaning $(\{0, 1, 2\}, 0)$, $(\{0, 1, 2\}, 1)$ and $(\{0, 2\}, 2)$ in the notations of Definition 17). It does not correspond to a layered execution: it corresponds to the equivalence class of traces $u_0u_2s_2u_1s_1s_0$. Transitivity does not hold either: $0 \in \{0, 2\}$ but $\{0, 1, 2\} \not\subseteq \{0, 2\}$.

This leads us to the last main result of our article :

Theorem 18. *Layered immediate snapshot executions correspond to the interval orders such that: $J \prec K$ and I is not comparable with J implies $I \prec K$. The subcomplex of the interval order complex on one round, $(X^n_{(1,\ldots,1)}, \preccurlyeq)$, that contains only immediate snapshot executions is isomorphic to the chromatic barycentric subdivision of Definition 17.*

Proof. For the first part, suppose that we have an interval order $\preccurlyeq$, representing some simplex in the interval order complex such that $J \prec K$ and I is not comparable with J and K. I, J and K correspond to some intervals of updates and scans local times on some process, $[u_i^{l_i}, s_i^{l_i}]$, $[u_j^{l_j}, s_j^{l_j}]$ and

$[u_k^{l_k}, s_k^{l_k}]$ respectively. Suppose that I is not comparable with K, this means that the interleaving path $\dots u_i^{l_i} \dots u_j^{l_j} \dots s_j^{l_j} \dots u_k^{l_k} \dots s_k^{l_k} \dots s_i^{l_i} \dots$ is in the equivalence class represented by the interval order we are considering. This is clearly not layered nor immediate snapshot, therefore being a layered immediate snapshot execution implies the condition on $\preccurlyeq$ of Theorem 18.

Conversely, we suppose that for I not comparable to J and $J \prec K$, then $I \prec K$ and we prove that all interleaving paths are layered and immediate snapshot ones. Suppose we have an interleaving path (up to equivalence) of the form: $Tu_j^{l_j}Us_j^{l_j}Vu_k^{l_k}Ws_k^{l_k}X$ where T, U, V, W and X are interleaving paths. This is a layered immediate snapshot execution except if there are update and scans $u_i^{l_i}$, $s_i^{l_i}$ such that $u_i^{l_i}$ appears in U and $s_i^{l_i}$ appears in W. But $u_i^{l_i}$ appearing in U implies $I = [u_i^{l_i}, s_i^{l_i}]$ is not comparable with J and hence, by hypothesis, I must be less that K, implying that $s_i^{l_i}$ appears in U or V.

Now, we prove the second statement. Consider a simplex $\sigma = \{(V_0, i_0), \dots, (V_d, i_d)\}$ with $d \geqslant 0$ (the case $d = -1$ is trivial) in the chromatic barycentric subdivision of Definition 17. We associate to σ the following simplex in the interval order complex: we construct a partial order $\preccurlyeq_\sigma$ on $\{(V_0, i_0), \dots, (V_d, i_d)\}$ such that $V_k \prec_\sigma V_l$ if $V_k \subsetneq V_l$ and the color of (V_l, i_l) is i_l, we just need to prove that this partial order is a colored interval order, and that the condition of Theorem 18 holds.

Let us now consider, in our partial order $\preccurlyeq_\sigma$, four elements (V_x, i_x), (V_y, i_y), (V_z, i_z) and (V_t, i_t), and suppose furthermore that $(V_x, i_x) \prec_\sigma (V_y, i_y)$ and $(V_z, i_z) \prec_\sigma (V_t, i_t)$. Then, as σ is "ordered" (see Definition 17), necessarily, either $V_x \subseteq V_z$ or $V_z \subseteq V_x$. Suppose we are in the first situation. We also have that $V_z \subseteq V_t$ and $V_z \neq V_t$ by definition of $\preccurlyeq$. Hence $V_x \prec_\sigma V_t$. We conclude that, as a partial order, $\preccurlyeq_\sigma$ is (2+2)-free, property which characterizes interval orders Fishburn [1970].

Now consider again σ in the chromatic barycentric subdivision, and its associated interval order $\preccurlyeq_\sigma$. Take $(V_y, i_y) \prec_\sigma (V_z, i_z)$ and (V_x, i_x) which is not comparable with (V_y, i_y). Hence, by definition of the (strict) order $\prec_\sigma$, $V_x = V_y$ or $V_x \not\subseteq V_y$. In the first case, $(V_x, i_x) \prec_\sigma (V_z, i_z)$, trivially, and in the second case, by property 2 ("ordered") of Definition 17, $V_y \subsetneq V_x$ which implies $(V_y, i_y) \prec_\sigma (V_x, i_x)$, impossible since (V_x, i_x) and (V_y, i_y) are supposed incomparable.

Finally, note that well-coloredness of σ implies that the labeling we define is indeed a labeling function of a colored interval order.

Conversely, suppose we have a 1-round colored interval order $(X, \preccurlyeq)$ on $d+1$ elements which satisfies the property from Theorem 18. We consider the interval orders V_i^k, restriction of X to $\mathcal{V}_i^k = \{(j,l) \mid (i,k) \| (j,l) \text{ or } (j,l) \prec (i,k)\}$. We construct a (colored) d-simplex in the chromatic barycentric subdivision of Definition 17 by defining k-simplices (for all $k \leqslant n$) $\sigma_X = ((|V_i^{k_i}|, i))_{i \in [k]}$ (where $|V|$ the set of elements of an interval order V). Indeed we check easily that this is well-colored. Suppose we have $(|V_k|, i_k)$ and $(|V_l|, i_l)$ such that $i_l \in |V_k|$. As V_k and V_l are restrictions of the same interval order to both the set of elements less than or incomparable to i_k, respectively i_l, and that by definition of V_l, $i_l \in V_l$, we have $|V_l| \subseteq |V_k|$. A similar argument shows that property 2 of Definition 17 holds as well. □

1.3 Conclusion

We have revealed strong connections between directed algebraic topology, with its applications to semantics and validation of concurrent systems, and the protocol complex approach to fault-tolerant distributed systems. This has been exemplified on the simple layered immediate snapshot model, but also on the more complicated (non layered, non immediate) iterated snapshot model. This, combined with the results of Kozlov [2013] and Goubault et al. [2015b], entirely classifies geometrically the computability of wait-free layered immediate snapshot protocols, directly from the semantics of the update and scan primitives. We classified combinatorially, en route, the potential schedules of executions (equivalently, the potential local views of processes) as an interesting and well-known combinatorial structure: interval orders.

This is a first step towards a more ambitious program. Fault-tolerant distributed models, whose protocol complex are more complex to guess combinatorially, may be handled by going through the very same steps we went through, starting with the geometric semantics of the communication primitives, and classifying dipaths modulo dihomotopy. We shall apply this to atomic read/write protocols with extra synchronization primitives such as test&set, compare&swap and others. In the long run, we would like to derive impossibility results directly by observing some obstructions in the semantics, in the form of suitable directed algebraic topological invariants.

1.4 Bibliography

Y. Afek, H. Attiya, D. Dolev, E. Gafni, M. Merritt, and N. Shavit. Atomic snapshots of shared memory. *J. ACM*, 40(4), September 1993. 14

J. H. Anderson. Composite registers. In *Conference on Principles of Distributed Computing*. ACM, New York, 1993. 14

Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*, PODC '83, pages 27–30, New York, NY, USA, 1983. ACM. ISBN 0-89791-110-5. 14

Ofer Biran, Shlomo Moran, and Shmuel Zaks. A combinatorial characterization of the distributed tasks which are solvable in the presence of one faulty processor. In *Proceedings of the seventh annual ACM Symposium on Principles of distributed computing*, pages 263–275. ACM, 1988. 14

R. Bonichon, G. Canet, L. Correnson, É. Goubault, E. Haucourt, M. Hirschowitz, S. Labbé, and S. Mimram. Rigorous evidence of freedom from concurrency faults in industrial control software. In *SAFECOMP*, 2011. 15

E. Borowsky and E. Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *Proc. of the 25th STOC*. ACM Press, 1993. 14

Soma Chaudhuri. More choices allow more faults: Set consensus problems in totally asynchronous systems. *Inf. Comput.*, 105(1):132–158, 1993. 12

L. Fajstrup, É. Goubault, and M. Raussen. Detecting deadlocks in concurrent systems. In *CONCUR*, number 1466 in LNCS. Springer-Verlag, 1998. 15

L. Fajstrup, É. Goubault, E. Haucourt, S. Mimram, and M. Raussen. *Directed Algebraic Topology and Concurrency*. Springer International Publishing, 2016. 15, 17

Lisbeth Fajstrup, Martin Raußen, and Eric Goubault. Algebraic topology and concurrency. *Theoretical Computer Science*, 357(1):241–278, 2006. 15

Lisbeth Fajstrup, Eric Goubault, Emmanuel Haucourt, Samuel Mimram, and Martin Raußen. Trace spaces: An efficient new technique for state-space reduction. In *ESOP*, pages 274–294, 2012. 15, 17

Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985. 14

Peter C Fishburn. Intransitive indifference with unequal indifference intervals. *Journal of Mathematical Psychology*, 7(1):144–149, 1970. 15, 21, 27

G. Gierz. *A Compendium of continuous lattices*. Springer, 1980. 17

É. Goubault. *The Geometry of Concurrency*. Ph.D. dissertation, ENS, 1995. 15

É. Goubault. Some geometric perspectives in concurrency theory. *Homology, Homotopy and Appl.*, 2003. 15, 18

É. Goubault and E. Haucourt. A practical application of geometric semantics to static analysis of concurrent programs. In *CONCUR 2005*. Springer, 2005. 15

É. Goubault and T. P. Jensen. Homology of higher-dimensional automata. In *Proc. of CONCUR*, 1992. 15

É. Goubault, T. Heindel, and S. Mimram. A geometric view of partial order reduction. *MFPS, Electr. Notes Theor. Comput. Sci.*, 298, 2013. 15

Éric Goubault, Samuel Mimram, and Christine Tasson. From geometric semantics to asynchronous computability. In *Distributed Computing - 29th International Symposium, DISC 2015, Tokyo, Japan, October 7-9, 2015, Proceedings*, volume 9363 of *Lecture Notes in Computer Science*, pages 436–451. Springer, 2015a. 15

Éric Goubault, Samuel Mimram, and Christine Tasson. Iterated chromatic subdivisions are collapsible. *Applied Categorical Structures*, 23(6):777–818, 2015b. 12, 14, 15, 26, 27

Éric Goubault, Samuel Mimram, and Christine Tasson. Geometric and combinatorial views on asynchronous computability. *Distributed Computing*, 31(4):289–316, 2018. 15

M. Grandis. *Directed Algebraic Topology : Models of Non-Reversible Worlds*, volume 13 of *New Mathematical Monographs*. Cambridge University Press, 2009. ISBN 978-0-521-76036-2. 15

J. Gunawardena. Homotopy and concurrency. *Bulletin of the EATCS*, 54:184–193, 1994. 15

Maurice Herlihy and Nir Shavit. The asynchronous computability theorem for t-resilient tasks. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 111–120. ACM, 1993. 14, 15, 16

Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *Journal of the ACM (JACM)*, 46(6):858–923, 1999. 12, 14, 16, 17, 23

Maurice Herlihy, Dmitry Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Elsevier, 2014. 15, 17, 24

Dmitry Kozlov. Chromatic subdivision of a simplicial complex. *Homology, Homotopy and Applications*, 14(2):197–209, 2012. 14, 15, 26

Dmitry Kozlov. Topology of the view complex. *arXiv preprint arXiv:1311.7283*, 2013. 12, 14, 27

M. C. Loui and H. H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. *Advances in Computing Research*, 4, 1987. 14

Nancy A Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996. 14, 16

Hammurabi Mendes, Christine Tasson, and Maurice Herlihy. Distributed computability in Byzantine asynchronous systems. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 704–713. ACM, 2014. 12, 15

L. Nachbin. *Topology and order*. Van Nostrand mathematical studies. Van Nostrand, 1965. 17

V. Pratt. Modeling concurrency with geometry. In *POPL*. ACM Press, 1991. 15

Michael E. Saks and Fotios Zaharoglou. Wait-free k-set agreement is impossible: the topology of public knowledge. In *STOC*, pages 101–110, 1993. 14

R. van Glabbeek. Bisimulation semantics for higher dimensional automata. Technical report, Stanford, 1991. 15

Chapter 2

Differential Semantics

Publications of the author

Marie Kerjean and Christine Tasson. Mackey-complete spaces and power series - a topological model of differential linear logic. *Mathematical Structures in Computer Science*, 28(4):472–507, 2018.

Christine Tasson and Lionel Vaux. Transport of finiteness structures and applications. *Mathematical Structures in Computer Science*, 28(7):1061–1096, 2018.

Michele Pagani, Christine Tasson, and Lionel Vaux. Strong normalizability as a finiteness structure via the taylor expansion of λ-terms. In *FoSSaCS*, volume 9634 of *Lecture Notes in Computer Science*, pages 408–423, 2016.

Richard Blute, Thomas Ehrhard, and Christine Tasson. A convenient differential category. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 53(3):211–232, 2012.

Contents

Introduction

Denotational semantics addresses one of the most fundamental challenges in computer science: describing how and what programs compute.

Programming language semantics. The well-known Curry-Howard-Lambek correspondence is at the heart of the design of programming languages, ensuring correctness *by construction*. The "Curry-Howard" approach (see Howard [1980]) relates programs (of λ-calculus) and proofs (of natural deduction) via a one-to-one correspondence, the types of programs corresponding to formulas. The third part of the correspondence (due to Lambek) relates proofs and programs with structured categories (cartesian closed categories in the λ-calculus case). In this way, it stresses the importance of compositionality for the properties of interest. Such categorical axiomatizations often result from the study of denotational semantics for programming languages. Indeed, semantics offers concrete descriptions of the program invariants, from which the categorical axiomatics can be extracted.

Power series, syntax and semantics. A fine understanding of the usage of resources is at the core of *quantitative semantics*, which is a flourishing research area with deep results and constructions. They have in particular enabled the construction of models of computational cost (time and space consumption) (Laird et al. [2013]), of higher-order quantum functional programming (Pagani et al. [2014]), and of probabilistic programming with continuous data types (Ehrhard et al. [2018]). In §3.5 of Chapter 3, we present this last model: the first one combining higher-order, recursion and measurability.

In this line of research, power series have emerged as one of the most useful tools to encode resource usage in the λ-calculus. More precisely, programs can be seen as a superposition of monomials representing finite approximations of the computation. The degree of those monomials is related to the number of times an argument is used by the program (see Ehrhard and Regnier [2003]). This approximation theory, called *syntactic Taylor expansion*, was studied in Ehrhard and Regnier [2008] and Ehrhard and Regnier [2006a]. Seen through the Curry-Howard correspondence, the Taylor expansion has a counterpart in intuitionistic logic: proofs can be interpreted as power series and their derivatives provide linear approximations of them. An early version of this observation led Girard to introduce **Linear Logic** (LL) (see Girard [1987]) and later Ehrhard and Regnier to introduce the **differential λ-calculus** (see Ehrhard and Regnier [2003]). Both these systems emerged after a careful study of denotational models based on power series. The first one was introduced by Girard [1988]. It was based on normal functors (power series whose coefficients are sets). Then, Ehrhard [2005] introduced finiteness spaces which is based based on analytic functions over *linearized* topological vector spaces and which is closely related to normalization as proved by Ehrhard [2010] and developed in Pagani et al. [2016]. Further models based on topological vector spaces were revealed in Blute et al. [2012] and equipped with analytic functions in Kerjean and Tasson [2016].

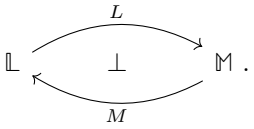
As a matter of fact, quantitative semantics completes the Curry-Howard correspondence by involving linear algebra, computer science and proof theory:

Denotations	$\Leftrightarrow$	Programs	$\Leftrightarrow$	Proofs
power series	$\Leftrightarrow$	duplicating or erasing resources	$\Leftrightarrow$	using structural and logical rules
linear functions	$\Leftrightarrow$	using resources exactly once	$\Leftrightarrow$	using only logical rules

Using methods from linear algebra and analysis in these quantitative approaches has allowed to solve problems which were left open in the traditional semantics based on domains. For instance, in Ehrhard et al. [2014], the regularity of power series has been the key ingredient for proving that the model of probabilistic programming with discrete data types is precise enough to characterize statically the computational behaviour of probabilistic programs. This is the subject of next Chapter 3.

Linear Logic is characterized by an involutive negation and a pair of dual, *exponential* modalities (denoted by ! and ?) which regulate the (explicit) use of structural rules: contraction and weakening. Such a control over contraction and weakening brings an analogue of the notion of resource to proof theory. This allows to consider quantitative properties of proofs. In particular, LL allows the decomposition of the type $A \Rightarrow B$ into more primitive connectives: $A \Rightarrow B = !A \multimap B$. Here, the arrow $A \multimap B$ denotes the space of the linear functions (from a semantic point of view), and the type of the programs using their resources exactly once (from an operational point of view). The type $!A$ denotes the space of the resources of type A that can be used at will; semantically, $!A$ is in some sense analogous to the space of the power series with formal indeterminates over A. For Linear Logic, the third part of the Curry-Howard-Lambek correspondence, namely the categorical axiomatisation, can be formalized

by Linear-Non-Linear categories as presented in Melliès [2009]. In those categorical models, there is a well-behaved adjunction between a symmetric monoidal category $\mathbb{L}$ and a cartesian one $\mathbb{M}$, allowing the exponential modality ! to be interpreted as the comonad $! = L \circ M$:



In order to be able to take quantitative properties into account, it is natural to interpret programs as analytic functions. In particular, these functions are infinitely differentiable. A crucial issue is then to understand whether differentiation is a meaningful syntactic operation. A first positive answer was given in Ehrhard and Regnier [2003] with the *differential* λ*-calculus* and in Ehrhard and Regnier [2006b] with *differential linear logic*. Both of these extensions feature two different ways of passing inputs to programs: the usual, *unrestricted* one, and a new, *linear* one. The latter one consists in defining the derivative $\mathtt{D}f \cdot x$ of the program f, seen as a function, on the argument x. The evaluation of $\mathtt{D}f \cdot x$ has a precise operational meaning: it corresponds to passing the input x to f *exactly once*. This imposes non-deterministic choices: if f contains several subroutines, each of them demanding for a copy of x, then there are different evaluations of $\mathtt{D}f \cdot x$, depending on which subroutine is fed with the unique available copy of x. We thus need a formal sum, where each summand represents a choice. Such sums express syntactically the addition between vectors in the quantitative semantics, and have a canonical mathematical interpretation — they correspond to the sums that one obtains when computing the derivative of a product of functions. The derivative operator allows to compute syntactically the optimal approximation of a program when applied to linear arguments.

As expected, iterated differentiation yields a natural notion of linear approximation of the ordinary application of a program to an input. This notion relates to ordinary application through the *Taylor formula* (see Ehrhard and Regnier [2008]):

$$f(x) = \sum_{n=0}^{\infty} \frac{1}{n!} (\mathtt{D}^n f \cdot x^n)0 \ , \tag{2.1}$$

(where $\mathtt{D}^2 f \cdot x^2 = \mathtt{D}(\mathtt{D}f \cdot x) \cdot x$ and so on). More generally, if one fully develops each application occurring in a program into its corresponding Taylor expansion, the result is an infinite sum of purely *differential programs* that contain only (multi)linear applications and applications to 0.

Species, operads and substitutions. Another closely related research area where power series are used in relationship with counting resources is analytic combinatorics as described in Flajolet and Sedgewick [2009]. Generating functions are one of the most efficient tools to count labelled structures (trees, lists, cycles,...) by solving the integro-differential equations that define these structures: those are power series such that the coefficient of their n-th monomial is the number of structures with n labels. This has led Joyal [1981] to generalize power series by replacing lists of coefficients with *species of structures* and power series by *analytic functors*. A species essentially consists of structures labelled by natural numbers, acted upon by permutations. Typically, the set of terms for some algebraic signature $\{f : 2, \ldots\}$ forms a species: permutations $\sigma \colon [n] \to [n]$ act on terms with variables in $[n] = \{1, \ldots, n\}$ by substitution. For instance, if σ swaps 1 and 2, we have $\sigma \cdot f(1,2) = f(2,1)$. Such an action may not be free (for instance, in the species of multisets of elements of $[n]$, any multiset consisting of several times the same element, i.e. $\{i, \ldots, i\}$, is invariant by permutation): the action thus carries more information than generating series.

Since their introduction, species and analytic functors have remained an active topic in combinatorics (see Chauve et al.; Jacquot [2014]; Vallette [2007]), where they are sometimes called grammars, but have also pervaded other areas of mathematics and computer science. For instance, in programming, they have inspired *indexed containers*, a particularly well-behaved class of data types, for which certain functions are automatically derivable (typically iterators and zippers, where the corresponding notion of differentiation of types is very similar to the one found in differential λ-calculus). Similarly, in semantics, analytic functors have been shown to form a model of LL in Hasegawa [2002] and of Differential LL in Hyvernat [2014]. Finally, in categorical algebra, species form the basis for *operads* (see Loday and Vallette [2012]), which are used to describe algebraic structures. Briefly, an operad is given by an underlying species of operations, together with a unital "multiplication" which describes composition of operations. Each operad thus generates a monad, whose algebras are precisely the desired algebraic structures.

More important, generalizations of species unifying combinatorics and semantics of LL have been proposed. In particular, the notion of *generalized species*, introduced in Fiore et al. [2008], is the 2-categorical version of Joyal's combinatorial species. Formally, a generalized species is a profunctor $F : \mathcal{L}A \nrightarrow B$ (that is a functor $B^{\mathrm{op}} \times \mathcal{L}A \to \mathbf{Set}$) where $\mathcal{L}$ is the 2-monad of free symmetric monoidal categories on categories. Concretely, for a category A, $\mathcal{L}A$ is the category of finite sequences $\langle a_1, \ldots, a_n \rangle$ of objects of A with concatenation as the symmetric monoidal product.

In particular, when applied to the terminal category $\mathbf{1}$, with one object and one morphism, this construction leads to $\mathbf{B}^{\mathrm{op}} = \mathcal{L}\mathbf{1}$, the category of finite cardinals and bijections. Notice that $\mathbf{B}^{\mathrm{op}}$ encapsulates the basic operation governing linear typing contexts: the exchange rule as described in Power and Tanaka [2005]. A standard species is nothing else but a single sorted generalized species $\mathcal{L}\mathbf{1} \nrightarrow \mathbf{1}$ (that is, a functor $\mathbf{B}^{\mathrm{op}} \to \mathbf{Set}$).

Generalized species can be used to represent a syntactic theory for a typed linear calculus as described in Tanaka [2000]. Let A denote the category of types and let $L : \mathcal{L}A \nrightarrow A$ represent the set of well typed terms: $L(\langle a_1, \ldots, a_n \rangle, a)$ is the set of terms t typed by the judgement $x_1 : a_1, \ldots, x_n : a_n \vdash t : a$. Substitution then yields a "multiplication" natural transformation $L \circ L \to L$, making L a monoid in the Kleisli bicategory of $\mathcal{L}$ in $\mathbf{Prof}$. This monoidal structure accounts for linear substitution: given terms $x_1 : a_1, \ldots, x_n : a_n \vdash t : a$ and $x_1^i : a_1^i, \ldots, x_{n_i}^i : a_{n_i}^i \vdash s_i : a_i$ (for $i = 1, \ldots, n$) in L, the monoid multiplication returns the substituted term $x_1 : a_1, \ldots, x_n : a_n \vdash t[s_i/x_i] : a$, which is again in L.

Several relevant structures with linear flavour have been shown to form monads in this Kleisli bicategory: the substitution structure of coloured operads (see Baez and Dolan [1995]), the linear substitution at the basis of LL, and the resource λ-calculus – the target of syntactic Taylor expansion (see Tsukada et al.). In this sense, composition in the Kleisli bicategory of $\mathcal{L}$ encodes an algebraic theory of linear substitution.

We have seen how generalized species arise naturally from the 2-monad $\mathcal{L}$ *extended* to $\mathbf{Prof}$. Because $\mathcal{L}$ is tightly linked with linear logic, it is natural to expect that a non-linear setting will be captured by moving from $\mathcal{L}$ to $\mathcal{M}$, the "free cartesian category" 2-monad. And indeed, composition in the Kleisli bicategory of $\mathcal{M}$ encodes an algebraic theory of cartesian substitution. More concretely, the objects of $\mathcal{M}A$ are the same as $\mathcal{L}A$, but the morphisms differ. In particular, when applied to $\mathbf{1}$, these constructions yields $\mathbf{F}^{\mathrm{op}}$ the cartesian category of finite cardinals equipped with the addition (as product) and morphisms $\alpha \in \mathcal{M}\mathbf{1}(k, m)$ consisting of functions $\alpha : [m] \to [k]$.

Notice that, from the structural rules point of view, $\mathbf{F}^{\mathrm{op}}$ covers non-linear operations such as forgetting and repeating. Indeed, for each m in $\mathbf{F}^{\mathrm{op}}$, there is the ith product projection $m \to 1$ corresponding to forgetting all m variables except the i-th one and the diagonal $1 \to m$ corresponding to repeating a variable m times (see Fiore et al. [1999]). Just as $\mathcal{L}$, the 2-monad $\mathcal{M}$ *extends* to the bicategory $\mathbf{Prof}$.

We can encode models of typed λ-calculus (with types in a category A) by a profunctor $M : \mathcal{M}A \to A$. Indeed, the composition in the Kleisli bicategory induces a monoidal structure $\mathcal{M} \circ \mathcal{M} \to \mathcal{M}$ that accounts for non-linear substitution. In particular, Hyland [2017] gives the abstract theory for pure λ-calculus as a profunctor $\mathcal{M}\mathbf{1} \nrightarrow \mathbf{1}$ endowed with retraction maps: $M(n+1) \to M(n)$ encoding abstraction (that binds the first of $n+1$ variables resulting in an n variables term) and $M(n) \to M(n+1)$ inducing application.

In this Chapter 2, we present a work resulting from a collaboration with Martin Hyland. This work is a first step in order to develop an adequate axiomatic theory based on category theory. As mentioned before, resource-sensitive calculi feature a notion of syntactic derivation, inducing a subtle interplay between linear and non-linear substitution.

Since its introduction by Ehrhard and Regnier [2003], there have been several equational axiomatisations of differential λ-calculus: Blute et al. [2006]; Ehrhard [2016]; Fiore [2007]. In this way, the French and Canadian schools have unified the syntactic derivation with the one found in usual calculus, through the use of categorical axiomatizations. These are based on the notion of *differential category*, which is a symmetric monoidal closed category with biproducts. The biproducts induce an additive structure on homsets, which is necessary for the equations (as for instance, the Leibniz rule) to even make sense. In a differential category, there is also a symmetric monoidal comonad (the exponential modality) denoted by !. The derivation is encoded thanks to the *codereliction* coder which induces a *deriving transformation* d when combined with coproduct:

$$\mathsf{coder}_A : A \to !A \qquad d_A : A \otimes !A \xrightarrow{\mathsf{coder}_A \otimes id_{!A}} !A \otimes !A \xrightarrow{\nabla} !A$$

Composing the deriving transformation with a non-linear morphism $f\colon A \otimes !A \xrightarrow{d_A} !A \xrightarrow{f} B$ produces the differential $d_A f \cdot u\ (x)$ of f in the direction given by u in A at x in $!A$. This deriving transformation has

to satisfy equations corresponding to standard rules of calculus and to other rules related to linearity and to the additive structure:

$\mathsf{coder}; e = 0$ (The derivative of a constant is 0)

$\mathsf{coder}; \Delta = \mathsf{coder} \otimes \nu + \nu \otimes \mathsf{coder}$ (Leibniz rule)

$\mathsf{coder}; \varepsilon = 1$ (The derivative of a linear function is a constant)

$(\mathsf{coder} \otimes 1); \nabla; \rho = (\mathsf{coder} \otimes \Delta); ((\nabla; \mathsf{coder}) \otimes \rho)); \nabla$ (Chain rule)

One of the main drawbacks of this presentation is that it is monolithic, in the sense that the axioms coming from the presence of substitution and those coming from the presence of derivation cannot be clearly identified. Instead, we would like to elaborate a construction which is similar to the one traditionally used in universal algebra (or its categorical framework given by Lawvere theories), which can be performed in two steps: first, the stage is set by considering categories with finite products and then, in that setting, particular theories may be defined (as free categories with finite products featuring such and such operations and equations). We aim at developing a similar distinction in the case of derivation by axiomatizing derivation on top of a primary setting, featuring a combination of linear and non-linear substitution.

Over the last decade, the theory of monads has been used to describe the structures of substitution and binding: Fiore et al. [1999]; Hirschowitz and Maggesi [2007]; Power and Tanaka [2005].The use of monads for these new purposes can be understood from the perspective of Kleisli bicategories (see Fiore et al. [2016]; Hyland [2014]): when extended to the bicategory of profunctors, 2-monads encapsulate a notion of substitution. In order to describe the world of the differential λ-calculus in primitive terms, one needs a setting in which both kinds of substitution (linear and non-linear) coexist. This suggests combining the free symmetric monoidal category and the free cartesian category monads. We plan to shift the focus from listing the axioms for a derivation theory to laying a foundation for an operadic theory emphasizing the interplay between linear and non-linear substitution. In this Chapter 2, we build this construction of monad and characterize its algebras.

Contributions

Finiteness Spaces. As mentioned before, they are vector spaces endowed with a linearized topology. Together with analytic functions, they constitue a model of simply typed λ-calculus. In Tasson and Vaux [2018], we studied type fixpoints in this model. We described a construction of finiteness spaces, and use it to prove the existence of least fixpoints of some functors. This result allowed us to interpret lazy recursive algebraic datatypes in finiteness spaces.

In addition, it is known from the very beginning that Finiteness Spaces cannot interpret fixpoint combinators, and so neither recursion nor untyped λ-calculus. Actually, finiteness spaces are related to the normalization property. In Pagani et al. [2016], we used finiteness spaces to characterize strongly normalizing terms of non-deterministic λ-calculus. For this, we interpreted a λ-term M as a power series in two steps, first by associating to M its Taylor expansion which is a sum of resource terms potentially containing redexes; second, by reducing each of the resource terms into a normal form and by summing the results. The problem is to characterize terms for which the resulting coefficients are finite, and which are thus strongly normalizable. We proved that *every strongly normalizable non-deterministic λ-term can be interpreted by a power series with finite coefficients.* It is a generalization of Ehrhard [2010] where terms typable by a non-deterministic variant of Girard's System $\mathcal{F}$ have always finite coefficients. Noteworthy, we generalize the proof technique of Ehrhard [2010] who uses a finiteness structure on the set of resource terms.

Vectorial semantics of Differential Linear logic. Although Finiteness Spaces constitute one of the model that gave rise to Differential Linear Logic, their *linearized* topology is unusual. We enlighten in Blute et al. [2012] the category of convenient vector spaces which is a model of Differential Linear Logic whose objects are topological vector spaces used in functional analysis. In this model, functions are interpreted as smooth function. Moreover, the differential operator of Differential Linear Logic is interpreted as the usual Gâteaux derivative in mathematics. Notice that much of the structure necessary to demonstrate that convenient vector spaces constitute a model were present in Kriegl and Michor [1997]. The interest of our work is to have extracted and revealed the structure of Linear Logic. As already mentioned, power series are a key notion in models of Linear Logic. It was then natural to refine

the model of convenient spaces and smooth functions into the model of convenient vector spaces and power series. More precisely, in Kerjean and Tasson [2018], we constructed an exponential whose Kleisli category is made of smooth functions that can be written as power series. Notice that it is still an open question to prove that this exponential is free.

Linear-Non-Linear substitution monad. In this Chapter 2, we develop a work in collaboration with Martin Hyland. As suggested in Power and Tanaka [2005], we propose to combine the operadic axiomatizations of linear substitution of Fiore et al. [2016] (given by the symmetric monoidal monad $\mathcal{L}$) and non-linear substitution of Hyland [2017, 2014] (given by the cartesian monoidal monad $\mathcal{M}$). There are many known ways to combine monads, for instance using distributive laws (see Cheng [2011]; Street [1972]) or by taking the sum or the tensor (see Hyland et al. [2006]). Unfortunately, none of these constructions is suitable for our setting and we investigate a new one based on lax colimits.

More precisely, note that any cartesian category is a symmetric monoidal category. Therefore, by the universal property of the free symmetric monoidal monad $\mathcal{L}$, there is a natural map $\mathcal{L}A \xrightarrow{k_A} \mathcal{M}A$. We construct the lax colimit $\mathcal{Q}A$ of this map and prove that $\mathcal{Q}$ is a 2-monad accounting for both linear and non-linear substitution. The resulting $\mathcal{Q}$ can be described similarly to $\mathcal{L}$ and $\mathcal{M}$ and the single typed variant $\mathcal{Q}\mathbf{1}$ corresponds to Tanaka and Power's abstract, mixed linear-non-linear variable binding. Concretely, for a category A, $\mathcal{Q}A$ is the category of finite sequences $\langle a_1, \dots, a_n, \underline{b}_1, \dots, \underline{b}_p \rangle$ with two types of objects a_i being linear and $\underline{b}_j$ being non-linear (duplicable and erasable).

Future works. To complete the picture, we will have to study, in future works, the linear-non-linear substitution structure and thus the operadic theories induced by $\mathcal{Q}$. We will have to extend $\mathcal{Q}$ to the Kleisli bicategory of profunctors. The first step will be to characterize lax $\mathcal{Q}$-algebras in order to show that the relative pseudomonad of presheaves has a lifting to $\mathcal{Q}$-algebras. This will give a new example of the constructions described in Fiore et al. [2016] and complete one step of the program described in Fiore [2006]. We will have to include derivative in this framework as an operator turning a non-linear context into a linear one. We will also incorporate the additive structure to recover a general theory of derivation (including the derivative of product, *Leibniz rule*).

Organization of the chapter

We begin in Section 2.1 with a presentation of monads and adjoints in a 2-categorical setting that will be needed in the following sections. Then, we describe the colimit construction of monads in Section 2.2. In Section 2.3, we prove a key Lemma that we then apply to show that our construction gives rise to a monad in Section 2.4 and to characterize its algebras in Section 2.5. In Section 2.6, we conclude by the example that we are interested in, that are the monads for symmetric monoidal categories and for cartesian categories.

2.1 Monads, adjoints and splittings

All over this Chapter 2, we will extensively use monads on the 2-category **Cat** of categories, functors and natural transformations. In this preliminary part, we recall the 2-categorical story for (co)monads, adjoints and splittings. Most of this appeared in Bénabou [1967], Street [1972] and Lack and Street [2002]. We follow terminology and definitions of Leinster [2004]. However, from our knowledge, Proposition 30 is new. As far as the reading of the remaining of this Chapter 2 is concerned, the important points are:

- Definition 19 of monad and map of monads
- Definitions 21 and 22 idempotent comonad,
- Definitions 28 of splitting,
- Proposition 29 stating a one-to-one correspondence between idempotent comonads and splittings,
- Proposition 30 that proves that a map of comonads between splittings can be splitted.

Although in the following sections, we will only use strict maps, we present the *lax* versions. Indeed, this work is preliminary and we will need the lax versions to lift to presheaves the comonad $\mathcal{Q}$ that we will build.

2.1.1 Monads and comonads

Definition 19. Let X be an object in a 2-category $\mathbf{C}$. A *monad* over X is a functor $X \xrightarrow{t} X$ with a unit $\eta : \mathrm{id}_X \Rightarrow t$ and a multiplication $\mu : t^2 \Rightarrow t$ satisfying the unit and associativity diagrams:

$$\begin{array}{ccccc} t & \overset{\eta\cdot t}{\Longrightarrow} & t^2 & \overset{t\cdot\eta}{\Longleftarrow} & t \\ & \searrow & \downarrow{\scriptstyle \mu} & \swarrow & \\ & & t & & \end{array} \qquad \text{and} \qquad \begin{array}{ccc} t^3 & \overset{\mu\cdot t}{\Longrightarrow} & t^2 \\ {\scriptstyle t\cdot\mu}\downarrow & & \downarrow{\scriptstyle \mu} \\ t^2 & \underset{\mu}{\Longrightarrow} & t \end{array} \tag{2.2}$$

A *lax map of monads* $(a, \varphi) : (X, t) \Rightarrow (X', t')$ is a functor $a : X \to X'$ and a 2-cell $\varphi : t'a \Rightarrow at$ satisfying the commutative diagrams for units and associativity:

$$\begin{array}{ccc} a & = & a \\ {\scriptstyle \eta a}\downarrow & & \downarrow{\scriptstyle a\eta} \\ t'a & \underset{\varphi}{\Longrightarrow} & at \end{array} \qquad \text{and} \qquad \begin{array}{ccccc} t'^2a & \overset{t'\varphi}{\Longrightarrow} & t'at & \overset{\varphi t}{\Longrightarrow} & at^2 \\ {\scriptstyle \mu a}\downarrow & & & & \downarrow{\scriptstyle a\mu} \\ t'a & & \underset{\varphi}{\Longrightarrow} & & at \end{array} \tag{2.3}$$

A *transformation* $\sigma : (a, \varphi) \to (a', \varphi')$ between map of monads is a 2-cell $\sigma : a' \Rightarrow a$ such that:

$$\begin{array}{ccc} t'a & \overset{\varphi}{\Longrightarrow} & at \\ {\scriptstyle t'\sigma}\downarrow & & \downarrow{\scriptstyle \sigma t} \\ t'a' & \underset{\varphi'}{\Longrightarrow} & a't \end{array} \tag{2.4}$$

This defines a 2-category $\mathbf{Mon}(\mathbf{C})$ whose objects are monads, whose functors are lax map of monads and whose 2-cells are monad functor transformations.

Thanks to the direction 2-cell in the definition of lax map of monads, the functor t together with the multiplication $(X, \mathrm{id}_X) \overset{(t,\mu)}{\Longrightarrow} (X, t)$ constitutes a lax map of monads.

In the following sections, we will use monads on **CAT** and their 2-category of algebras.

Definition 20. Let $(\mathcal{L}, \eta^{\mathcal{L}}, \mu^{\mathcal{L}})$ be a monad in **CAT**. The 2-category $\mathbf{Alg}(\mathcal{L})$ of $\mathcal{L}$-algebras is given by:
An algebra (A, a) is a category A with a functor $a : \mathcal{L}A \to A$ such that $a\eta^{\mathcal{L}} = \mathrm{id}_A$ and $a\mathcal{L}a = a\mu^{\mathcal{L}} : \mathcal{L}^2A \to A$.
A map of algebras $g : (A, a) \to (B, b)$ is a functor $g : A \to B$ such that $ga = b\mathcal{L}g : \mathcal{L}A \to B$.
A transformation of maps of algebras α between map of algebras $g, h : (A, a) \to (B, b)$ is a natural transformation $\alpha : g \Rightarrow h$ such that $b\mathcal{L}\alpha = \alpha a$.

Thanks to the direction 2-cell in its definition, a lax map of monads from $\mathcal{L}$ to $\mathcal{M}$ (monads in **CAT**) induces a map from $\mathcal{L}$-algebras to $\mathcal{M}$-algebras.

Definition 21. Let X be an object in the 2-category $\mathbf{C}$.

A *comonad* is a functor $X \xrightarrow{f} X$ together with a counit $f \overset{\varepsilon}{\Rightarrow} \mathrm{id}_X$ and a multiplication $f^2 \overset{\delta}{\Rightarrow} f$. which satisfies unit and associativity laws:

$$\begin{array}{ccccc} & & f & & \\ & \swarrow & \Downarrow\delta & \searrow & \\ f & \xleftarrow[\varepsilon\cdot f]{} & f^2 & \xrightarrow[f\cdot\varepsilon]{} & f \end{array} \qquad \text{and} \qquad \begin{array}{ccc} f^3 & \xleftarrow{\delta\cdot f} & f^2 \\ {\scriptstyle f\cdot\delta}\Uparrow & & \Uparrow{\scriptstyle \delta} \\ f^2 & \xleftarrow[\delta]{} & f \end{array} \tag{2.5}$$

A *lax map of comonads* (or perhaps colax opmap of comonads) $(a, \varphi) : (X, f) \Rightarrow (X', f')$ is a functor $a : X \to X'$ together with a 2-cell $\varphi : af \Rightarrow f'a$ satisfying the commutative diagrams for units and associativity:

$$\begin{array}{ccc} a & = & a \\ {\scriptstyle \varepsilon a}\Uparrow & & \Uparrow{\scriptstyle a\varepsilon} \\ f'a & \xleftarrow[\varphi]{} & af \end{array} \qquad \text{and} \qquad \begin{array}{ccccc} f'^2a & \xleftarrow{f'\varphi} & f'af & \xleftarrow{\varphi f} & af^2 \\ {\scriptstyle \delta a}\Uparrow & & & & \Uparrow{\scriptstyle a\delta} \\ f'a & & \xleftarrow[\varphi]{} & & af \end{array} \tag{2.6}$$

A *transformation* $\sigma : (a, \varphi) \to (a', \varphi')$ of map of comonads is a 2-cell $\sigma : a \Rightarrow a'$ such that:

$$\begin{array}{ccc} f'a & \xleftarrow{\varphi} & af \\ {\scriptstyle f'\sigma}\Downarrow & & \Downarrow{\scriptstyle \sigma f} \\ f'a' & \xleftarrow[\varphi']{} & a'f \end{array} \tag{2.7}$$

This defines a 2-category $\mathbf{CoMon}(\mathbf{C})$ whose objects are comonads, whose functors are lax map of comonads and whose 2-cells are comonad functor transformations.

Remark. The comonad $f : X \to X$ together with the comultiplication $\delta : f \Rightarrow f^2$ induces a map of comonads from (X, id_X) to (X, f).

In this chapter, we will extensively use pasting diagrams. The definition of comonads uses counit and comultiplication

$$\begin{array}{ccccc} X & \xrightarrow{f} & X & \xrightarrow{f} & X \\ & & \Downarrow\varepsilon & & \end{array} \qquad \text{and} \qquad \begin{array}{ccccc} & & \overset{f}{\frown} & & \\ X & \xrightarrow{f} & X & \xrightarrow{f} & X \end{array} \tag{2.8}$$

which satisfy unit law

$$\begin{array}{ccccc} & & \overset{f}{\frown} & & \\ X & \xrightarrow{f} & X & \xrightarrow{f} & X \\ & & \Downarrow\varepsilon & & \end{array} = \begin{array}{ccc} X & \xrightarrow{f} & X \\ & \underset{f}{\smile} & \| \end{array} = \begin{array}{ccccc} & & \overset{f}{\frown} & & \\ X & \xrightarrow{f} & X & \xrightarrow{f} & X \\ & & & & \Downarrow\varepsilon \end{array} \tag{2.9}$$

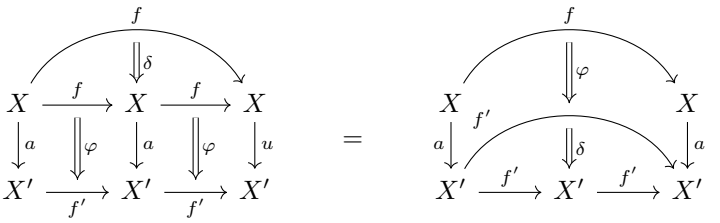
and associativity

$$\begin{array}{ccccccc} & & & \overset{f}{\frown} & & & \\ & & \overset{f}{\frown} & & & \Downarrow\delta & \\ X & \xrightarrow{f} & X & \xrightarrow{f} & X & \xrightarrow{f} & X \end{array} = \begin{array}{ccccccc} & & & \overset{f}{\frown} & & & \\ & & \Downarrow\delta & & & \overset{f}{\frown} & \\ X & \xrightarrow{f} & X & \xrightarrow{f} & X & \xrightarrow{f} & X \end{array} \tag{2.10}$$

The definition of lax map of comonads can be rephrased as

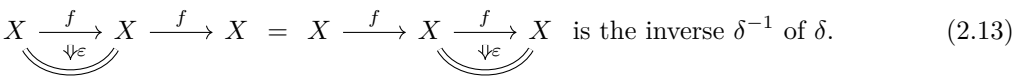
$$\begin{array}{ccc} X & \xrightarrow{f} & X \\ {\scriptstyle a}\downarrow & \Downarrow{\scriptstyle \varphi} & \downarrow{\scriptstyle a} \\ X' & \xrightarrow[\Downarrow\varepsilon]{f'} & X' \end{array} = \begin{array}{ccc} X & \xrightarrow{f} & X \\ & \Downarrow\varepsilon & \downarrow{\scriptstyle a} \\ & & X' \end{array} \tag{2.11}$$

and



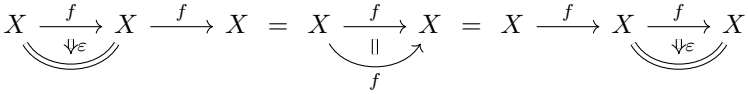
(2.12)

Definition 22. An *idempotent* comonad is a comonad whose comultiplication is invertible. In this case, $\delta^{-1} = \varepsilon \cdot f = f \cdot \varepsilon$, in other words



(2.13)

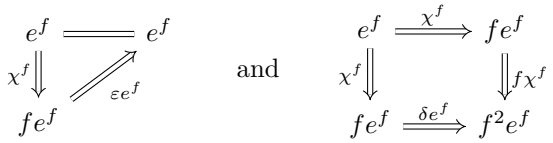
A *strict idempotent* comonad is a comonad whose comultiplication δ is the identity. In this case, $f^2 = f$ and



(2.14)

Eilenberg-Moore objects correspond to algebras for a monad. In Street [1976], it was shown that Eilenberg-Moore objects are weighted limits in the 2-category $\mathbf{C}$. From now on, we assusme $\mathbf{C}$ has all limits.

Definition 23. Let (X, f) be a comonad in a 2-category $\mathbf{C}$. The *Eilenberg-Moore* object of f consists of an object X^f equipped with a functor $e^f : X^f \to X$ and a 2-cell $\chi^f : e^f \Rightarrow fe^f$ such that $(e^f, \chi^f) : (X^f, \mathrm{id}_{X^f}) \Rightarrow (X, f)$ is a lax map of comonads, that is:

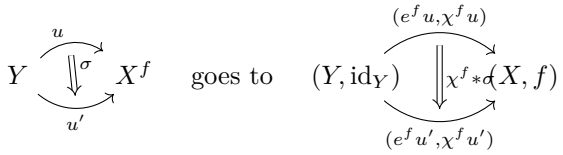


(2.15)

and there is a natural isomorphism:

$$\mathbf{C}(Y, X^f) \simeq \mathbf{coMnd}(\mathbf{C})((Y, \mathrm{id}_Y), (X, f)) \tag{2.16}$$

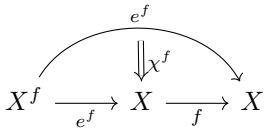
given by:



(2.17)

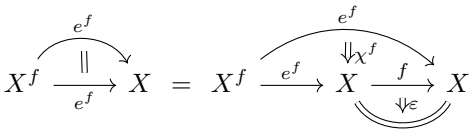
Remark. Because $(e^f, \chi^f) : (X^f, \mathrm{id}_{X^f}) \Rightarrow (X, f)$ is a lax map of comonads, $\chi^f : (e^f, \chi^f) \Rightarrow (fe^f, \delta e^f)$ is a transformation of lax maps of comonads (see definition Diagram (2.7) and right Diagram of (2.15)).

We can rephrase the condition of the above definition in terms of pasting diagrams using the 2-cell



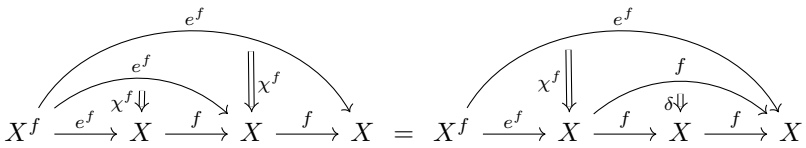
(2.18)

as



(2.19)

and



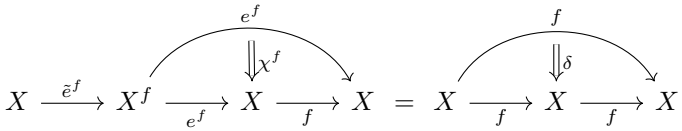
(2.20)

Proposition 24. *Let $X \xrightarrow{f} X$ be a comonad with counit ε and comultiplication δ. There is $\tilde{e}^f : X \to X^f$ and $\eta : e^f \tilde{e}^f \Rightarrow \mathrm{id}_X$ such that*

$$f = e^f \tilde{e}^f, \qquad \delta = \chi^f \tilde{e}^f, \qquad \chi^f = e^f \eta \qquad \text{and} \qquad \chi^f = e^f \varepsilon. \tag{2.21}$$

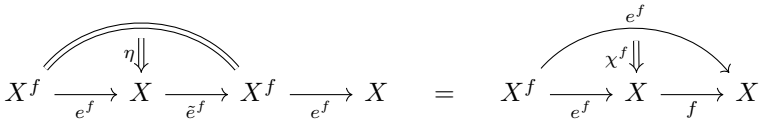
Moreover, if f is an idempotent comonad, then $\chi^f = e^f \eta$ and εe^f are inversed and so isomorphisms and if f is a strict idempotent comonad, then $\mathrm{id}^f_X = \tilde{e}^f e^f$.

Proof. Since $(f, \delta) : (X, \mathrm{id}_X) \to (X, f)$ is a map of comonad (see remark Page 38) and by 1-cell universality of X^f (Property (2.16)), there exists a unique 1-cell $\tilde{e}^f : X \to X^f$ such that:



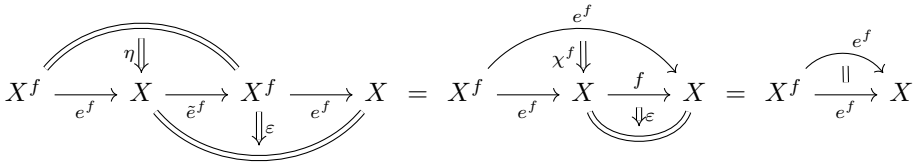
(2.22)

We define the η by 2-cell universality of X^f (Property (2.16)). Indeed, we noticed Page 39 that $\chi^f : (e^f, \chi^f) \Rightarrow (fe^f, \delta e^f)$ is a transformation of maps of comonads, so by 2-cell universality (Property (2.16)), there is a unique 2-cell $\eta : \mathrm{id}_{X^f} \Rightarrow \tilde{e}^f e^f$ such that $e^f \eta = \chi^f$, that is:



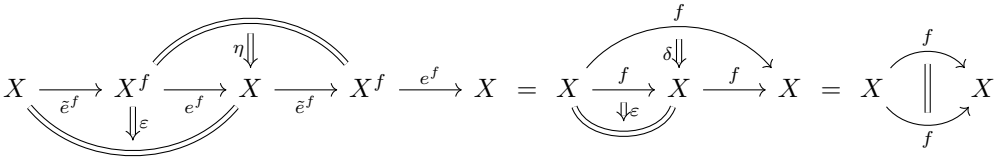
(2.23)

It follows, by Diagrams (2.19) and (2.23), that



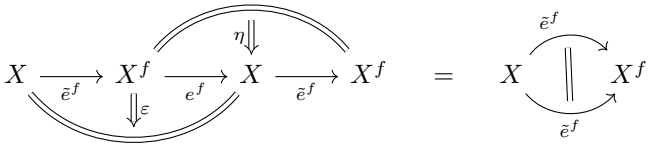
(2.24)

Then, by Diagrams (2.22) and (2.23),



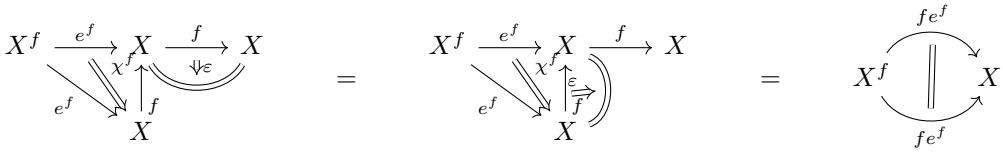
(2.25)

So that, by 2-cell universality of Eilenberg-Moore object (Property 2.16),:



(2.26)

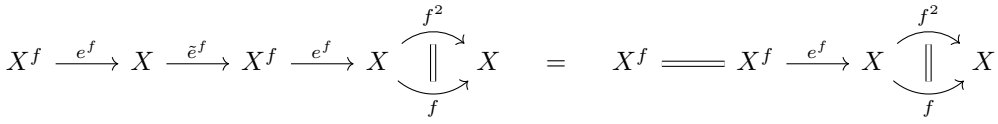
If f is an *idempotent* comonad, let us prove that χ^f is the inverse of εe^f. Indeed, from the definition of the Eilenberg-Moore object, the composite $e^f \overset{\chi^f}{\Longrightarrow} fe^f \overset{\varepsilon e^f}{\Longrightarrow} e^f$ is equal to $e^f \overset{\mathrm{id}}{\Rightarrow} e^f$ by Diagram (2.19). Moreover, by Diagram (2.13),



(2.27)

Hence, $\chi^f = e^f \eta$ and εe^f are inversed and so isomorphisms.

Assume now that f is a *strict* idempotent comonad, that is $\delta : f^2 = f$. Then, both id_{X^f} and $\tilde{e}^f e^f$ factorize the cone δe^f:



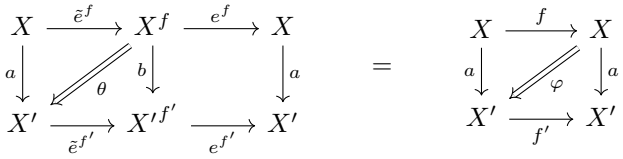
Indeed, $f^2 e^f \tilde{e}^f = f^2 f = f^2$ for the upper map and $f e^f \tilde{e}^f = f^2 = f$ for the lower map By 1-cell universality, we conclude that $\mathrm{id}_{X^f} = \tilde{e}^f e^f$. □

The following Proposition 25 corresponds to a comonadic presentation of Lemma 6.6.1 in Leinster [2004].

Proposition 25. *Let $X \xrightarrow{f} X$ and $X' \xrightarrow{f'} X'$ be two comonads. There is a one-to-one correspondence between lax map of comonads $(a, \varphi) : (X, f) \Rightarrow (X', f')$ and pairs (a, b) of functors such that the following square commutes:*

$$\begin{array}{ccc} X^f & \xrightarrow[e^f]{} & X \\ {\scriptstyle b}\downarrow & & \downarrow{\scriptstyle a} \\ X'^{f'} & \xrightarrow[e^{f'}]{} & X' \end{array} \tag{2.28}$$

Moreover, if there is a lax map of comonads $(a, \varphi) : (X, f) \Rightarrow (X', f')$, then there is $\theta : b\,\tilde{e}^f \Rightarrow \tilde{e}^{f'} a$ such that:



Proof. The composite $(X^f, \mathrm{id}_{X^f}) \overset{(e^f, \chi^f)}{\Longrightarrow} (X, f) \overset{(a,\varphi)}{\Longrightarrow} (X', f')$ is a map of comonads. Thus, by the 1-cell universality (Property (2.16)) of $X'^{f'}$, there is $b : X^f \to X'^{f'}$ such that $e^{f'} b = a e^f$ which corresponds to the Square (2.28).

For the converse, using the Square (2.28) and Equation (2.21), we build the lax map of comonads $af \Rightarrow f'a$ as the composite:

$$af = a e^f \tilde{e}^f = e^{f'} b \tilde{e}^f \overset{\chi^{f'} b \tilde{e}^f}{\Longrightarrow} f' e^{f'} b \tilde{e}^f = f' a e^f \tilde{e}^f = f' a f \overset{f' a \eta}{\Longrightarrow} f' a$$

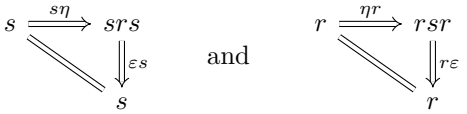
The 2-cell $\varphi : af \Rightarrow f'a$ is a transformation of maps of comonads from $(e^{f'} b \tilde{e}^f, \varphi f) : (X, \mathrm{id}_X) \to (X', f')$ to $(e^{f'} \tilde{e}^{f'} a, f'\varphi) : (X, \mathrm{id}_X) \to (X', f')$ because $e^f b \tilde{e}^f = a e^f \tilde{e}^f = af$ and $e^{f'} \tilde{e}^{f'} a = f'a$. Indeed, the transformation condition (2.7) boils down to the obvious square:

$$\begin{array}{ccc} af^2 & \overset{\varphi f}{\Longrightarrow} & f'af \\ {\scriptstyle \varphi f}\Downarrow & & \Downarrow{\scriptstyle f'\varphi} \\ f'af & \underset{f'\varphi}{\Longrightarrow} & f'^2 a \end{array}$$

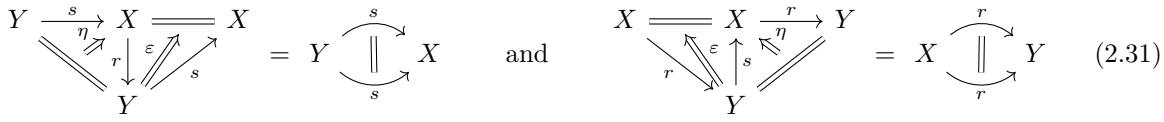
The existence of $\theta : b\,\tilde{e}^f \Rightarrow \tilde{e}^{f'} a$ and Diagram (2.28) follows by 2-cell universality (Property (2.16)). □

2.1.2 Adjoints

Definition 26. Let X and Y be two objects in a 2-cateory $\mathbf{C}$. An adjunction $Y \underset{s}{\overset{r}{\underset{\longrightarrow}{\overset{\longleftarrow}{\top}}}} X$ between the functors r and s is given by the unit $\eta : \mathrm{id}_Y \Rightarrow rs$ and the counit $\varepsilon : sr \Rightarrow \mathrm{id}_Y$ satisfying the triangle equalities:



which can be rephrased as diagrams:



(2.31)

Of course, an adjunction induces a monad rs on Y and a comonad sr on X.

Map of adjunctions. Suppose $Y \underset{s}{\overset{r}{\leftarrow \top \rightarrow}} X$ and $W \underset{v}{\overset{u}{\leftarrow \top \rightarrow}} Z$ are adjoints with units $\eta : \mathrm{id}_Y \Rightarrow rs$ and $\eta : \mathrm{id}_W \Rightarrow uv$ and counits $\varepsilon : sr \Rightarrow \mathrm{id}_Y$ and $\varepsilon : vu \Rightarrow \mathrm{id}_W$. There are two good notions of map of adjunctions from $(s \dashv r)$ to $(v \dashv u)$.

In both cases, we have functors $X \xrightarrow{a} Z$ and $Y \xrightarrow{b} W$ as in the diagram:

$$\begin{array}{ccc} Y & \underset{s}{\overset{r}{\leftarrow \top \rightarrow}} & X \\ {\scriptstyle b}\downarrow & & \downarrow{\scriptstyle a} \\ W & \underset{v}{\overset{u}{\leftarrow \top \rightarrow}} & Z \end{array}$$

From this diagram, we see that we could consider 2-cells:

$$\varphi : vb \Rightarrow as \qquad\qquad \theta : br \Rightarrow ua$$
$$\chi : as \Rightarrow vb \qquad\qquad \psi : ua \Rightarrow br.$$

In both cases, we have φ and θ and, as one might naturally suppose, they are mates: that is,
θ is the composite given by the composite $br \overset{\eta br}{\Longrightarrow} uvbr \overset{u\varphi r}{\Longrightarrow} uasr \overset{ua\varepsilon}{\Longrightarrow} ua$,
φ is the composite given by the composite $vb \overset{vb\eta}{\Longrightarrow} vbrs \overset{v\theta s}{\Longrightarrow} vuas \overset{\varepsilon as}{\Longrightarrow} as$.

- In one case, maybe the *lax* case, we have $\psi : ua \Rightarrow br$ and the natural unit an counit conditions holds:

  So, the main data are a, b, φ, ψ. It is then automatic that ψ is an isomorphism with inverse θ, the mate of φ.

- In the other case, maybe the *colax* case, we have $\chi : as \Rightarrow vb$ and the natural unit and counit condition holds for θ and χ: the main data are a, b, θ, χ.

  It is then automatic that χ is an isomorphism with inverse φ, the mate of θ.

If (a, b, φ, ψ) gives a lax map of adjunctions, then we get a map of monads $(Y, rs) \to (W, uv)$ with 2-cell $uvb \Rightarrow brs$ given by the composite $uvb \overset{u\varphi}{\Longrightarrow} uas \overset{\psi s}{\Longrightarrow} brs$. This direction of 2-cell induces a functor between the corresponding Eilenberg-Moore categories.

Also, one gets what might be called an op-map of comonads $(X, sr) \to (Z, vu)$ with 2-cell $vua \Rightarrow asr$ given by the composite $vua \overset{v\psi}{\Longrightarrow} vbr \overset{\varphi r}{\Longrightarrow} asr$. This direction of 2-cell induces a functor between the corresponding Kleisli categories.

In this work, we will use the dual story for the colax map (a, b, θ, χ) of adjunctions. In particular, we get a lax map of comonad (following Definition 21) $(X, sr) \to (Z, vu)$ with functor a and 2-cell $asr \Rightarrow vua$ given by the composite $asr \overset{\chi r}{\Longrightarrow} vbr \overset{v\theta}{\Longrightarrow} vua$. Actually, the converse is also true.

Proposition 27. *There is a one-to-one correspondence between adjoints with colax maps of adjunctions and comonads with lax maps of comonads.*

Proof. An adjoint induces a comonad. We have already noticed that a pair of adjoints $Y \underset{s}{\overset{r}{\leftarrows \top \rightarrow}} X$ with unit η and counit ε induces a comonad $X \xrightarrow{sr} X$ with counit ε and comultiplication δ given by the composite: $sr \xRightarrow{s\eta r} srsr$. We also just noticed that a colax map of adjunctions induces a lax map of comonad.

A comonad induces an adjoint. Let $X \xrightarrow{f} X$ be a comonad with counit ε and comultiplication δ. We set $Y = X^f$ the Eilenberg-Moore object, $s = e^f : X^f \to X$. Proposition 24 gives us the right adjoint $r = \tilde{e}^f : X \to X^f$ and the unit η.

Let $(a, \varphi) : (X, f) \Rightarrow (X', f')$ be a lax map of comonad (see Definition 21). By Proposition 25, there is a map $b : X^f \to X'^{f'}$ and a 2-cell $\theta : b\,\tilde{e}^f \Rightarrow \tilde{e}^{f'} a$ such that:

$$\begin{array}{ccccc} X & \xrightarrow{\tilde{e}^f} & X^f & \xrightarrow{e^f} & X \\ {\scriptstyle a}\downarrow & \swarrow_{\theta} & {\scriptstyle b}\downarrow & & \downarrow{\scriptstyle a} \\ X' & \xrightarrow[\tilde{e}^{f'}]{} & X'^{f'} & \xrightarrow[e^{f'}]{} & X' \end{array} \quad = \quad \begin{array}{ccc} X & \xrightarrow{f} & X \\ {\scriptstyle a}\downarrow & \swarrow_{\varphi} & \downarrow{\scriptstyle a} \\ X' & \xrightarrow[f']{} & X' \end{array} \tag{2.32}$$

Thus, we have a colax map of adjunctions $(a, b, \theta, =)$. □

2.1.3 Splittings

We are now interested in (strict) idempotent comonads and splittings in a 2-category. This is a generalization to 2-categories of what happens in the category of categories, as presented in [Borceux, 1994, Part 4.2].

Definition 28. Let $\mathbf{C}$ be a 2-category.

A *splitting* is an adjoint pair such that $s\eta$ (or equivalently εs) is an isormorphism. For a *strict*splitting, $sr = \mathrm{id}_X$. A *map of splitting* is a colax map of adjunctions between two splittings.

Proposition 29. *There is a one-to-one correspondence between idempotent comonads with lax map of comonads and splittings with map of splittings. There is a one-to-one correspondance between strict idempotent comonads with strict maps and strict splitting with strict map of splittings.*

Proof. It is a direct consequence of Proposition 27 by noticing that if an adjunction is a splitting, then ηs is an isomorphism, then the comultiplication $\delta = r\eta s$ of the associated comonad $f = sr$ is also an isomorphism. Conversely, by Proposition 24 if f is an idempotent comonad, then the pair of adjoints $X^f \underset{e^f}{\overset{\tilde{e}^f}{\leftarrows \top \rightarrow}} X$ is a splitting as $e^f\eta$ and εe^f are inverse from each other and so both isomorphisms. By the same Proposition 24, if f is a strict idempotent comonad, then $\tilde{e}^f e^f = \mathrm{id}_Y$, that is the splitting is strict. □

Proposition 30. *Suppose* $Y \underset{s}{\overset{r}{\leftarrows \top \rightarrow}} X$ *and* $Y' \underset{s'}{\overset{r'}{\leftarrows \top \rightarrow}} X'$ *are two splittings with units* $\eta : \mathrm{id}_Y \Rightarrow rs$ *and* $\eta : \mathrm{id}_{Y'} \Rightarrow r's'$ *and counits* $\varepsilon : sr \Rightarrow \mathrm{id}_X$ *and* $\varepsilon : s'r' \Rightarrow \mathrm{id}_{X'}$. *A map of comonads* $(a, \varphi) : sr \Rightarrow s'r'$ *induces a map of splitting* (a, b, θ, χ) *such that:*

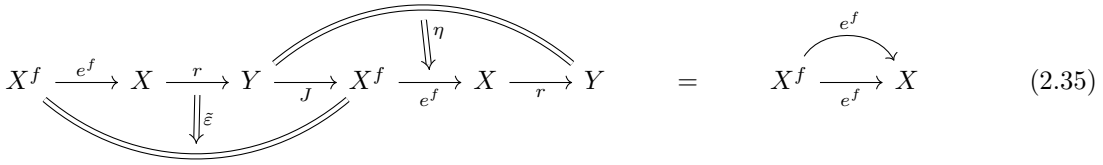
$$\begin{array}{ccccc} X & \xrightarrow{r} & Y & \xrightarrow{s} & X \\ {\scriptstyle a}\downarrow & \swarrow_{\varphi} & & & \downarrow{\scriptstyle a} \\ X' & \xrightarrow[r']{} & Y' & \xrightarrow[s']{} & X' \end{array} \quad = \quad \begin{array}{ccccc} X & \xrightarrow{r} & Y & \xrightarrow{s} & X \\ {\scriptstyle a}\downarrow & \swarrow_{\theta} & {\scriptstyle b}\downarrow & \swarrow_{\chi} & \downarrow{\scriptstyle a} \\ X' & \xrightarrow[r']{} & Y' & \xrightarrow[s']{} & X' \end{array} \tag{2.33}$$

Proof. An equivalence of categories. Starting from a pair of adjoints, $Y \underset{s}{\overset{r}{\leftarrows \top \rightarrow}} X$ we show that there is an equivalence of categories between the Eilenberg-Moore object of the comonad induced by the adjunction and Y.

We construct $J : Y \to X^f$ by to 1-cell universality (Property 2.16). Indeed $(s, s\eta)$ is a map of comonads from $(Y, \mathrm{id}_Y) \Rightarrow (X, f)$ (recall that $f = sr$). Thus, $e^f J = s$ and $s\eta = \chi^f J$:

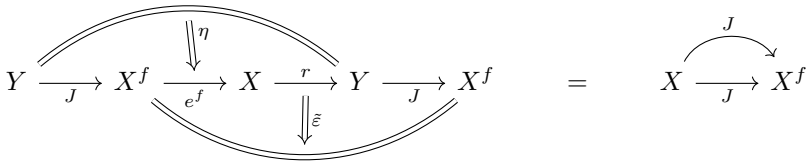
$$\begin{array}{ccccc} & & \overset{\Downarrow \eta}{\frown} & & \\ Y & \xrightarrow{s} & X & \xrightarrow{r} & Y \xrightarrow{s} X \end{array} \quad = \quad \begin{array}{ccccccc} & & & & \overset{e^f}{\frown} & & \\ Y & \xrightarrow{J} & X^f & \xrightarrow{e^f} & X & \xrightarrow[r]{\Downarrow \chi^f} & Y \xrightarrow{s} X \end{array} \tag{2.34}$$

We remarked in Page 39 that $\chi^f = e^f\eta : (e^f, \chi^f) \Rightarrow (fe^f, \delta e^f)$ is a transformation of maps of comonads. By proposition 29, $f = sr$ is an idempotent comonad. By Proposition 24, $\varepsilon e^f : (fe^f, \delta e^f) \Rightarrow (e^f, \chi^f)$ is the inverse of $e^f\eta$. Thus, εe^f is a transformation of maps of comonads. Because $fe^f = e^f Jre^f$ and by 2-cell universality (Property 2.16), the isomorphic transformation $\varepsilon e^f : (e^f Jre^f, \delta e^f) \Rightarrow (e^f, \chi^f)$ corresponds to the isomorphic 2-cell: $\tilde{\varepsilon} : Jre^f \Rightarrow \mathrm{id}_{X^f}$. Moreover, η is the mate of $\tilde{\varepsilon}$:



(2.35)

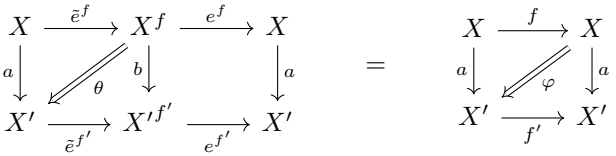
and



(2.36)

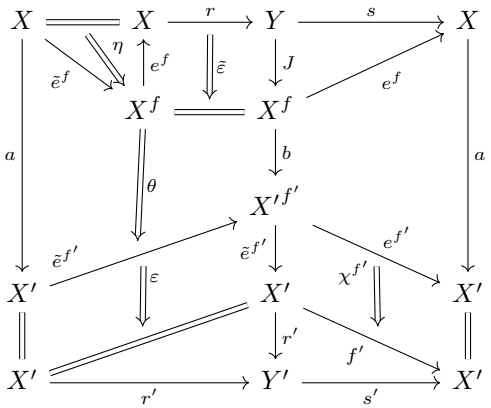
Together, they induce the equivalence of categories between X^f and Y.

A lax map of comonads induces a map of splittings. Consider two pair of adjoints $Y \underset{s}{\overset{r}{\underset{\longrightarrow}{\overset{\longleftarrow}{\top}}}} X$ and $Y \underset{s'}{\overset{r'}{\underset{\longrightarrow}{\overset{\longleftarrow}{\top}}}} X$ and a lax map of comonads $(a, \varphi) : sr \Rightarrow s'r'$. By Proposition 25. There are $b : X^f \to X'^{f'}$ and $\theta : b\,\tilde{e}^f \Rightarrow \tilde{e}^{f'}a$ such that:



(2.37)

The map of splitting is given by the pasting diagram:



(2.38)

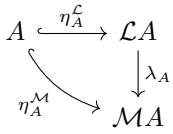
We check that it is indeed a colax map of adjunctions. □

From now on, we will only consider *strict* maps of comonads and splitting.

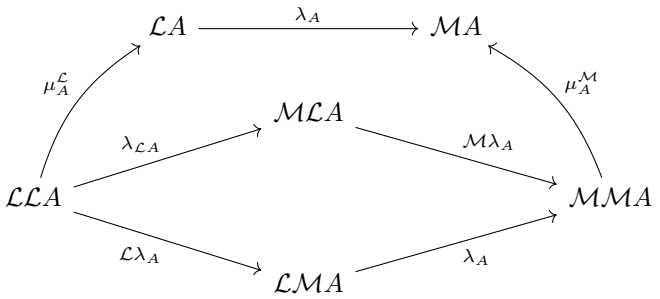
2.2 A colimit of monads

We consider two monads $\mathcal{L}$ and $\mathcal{M}$ on **Cat** with units $\eta^{\mathcal{L}}$ and $\eta^{\mathcal{M}}$ and with multiplications $\mu^{\mathcal{L}}$ and $\mu^{\mathcal{M}}$. From now on, we work in the 2-category $\mathbf{Alg}(\mathcal{L})$ of $\mathcal{L}$-algebras and strict maps (see Definition 20). We assume that $\mathcal{L}$ is finitary, that is its underlying functor preserves filtered colimits. It implies that the 2-category of algebras $\mathbf{Alg}(\mathcal{L})$ is complete and cocomplete, so that we can compute colimits. Note that our constructions does not take place in the 2-category $\mathbf{Mon}(\mathbf{C})$ of 2-monads, but in the 2-category $\mathbf{Alg}(\mathcal{L})$ of $\mathcal{L}$-algebras.

We assume that there is a strict map of monad $(\mathrm{id}, \lambda) : (\mathbf{Cat}, \mathcal{L}) \Rightarrow (\mathbf{Cat}, \mathcal{M})$. So that, by Diagrams (2.3) of Definition 19, the following standard diagrams commute for any category A:



(2.39)



(2.40)

Remark that $\lambda : \mathcal{L} \to \mathcal{M}$ makes any $\mathcal{M}$-algebra an $\mathcal{L}$-algebra. Besides, $\lambda_A : \mathcal{L}A \to \mathcal{M}A$ is a map of $\mathcal{L}$-algebras from the free $\mathcal{L}$-algebra $\mu^{\mathcal{L}}_A : \mathcal{L}\mathcal{L}A \to \mathcal{L}A$ to the $\mathcal{L}$-algebra $\mathcal{L}\mathcal{M}A \xrightarrow{\lambda_{\mathcal{M}A}} \mathcal{M}\mathcal{M}A \xrightarrow{\mu^{\mathcal{M}}_A} \mathcal{M}A$ induced by the free $\mathcal{M}$-algebra.

We will build a new monad $\mathcal{Q}$ as a colimit over the map of monad $\lambda : \mathcal{L} \to \mathcal{M}$. A concrete example of this construction is given in Section 2.6.

2.2.1 The lax colimit $\mathcal{Q}$.

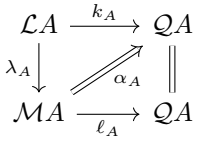
From now on, we fix a category A. Nevertheless, all the construction that appears in what follows are functorial and natural with respect to this category.

In $\mathbf{Alg}(\mathcal{L})$, we take the lax colimit $\mathcal{Q}A$ of $\lambda_A : \mathcal{L}A \to \mathcal{M}A$ as a map of $\mathcal{L}$-algebras.

Let ΔC be the constant functor with value C and $\mathbf{OpLax}(\lambda_A, \Delta C)$ be the category of *oplax* natural transformations from λ_A to ΔC and modifications between them[1]. Then, there is a natural isomorphism:

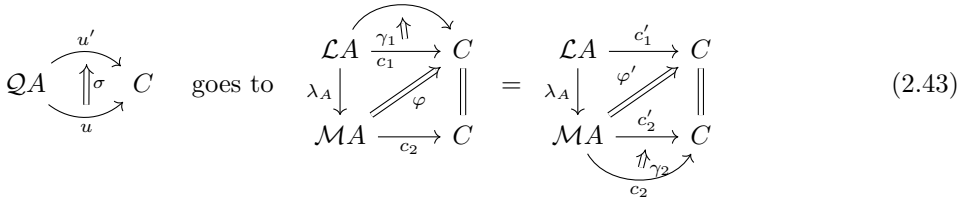
$$\mathbf{Alg}(\mathcal{L})(\mathcal{Q}A, C) \simeq \mathbf{OpLax}(\mathcal{L}A \xrightarrow{\lambda_A} \mathcal{M}A, \Delta C) \tag{2.41}$$

over the universal cone (2.42) in $\mathbf{Alg}(\mathcal{L})$:



(2.42)

given by:



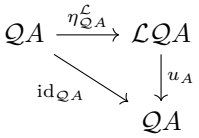
(2.43)

The universal property of $\mathcal{Q}A$ has both 1-cell: $uk_A = c_1$, $u\ell_A = c_2$, $u\lambda_A = \varphi$ and 2-cell aspects: $\sigma k_A = \gamma_1$ and $\sigma \ell_A = \gamma_2$.

The universal property at once gives $\mathcal{Q}$ the structure of a 2-functor, we shall use that functoriality and the associated naturality of the constructions without further comment.

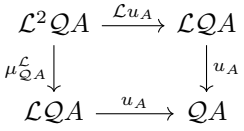
Moreover, using the fact that the colimit is taken in the 2-category $\mathbf{Alg}(\mathcal{L})$, we compute the data:

- Since $\mathcal{Q}A$ is itself an $\mathcal{L}$-algebra, we have $\mathcal{L}\mathcal{Q}A \xrightarrow{u_A} \mathcal{Q}A$ with the following commuting diagrams:



(2.44)

and



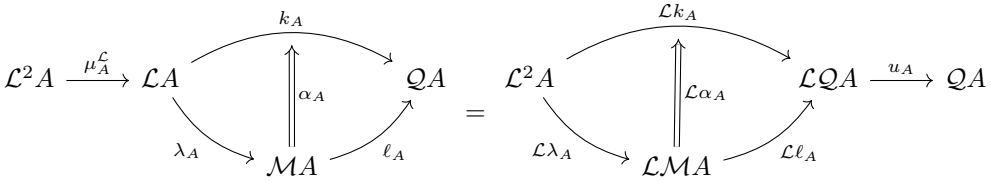
(2.45)

- Since $\mathcal{L}A \xrightarrow{k_A} \mathcal{Q}A$ and $\mathcal{M}A \xrightarrow{\ell_A} \mathcal{Q}A$ are $\mathcal{L}$-algebra maps, we get the commuting diagrams:

1. Notice that we get oplax natural transformations when taking a lax colimit over $\mathbf{Alg}(\mathcal{L})$ since it is a limit over $\mathbf{Alg}(\mathcal{L})^{\mathrm{op}}$.

$$\begin{array}{ccc} \mathcal{L}^2A & \xrightarrow{\mathcal{L}k_A} & \mathcal{L}\mathcal{Q}A \\ {\scriptstyle \mu^{\mathcal{L}}_A}\downarrow & & \downarrow{\scriptstyle u_A} \\ \mathcal{L}A & \xrightarrow{k_A} & \mathcal{Q}A \end{array} \qquad (2.46) \qquad \text{and} \qquad \begin{array}{ccc} \mathcal{L}\mathcal{M}A & \xrightarrow{\mathcal{L}\ell_A} & \mathcal{L}\mathcal{Q}A \\ {\scriptstyle \lambda_{\mathcal{M}A}}\downarrow & & \\ \mathcal{M}^2A & & \downarrow{\scriptstyle u_A} \\ {\scriptstyle \mu^{\mathcal{M}}_A}\downarrow & & \\ \mathcal{M}A & \xrightarrow{\ell_A} & \mathcal{Q}A \end{array} \qquad (2.47)$$

- Since α_A is an $\mathcal{L}$-algebra 2-cell and by commuting Diagrams (2.40), (2.46) and (2.47), the following pastings are equal 2-cells:

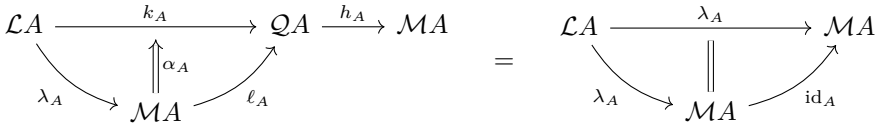


(2.48)

2.2.2 The splitting of $\mathcal{Q}$

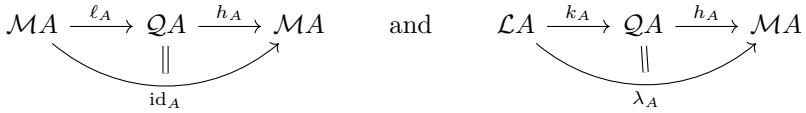
In this paragraph, we show that $\ell_A : \mathcal{M}_A \to \mathcal{Q}_A$ has a right adjoint $h_A : \mathcal{Q}_A \to \mathcal{M}_A$ whose unit is the identity and whose counit β_A is idempotent. Thus, the pair of adjoints $\mathcal{M}_A \underset{\ell_A}{\overset{h_A}{\underset{\longrightarrow}{\overset{\longleftarrow}{\top}}}} \mathcal{Q}_A$ is a splitting (see Definition 28). Thus, by Proposition 29, $h_A\ell_A$ is an idempotent comonad (see Definition 21).

A retract. The 1-cell universality of the colimit applied to the evident identity 2-cells gives the existence of the unique map $\mathcal{Q}A \xrightarrow{h_A} \mathcal{M}A$ of $\mathcal{L}$-algebras such that:



(2.49)

We see from the 1-cell universality, that we get the following commutative diagrams:



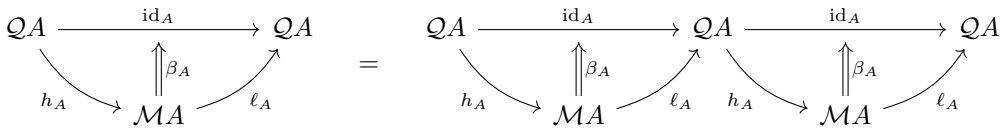
(2.50)

so that from the left hand side, we get that $\mathcal{M}A \xrightarrow{\ell_A} \mathcal{Q}A$ and $\mathcal{Q}A \xrightarrow{h_A} \mathcal{M}A$ present $\mathcal{M}A$ as a retract of $\mathcal{Q}A$ in $\mathbf{Alg}(\mathcal{L})$.

The fact that h_A is a map of $\mathcal{L}$-algebra gives the commuting diagram:

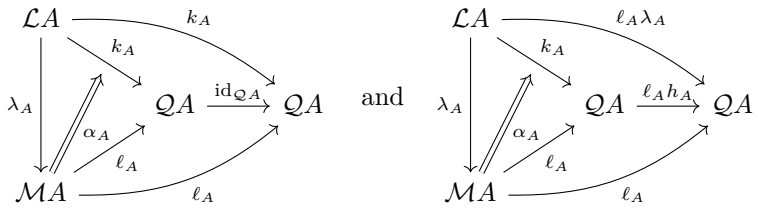
$$\begin{array}{ccc} \mathcal{L}\mathcal{Q}A & \xrightarrow{\mathcal{L}h_A} & \mathcal{L}\mathcal{M}A \\ & & \downarrow{\scriptstyle \lambda_{\mathcal{M}A}} \\ {\scriptstyle u_A}\downarrow & & \mathcal{M}^2A \\ & & \downarrow{\scriptstyle \mu^{\mathcal{M}}_A} \\ \mathcal{Q}A & \xrightarrow{h_A} & \mathcal{M}A \end{array} \qquad (2.51)$$

An idempotent counit. We now prove the the existence of a 2-cell β_A such that:



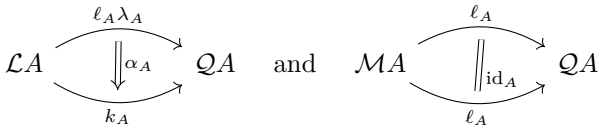
(2.52)

The 2-cell β_A comes from the 2-cell universality of the colimit. The two cones



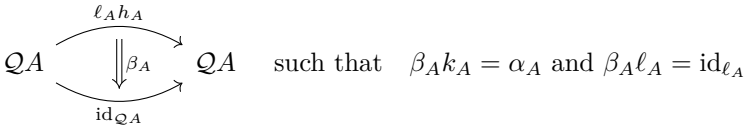
(2.53)

are in correspondance by the two 2-cells:



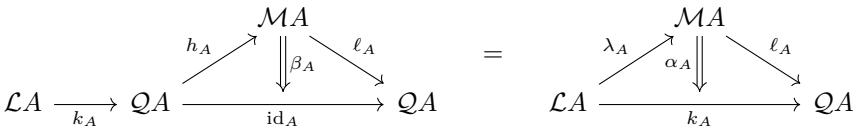
(2.54)

So that, there is a unique 2-cell:

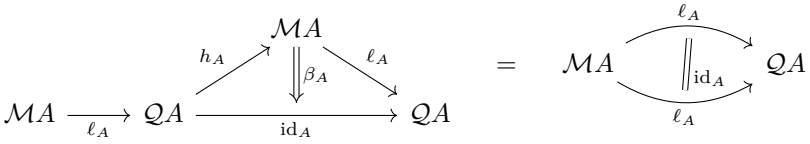


(2.55)

This can be rephrased by diagrams:



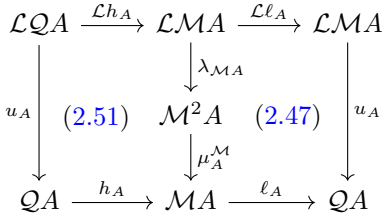
(2.56)



(2.57)

We conclude the idempotence of β_A from Diagram (2.57). Moreover, $\beta_A k_A \eta^{\mathcal{L}}_A = \alpha_A \eta^{\mathcal{L}}_A$ because of Diagram (2.56).

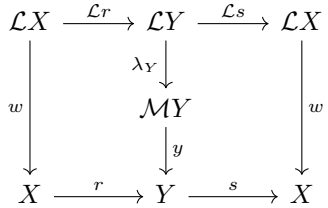
Finally, we will need the following commuting diagram, that comes from the definition of h_A (Equation (2.51)) and the fact that ℓ is a map of $\mathcal{L}$-algebra (Equation (2.47)).



(2.58)

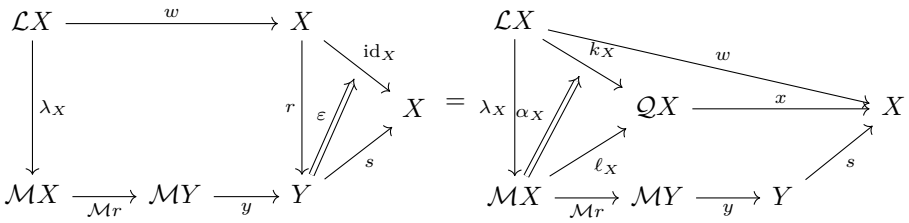
2.3 A technical lemma

Lemma 31. *Assume that* $\mathcal{L}X \xrightarrow{w} X$ *is an* $\mathcal{L}$*-algebra, and that* $\mathcal{M}Y \xrightarrow{y} Y$ *is an* $\mathcal{M}$*-algebra, and that there is a splitting* $Y \underset{s}{\overset{r}{\leftrightarrows}} X$ *with unit* $\eta : \mathrm{id}_Y = rs$ *and counit* $\varepsilon : sr \Rightarrow \mathrm{id}_X$ *such that*



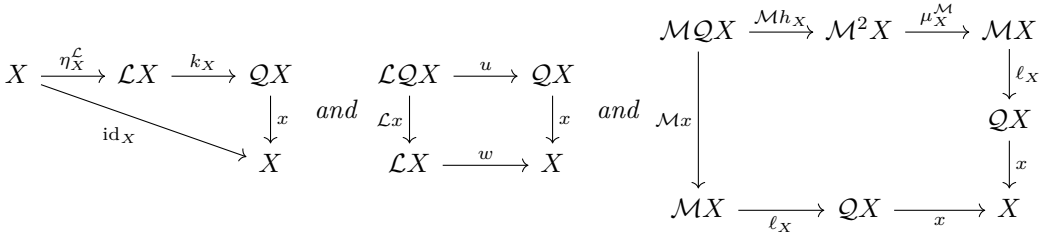
(2.59)

Then, the 1*-cell* $\mathcal{Q}X \xrightarrow{x} X$ *, obtained by universality on the cone*



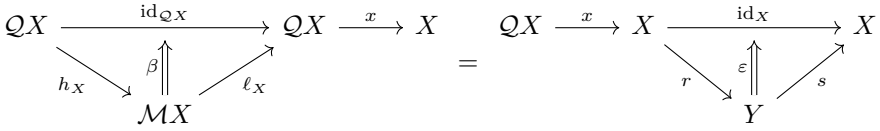
(2.60)

satisfies the 1*-cell equalities:*



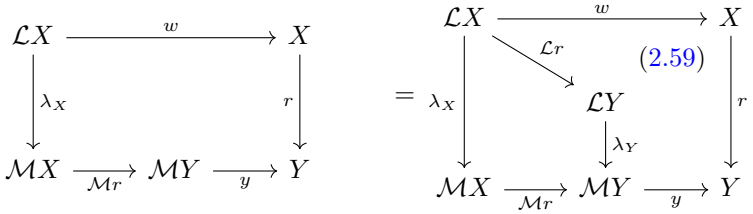
(2.61)

and the 2*-cell equality:*



(2.62)

Proof. First notice that, by naturality of λ, Diagram (2.59) implies the commutation of:



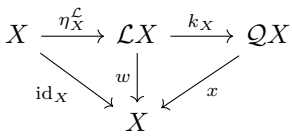
(2.63)

The 2-cell in Diagram (2.62) are the same, by 2-cell universality, as they satisfy the same equations:

$$x \cdot \beta \cdot \ell = x \cdot \alpha = \varepsilon \cdot x \cdot \ell$$
$$x \cdot \beta \cdot k = x \cdot \mathrm{id}_k = \varepsilon \cdot x \cdot k$$

that come from the respective definition of β and x.

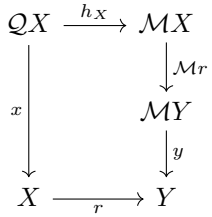
The left-most diagram of (2.61) commutes because w is an $\mathcal{L}$-algebra and by definition of x (2.60):



(2.64)

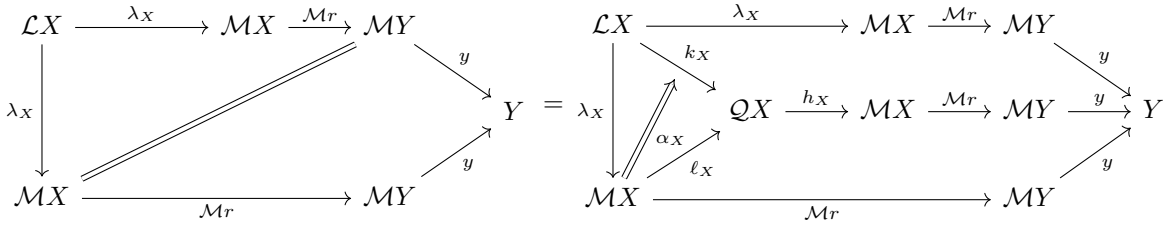
The middle diagram of (2.61) commutes because x is a map of $\mathcal{L}$-algebra.

For the right-most diagram of (2.61), we will need the commutation of Diagram:



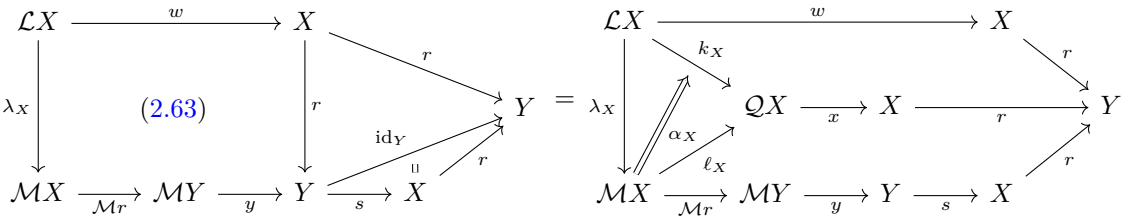
(2.65)

It comes from 2-cell universality, as the right-then-down map comes from the identity 2-cell:



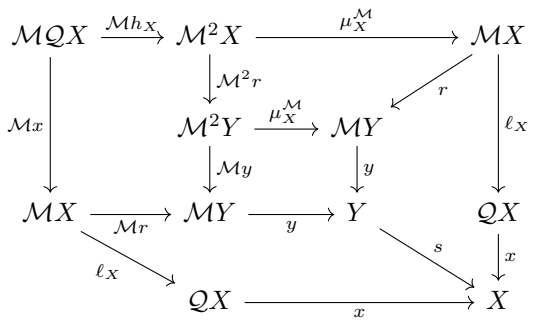
(2.66)

and the down-then-right map comes from the identity 2-cell:



(2.67)

Finally, we apply $\mathcal{M}$ to Diagram (2.65) and use naturality of $\mu^{\mathcal{M}}_X$, the fact that (Y, y) is an $\mathcal{M}$-algebra and the definition of x (2.60) as $x\ell_X = sy\mathcal{M}r$:



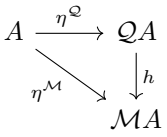
(2.68)

□

2.4 The colimit is a monad

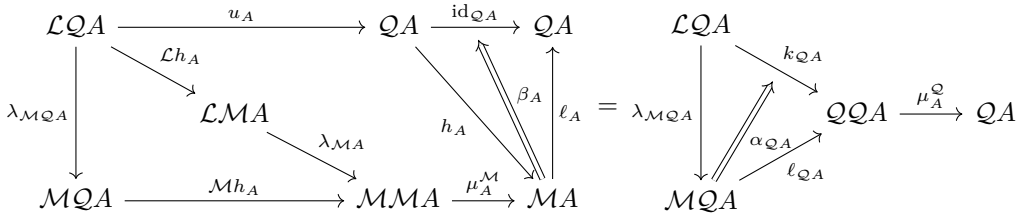
We will use the technical Lemma 31 to prove that the colimit $\mathcal{Q}$ is also a monad. Let us first define the *unit* $\eta^{\mathcal{Q}}$ and the *multiplication* $\mu^{\mathcal{Q}}$ for $\mathcal{Q}$

Unit. We set $A \xrightarrow{\eta^{\mathcal{L}}} \mathcal{L}A \xrightarrow{k_A} \mathcal{Q}A$. Notice that, thanks to definition of h, see Diagramm (2.50),



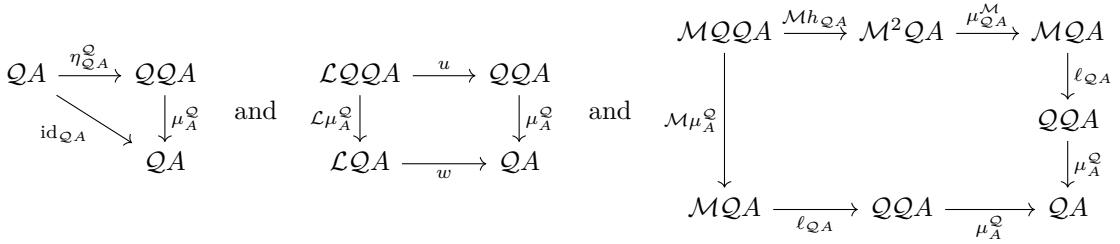
(2.69)

Multiplication. By Paragraph §2.2.2 and Diagram 2.58, we can apply Lemma 31 to $X = \mathcal{Q}A$ and $Y = \mathcal{M}A$, $r_{\mathcal{Q}A} = h_A$, $s_{\mathcal{Q}A} = \ell_A$ and $\varepsilon_{\mathcal{Q}A} = \beta_A$. Then, $\mathcal{Q}^2A \xrightarrow{\mu^{\mathcal{Q}}_A} \mathcal{Q}A$ is unique such that the two following cones are equal:



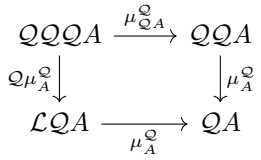
(2.70)

Monad laws. They stem from Diagram (2.61) which can be rewritten as:



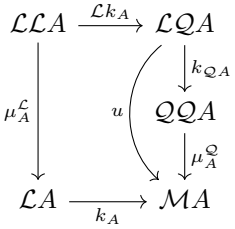
(2.71)

The two right-most-diagrams, give by 1-cell and 2-cell universality, the following commutative diagram:



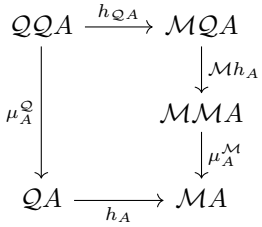
(2.72)

Map of monads. Notice that k is a map of monad, indeed it comes from the definition of $\mu^{\mathcal{Q}}$ and the fact that u is a map of $\mathcal{L}$ algebra:



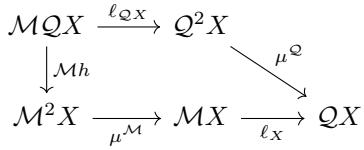
(2.73)

Notice that h is also a map of monad. As a by-product of Lemma 31, we get Diagram (2.65), which can be rewritten as:



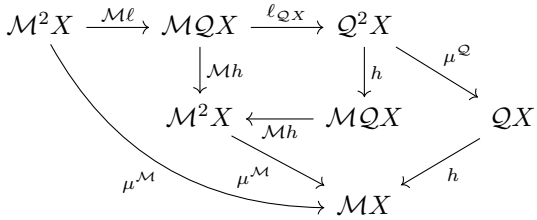
(2.74)

In general, $\mathcal{M} \xrightarrow{\ell} \mathcal{Q}$ is not a map of monad. However, we have the following commuting diagram, which is coming from the definition of the multiplication in Diagram (2.70).



(2.75)

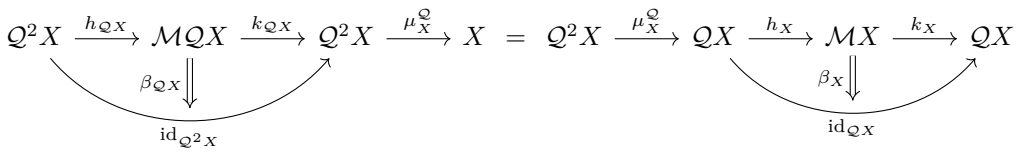
We will also use the following commuting diagram which commutes by definition of h, see Diagram (2.50), and by Diagram (2.74)



(2.76)

We can sum up in the following these results.

Theorem 32. *$\mathcal{Q}$ is a monad, $\mathcal{L} \xrightarrow{k} \mathcal{Q}$ and $\mathcal{Q} \xrightarrow{h} \mathcal{M}$ are map of monads and $\mu^{\mathcal{Q}}_X$ is a strict map of comonad from the comonad $\mathcal{Q}^2X \xrightarrow{\ell_{\mathcal{Q}X} h_{\mathcal{Q}X}} \mathcal{Q}^2X$ to the comonad $\mathcal{Q}X \xrightarrow{\ell_X h_X} \mathcal{Q}X$, that is:*

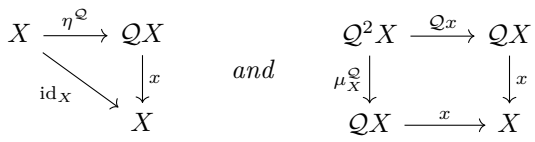


(2.77)

2.5 A characterization of $\mathcal{Q}$-algebras.

We want to describe the algebras of the colimit monad, Lemma 31 gives us a sufficient condition:

Proposition 33. *Under the hypothesis of Lemma 31, x is a $\mathcal{Q}$-algebra, that is:*



(2.78)

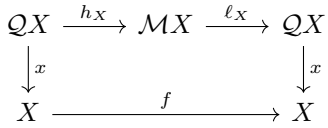
Proof. The left-most Diagram of (2.61) gives the unit property. The two right-most Diagrams of (2.61) define two cones which are equal so that by 2-cell universality, we get the multiplication property. □

We want to prove the reverse and check that a $\mathcal{Q}$-algebra satisfies properties of Lemma 31 !

Let X be a $\mathcal{Q}$-algebra: $\mathcal{Q}X \xrightarrow{x} X$. Then, $\mathcal{L}X \xrightarrow{k_X} \mathcal{Q}X \xrightarrow{x} X$ is the structure of $\mathcal{L}$-algebra over X.

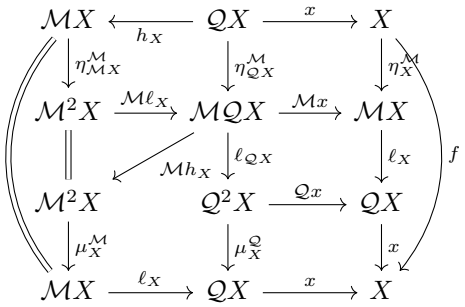
An idempotent comonad. We define a map of $\mathcal{L}$-algebra $X \xrightarrow{f} X$ as $f : X \xrightarrow{\eta^{\mathcal{M}}} \mathcal{M}X \xrightarrow{\ell_X} \mathcal{Q}X \xrightarrow{x}$.

We will need the following square:



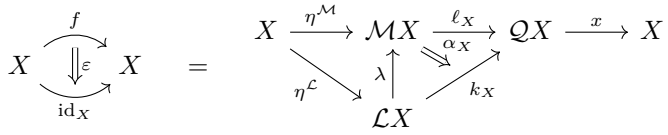
(2.79)

that results from the definition of the Retract h_X in Diagram (2.50), the definition of the multiplication of $\mathcal{Q}$ in Diagram (2.75) and the naturality of ℓ and $\eta^{\mathcal{M}}$, and the fact that x is a $\mathcal{Q}$-algebra.



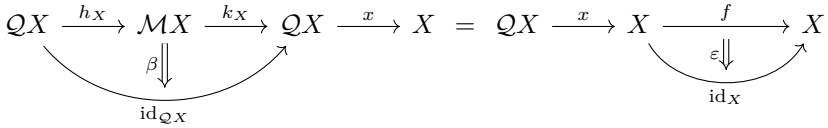
(2.80)

By composing the Square (2.79) by $X \xrightarrow{\eta^{\mathcal{M}}_X} \mathcal{M}X \xrightarrow{\ell_X} \mathcal{Q}X$ and because $h_X \ell_x = \mathrm{id}_{\mathcal{M}X}$, we get $f^2 = f$. Notice that $x\ell_X\eta^{\mathcal{L}} = x\eta^{\mathcal{Q}} = \mathrm{id}_X$, so that we can define the counit as $\varepsilon : x\alpha\eta^{\mathcal{M}}$



(2.81)

A map of idempotent comonad. To prove that x is a map of idempotent comonad, we have to prove at the 2-cell level, that $x\beta_X = \varepsilon x$,

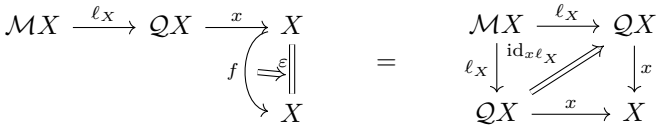


(2.82)

By 2-cell universality, it boils down to proving that $x\alpha_X = x\beta_X k_X = \varepsilon x k_X$ and $\mathrm{id}_{x\ell_X} = x\beta_X \ell_X = \varepsilon x \ell_X$. Remind from Square (2.75) that $fx = xk_X h_X$ and from definition of h_X in Diagram (2.50) that $h_X k_X = \lambda$ and $h_X \ell_X = \mathrm{id}_{\mathcal{M}X}$, so that $fxk_X = x\ell_X\lambda$ and $fx\ell_X = x\ell_X$. We deduce that $x\alpha_X = \varepsilon x \ell_X$:

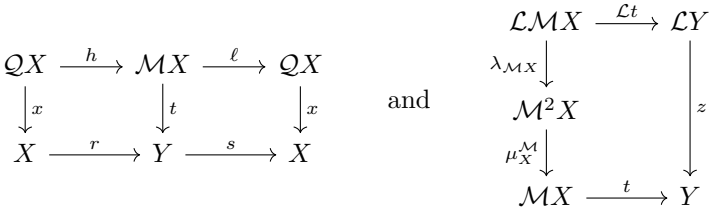
(2.83)

and that $\mathrm{id}_{x\ell_X} = \varepsilon x k_X$:



(2.84)

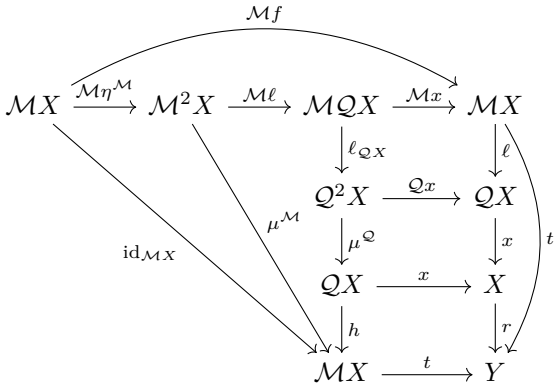
The splitting induced by the comonad f and the map of splitting induced by x. By Proposition 29, since f is a (strict) idempotent comonad, there is a stricit splitting $X \xrightarrow{f} X = X \xrightarrow{r} Y \xrightarrow{s} X$ in $\mathcal{L}$-algebras such that $rs = \mathrm{id}_Y$. Moreover by Proposition 30, since x is a map of idempotent comonad, it is also a map of splitting in $\mathcal{L}$-algebras. Thus, there is a map $\mathcal{M}X \xrightarrow{t} Y$ of $\mathcal{L}$-algebras such that:



(2.85)

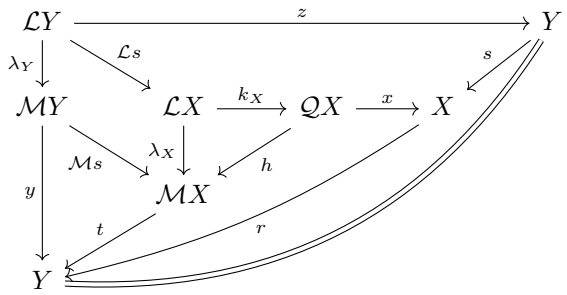
Notice that $\mathcal{M}X \xrightarrow{t} Y = \mathcal{M}X \xrightarrow{\ell} \mathcal{Q}X \xrightarrow{x} X \xrightarrow{r} Y$. Indeed, $rs) \mathrm{id}_Y$, thus $t = rst = rx\ell_X$.

Moreover, we have $\mathcal{M}X \xrightarrow{\mathcal{M}f} \mathcal{M}X \xrightarrow{t} Y = \mathcal{M}X \xrightarrow{t} Y$. Indeed, the following diagram commutes by naturality of ℓ, because x is a $\mathcal{Q}$ algebra and by using Diagrams (2.85) and (2.76)



(2.86)

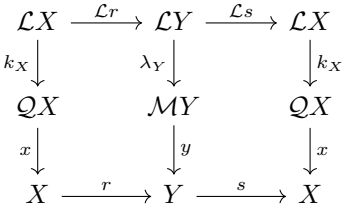
An $\mathcal{M}$-algebra. We define $\mathcal{M}Y \xrightarrow{y} Y = \mathcal{M}Y \xrightarrow{\mathcal{M}s} \mathcal{M}X \xrightarrow{t} Y$. It is by definition a map of $\mathcal{L}$-algebra. Moreover, the structure of $\mathcal{L}$-algebra over Y is:



(2.87)

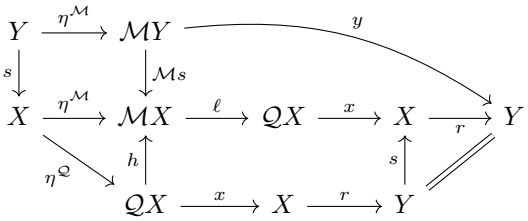
by naturality of s and λ, by definitions of h (see Diagram (2.50)), of y, of the splitting r and s and by Diagram (2.85).

The following diagram commutes because r and s are maps of $\mathcal{L}$-algebras:



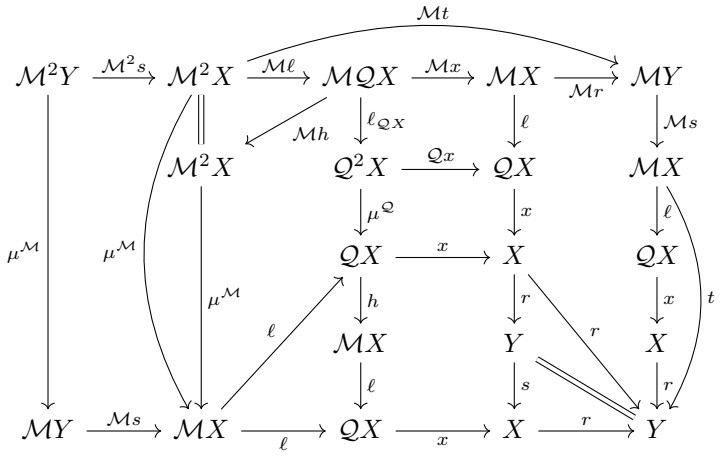
(2.88)

Let us prove that $\mathcal{M}Y \xrightarrow{y} Y$ is an $\mathcal{M}$-algebra. First the unit law results from naturality of $\eta^{\mathcal{M}}$, from Diagrams (2.69) and (2.85) and by definition of y and of the splitting s and r.



(2.89)

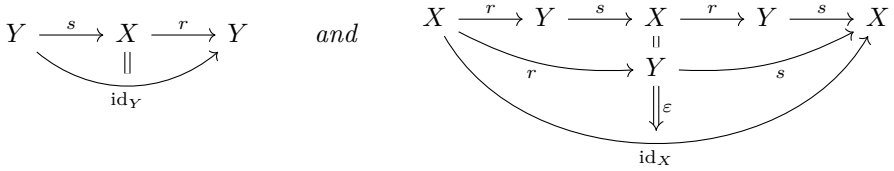
The multiplication law results from naturality of $\mu^{\mathcal{M}}$ and k, by Definition of h see Diagram (2.50), because x is a $\mathcal{Q}$-algebra and by Diagram (2.86):



(2.90)

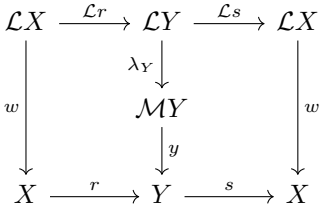
We can sum up these results as:

Proposition 34. *Let $\mathcal{Q}X \xrightarrow{x} X$ be a $\mathcal{Q}$-algebra, then $\mathcal{L}X \xrightarrow{w} X = \mathcal{L}X \xrightarrow{k_X} \mathcal{Q}X \xrightarrow{x} X$ is an $\mathcal{L}$-algebra, there are an $\mathcal{M}$-algebra $\mathcal{M}Y \xrightarrow{y} Y$ and a splitting $X \xrightarrow{r} Y \xrightarrow{s} X$ such that:*



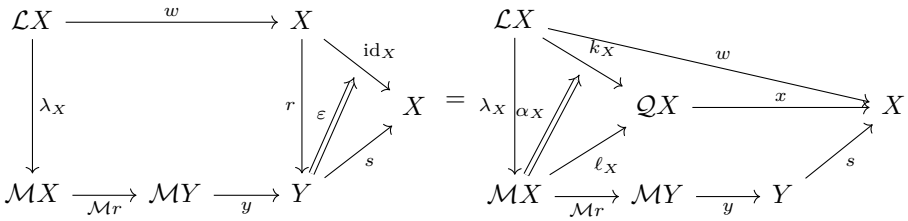
(2.91)

and



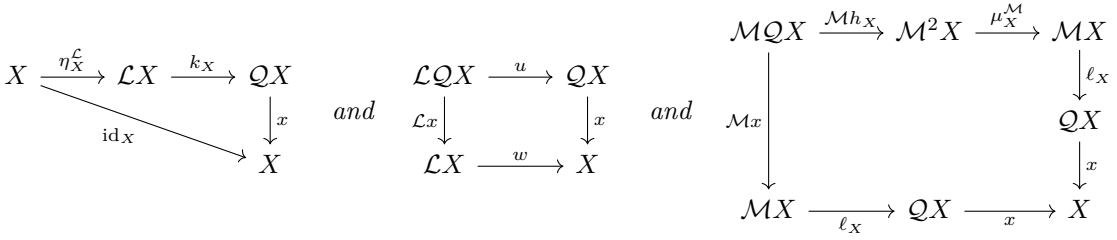
(2.92)

then, $\mathcal{Q}X \xrightarrow{x} X$ *satisfies the* 1*-cell universality on the cone*



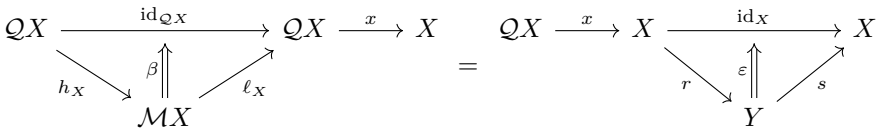
(2.93)

and it satisfies the 1*-cell equalities:*



(2.94)

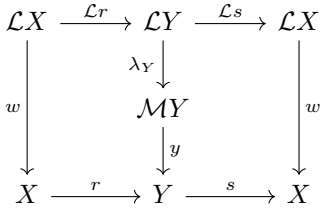
and the 2*-cell equality:*



(2.95)

This gives us a characterization of $\mathcal{Q}$-algebras:

Theorem 35. *A $\mathcal{Q}$-algebra is the data of an $\mathcal{L}$-algebra $\mathcal{L}X \xrightarrow{w} X$, a splitting in $\mathcal{L}$-algebras $X \xrightarrow{r} Y \xrightarrow{s} X$ which is an idempotent comonad, and an $\mathcal{M}$-algebra $\mathcal{M}Y \xrightarrow{y} Y$ such that we have the factorization:*



(2.96)

2.6 Example

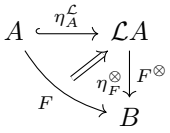
We develop our colimit construction on the 2-monad $\mathcal{L}$ on the 2-category **CAT** for symmetric monoidal categories (see Example 36) and in the 2-monad $\mathcal{M}$ for finite product categories (see Example 37). Their colimit $\mathcal{Q}$ is presented in Example 38. As mentioned in the introduction of the Chapter, we have in mind the application of this construction to model of differential λ-calculus. If we can prove that the monad $\mathcal{Q}$ lifts from **CAT** to profunctors, then the bikleisli category will be a good candidate for a model of the linear and the non-linear substitution appearing in differential λ-calculus.

Example 36 (The 2-monad for symmetric monoidal categories)**.** For a category A, we denote by $\mathcal{L}A$ the category whose objects are finite sequences $\langle a_i \rangle_{i\in[n]}$, with $n \in \mathbb{N}$ and $a_i \in A$, whose morphisms $\langle a_i \rangle_{i\in[n]} \to \langle a'_j \rangle_{j\in[n]}$ consist of a pair of a bijection σ of $[n]$, and a sequence $\langle f_i \rangle_{i\in[n]}$ of morphisms $f_i \in A(a_i, a'_{\sigma(i)})$. Notice that $\mathcal{L}A$ is the free symmetric strict monoidal category on A with unit given by the empty sequence $\langle \_ \rangle$ and tensor product given by concatenation $\langle a_i \rangle_{i\in[n]} \oplus \langle a'_j \rangle_{j\in[m]} \triangleq \langle a_1, \ldots, a_n, a'_1, \ldots, a'_m \rangle$.

We have just described the action on objects of the 2-monad $\mathcal{L} : \mathbf{CAT} \to \mathbf{CAT}$. We describe the structure of the 2-monad. The unit $\eta^{\mathcal{L}} : Id \to \mathcal{L}$ has components given by embeddings $\eta^{\mathcal{L}}_A : A \hookrightarrow \mathcal{L}A$ and the multiplication $\mu^{\mathcal{L}} : \mathcal{L}^2 \to \mathcal{L}$ has components given by the tensor product on $\mathcal{L}A$:

$$\eta^{\mathcal{L}}_A(a) \triangleq \langle a \rangle \qquad \mu^{\mathcal{L}}_A(\langle u_i \rangle_{i\in[n]}) \triangleq \oplus_{i\in[n]} u_i$$

The universal property of $\mathcal{L}$ requires that for any functor $F : A \to B$, where B is a symmetric monoidal category, we have a diagram:



In particular, the free symmetric monoidal category $\mathbf{B}^{\mathrm{op}} \triangleq \mathcal{L}\mathbf{1}$ over $\mathbf{1}$ is the category of natural numbers whose morphisms $\mathbf{B}^{\mathrm{op}}(u, v) = \mathfrak{S}_n$ whenever $u = v = [n]$. Notice that $\mathbf{B}^{\mathrm{op}}$ encapsulates the basic operation for ruling contexts with linear variable: the exchange rule (see Tanaka [2000])

The 2-category $\mathbf{Alg}(\mathcal{L})$ is made of strict algebras of $\mathcal{L}$, i.e. symmetric strict monoidal categories, strict morphims and strict transformations.

Example 37 (The 2-monad for cartesian categories)**.** Let us now describe $\mathcal{M}A$, the free cartesian categories over A. The object are the same as $\mathcal{L}A$, but the morphisms differ: a morphism $\langle b_i \rangle_{i\in[n]} \to \langle b'_j \rangle_{j\in[m]}$ consists of a pair of a function $\rho : [m] \to [n]$ and for each $i \in [m]$, a morphism $f_i \in A(b_{\rho(i)}, b'_i)$. The terminal object is the empty sequence and the product of two sequences is given by concatenation.

In particular, $\mathbf{F}^{\mathrm{op}}$ the free cartesian category on $\mathbf{1}$ has *natural numbers* as objects, finite products are given by *addition* of natural numbers, so that a morphism $\alpha \in \mathcal{M}\mathbf{1}(k, m)$ is given by, for each $i \in [m]$, a choice of projection $k \to 1$. Hence, α is defined by a function $[m] \to [k]$. Notice that, from the variable ruling view, $\mathbf{F}^{\mathrm{op}}$ encapsulates non linear operations such as forgetting and repeating. For each m in $\mathbf{F}^{\mathrm{op}}$, there is: the ith product projection $m \to 1$ corresponding to forgetting all m variables except the ith one ; the diagonal $1 \to m$ corresponding to repeating a variable m times (see Fiore et al. [1999]).

Actually, $\mathcal{M}$ is a 2-monad on $\mathbf{CAT}$ whose unit $\eta^{\mathcal{M}}$ and multiplication $\mu^{\mathcal{M}}$ are defined similarly as the ones of $\mathcal{L}$.

$$\eta^{\mathcal{M}}_A(b) \triangleq \langle b \rangle \quad \text{and} \quad \mu^{\mathcal{M}}_A(\langle v_i \rangle_{i\in[n]}) \triangleq \oplus_{i\in[n]} v_i$$

It enjoys also an extension property of any functor $F : A \to B$ with B cartesian to a cartesian functor $F^{\times}$.

The 2-category $\mathbf{Alg}(\mathcal{M})$ is made of strict algebras of $\mathcal{M}$, i.e. cartesian categories, cartesian morphims and strict transformations.

Example 38 (The 2-monad for linear-non-linear categories)**.** Let us now describe $\mathcal{Q}$, the 2-monad obtained by our colimit construction. The objects are finite sequences $\langle a_1, \ldots, a_n, \underline{b}_1, \ldots, \underline{b}_p \rangle$ of two kinds of objects of A. Intuitively, the a_is are linear (they behave as in $\mathcal{L}A$) and the underlined b_i are not linear (they behave as in $\mathcal{M}A$). The morphisms are

$$\langle a_1, \ldots, a_n, \underline{b}_1, \ldots, \underline{b}_p \rangle \xrightarrow{\tau:[m+q]\to[n+p], f_i : a_{\rho(i)} \to a'_i} \langle a'_1, \ldots, a'_m, \underline{b}'_1, \ldots, \underline{b}'_q \rangle$$

where τ is a function whose restriction to $[m]$ is an injection. This means that a morphism can forget that an a is linear and transform it into a non linear b.

We have a map $h_A : \mathcal{M}A \to \mathcal{Q}A$ which transform linear as to the same categories $\underline{a}$s, but considered as non linear, $h_A : \langle a_1, \ldots, a_n, \underline{b}_1, \ldots, \underline{b}_p \rangle \mapsto \langle \underline{a}_1, \ldots, \underline{a}_n, \underline{b}_1, \ldots, \underline{b}_p \rangle$.

Let us now describe the structure of monad:

$$\eta^{\mathcal{Q}}_A(a) \triangleq \langle a \rangle \quad \text{and} \quad \mu^{\mathcal{Q}}_A : \langle u_1, \ldots, u_n, \underline{v}_1, \ldots, \underline{v}_p \rangle \mapsto u_1 \oplus \cdots \oplus u_n \oplus h_A(\underline{v}_1) \oplus \cdots \oplus h_A(\underline{v}_p)$$

Finally, thanks to the characterization Theorem 35, the 2-category $\mathbf{Alg}(\mathcal{Q})$ is made of a symmetric monoidal category X, together with a cartesian category Y and a splitting $Y \underset{s}{\overset{r}{\underset{\longrightarrow}{\overset{\longleftarrow}{\top}}}} X$ such that the monoidal structure of Y is the same as its cartesian structure.

2.7 Bibliography

J. C. Baez and J. Dolan. Higher-dimensional algebra and topological quantum field theory. *J. Math. Phys.*, 36(11), 1995. 34

J. Bénabou. Introduction to bicategories. In *Reports of the Midwest Category Seminar*, volume 47, pages 1–77. Springer, 1967. 37

R. Blute, T. Ehrhard, and C. Tasson. A Convenient Differential Category. *Cah. Topologies géom. diff.*, 2012. 32

RF Blute, JRB Cockett, and RAG Seely. Differential categories. *Mathematical structures in computer science*, 16(06):1049–1083, 2006. 34

Richard Blute, Thomas Ehrhard, and Christine Tasson. A convenient differential category. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 53(3):211–232, 2012. 35

Francis Borceux. *Handbook of Categorical Algebra 2 – Categories and Structures.* Cambridge Univ. Press, 1994. 43

C. Chauve, É. Fusy, and J. Lumbroso. An exact enumeration of distance-hereditary graphs. In *ANALCO 2017*, pages 31–45. 33

E. Cheng. Distributive laws for Lawvere theories. *ArXiv e-prints*, December 2011. 36

T. Ehrhard. Finiteness spaces. *Math. Struct. Comput. Sci.*, 15(4), 2005. 32

T. Ehrhard. A finiteness structure on resource terms. In *LICS*, pages 402–410. IEEE Computer Society, 2010. 32, 35

T. Ehrhard. An introduction to differential linear logic: proof-nets, models and antiderivatives. *CoRR*, abs/1606.01642, 2016. 34

T. Ehrhard and L. Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1-3): 1–41, 2003. 32, 33, 34

T. Ehrhard and L. Regnier. Böhm trees, Krivine's Machine and the Taylor Expansion of Lambda-Terms. In *CiE*, 2006a. 32

T. Ehrhard and L. Regnier. Differential interaction nets. *Theoretical Computer Science*, 364(2):166–195, 2006b. 33

T. Ehrhard and L. Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theor. Comput. Sci.*, 403(2-3), 2008. 32, 33

Thomas Ehrhard, Michele Pagani, and Christine Tasson. Probabilistic coherence spaces are fully abstract for probabilistic PCF. In *POPL*, pages 309–320. ACM, 2014. 32

Thomas Ehrhard, Michele Pagani, and Christine Tasson. Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming. In *POPL*, pages 59:1–59:28. ACM, 2018. 32

M. Fiore. On the structure of substitution. Invited address for MFPSXXII, 2006. 36

M. Fiore, G. Plotkin, and D. Turi. Abstract syntax and variable binding. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*, page 193, 1999. 34, 35, 55

M. Fiore, N. Gambino, M. Hyland, and G. Winskel. The cartesian closed bicategory of generalised species of structures. *J. London Math. Soc.*, 77(1), 2008. 34

M. Fiore, N. Gambino, M. Hyland, and G. Winskel. Relative pseudomonads, Kleisli bicategories, and substitution monoidal structures. *ArXiv e-prints*, 2016. 35, 36

M.P. Fiore. Differential structure in models of multiplicative biadditive intuitionistic linear logic. *Lecture Notes in Computer Science*, 4583:163, 2007. 34

P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009. ISBN 978-0-521-89806-5. URL http://www.cambridge.org/uk/catalogue/catalogue.asp?isbn=9780521898065. 33

J.-Y. Girard. Normal functors, power series and lambda-calculus. *Annals of Pure and Applied Logic*, 37 (2), 1988. 32

Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987. 32

Ryu Hasegawa. Two applications of analytic functors. *Theor. Comput. Sci.*, 272(1-2):113–175, 2002. 33

A. Hirschowitz and M. Maggesi. Modules over monads and linearity. *Lecture Notes in Computer Science*, 4576:218, 2007. 35

W.A. Howard. The formulae-as-types notion of construction. In Jonathan P. Seldin and J. Roger Hindley, editors, *Essays on Combinatory Logic, Lambda Calculus, and Formalism*, volume to H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, pages 479–490. Academic Press, 1980. 32

M. Hyland. Classical lambda calculus in modern dress. *Math. Struct. Comput. Sci.*, 27(5), 2017. 34, 36

M. Hyland, G. Plotkin, and J. Power. Combining effects: Sum and tensor. *Theoretical Computer Science*, 357(1-3):70–99, 2006. 36

Martin Hyland. Elements of a theory of algebraic theories. *Theor. Comput. Sci.*, 546:132–144, 2014. 35, 36

P. Hyvernat. A linear category of polynomial functors (extensional part). *Log. Meth. Comput. Sci.*, 10 (2), 2014. 33

A. Jacquot. *Constructions par greffe, combinatoire analytique et génération analytique. (Graft reconstruction, analytic combinatorics and analytical generation)*. PhD thesis, Paris 13 University, Villetaneuse, Saint-Denis, Bobigny, France, 2014. 33

A. Joyal. Une théorie combinatoire des séries formelles. *Adv. Math.*, 42(1):1 – 82, 1981. 33

M. Kerjean and C. Tasson. Mackey-complete spaces and power series–a topological model of differential linear logic. *Math. Struct. Comput. Sci.*, 2016. 32

Marie Kerjean and Christine Tasson. Mackey-complete spaces and power series - a topological model of differential linear logic. *Mathematical Structures in Computer Science*, 28(4):472–507, 2018. 36

A. Kriegl and P.W. Michor. *The convenient setting of global analysis*, volume 53. American Mathematical Society, 1997. 35

Stephen Lack and Ross Street. The formal theory of monads ii. *Journal of Pure and Applied Algebra*, 175(1):243 – 265, 2002. Special Volume celebrating the 70th birthday of Professor Max Kelly. 37

J. Laird, G. Manzonetto, G. McCusker, and M. Pagani. Weighted relational models of typed lambda-calculi. In *LICS*, 2013. 32

T. Leinster. *Higher Operads, Higher Categories*. August 2004. 37, 41

J.-L. Loday and B. Vallette. *Algebraic operads*, volume 346 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. 2012. 33

P.-A. Melliès. Categorical semantics of linear logic. 2009. 33

M. Pagani, P. Selinger, and B. Valiron. Applying quantitative semantics to higher-order quantum computing. In *ACM SIGPLAN Notices*, volume 49. ACM, 2014. 32

Michele Pagani, Christine Tasson, and Lionel Vaux. Strong normalizability as a finiteness structure via the taylor expansion of λ-terms. In *FoSSaCS*, volume 9634 of *Lecture Notes in Computer Science*, pages 408–423, 2016. 32, 35

John Power and Miki Tanaka. Binding signatures for generic contexts. In *TLCA*, volume 3461 of *Lecture Notes in Computer Science*, pages 308–323. Springer, 2005. 34, 35, 36

R. Street. The formal theory of monads. *J. Pure Appl. Algebra*, 2(2):149–168, 1972. 36, 37

Ross Street. Limits indexed by category-valued 2-functors. *Journal of Pure and Applied Algebra*, 8(2): 149 – 181, 1976. 39

Miki Tanaka. Abstract syntax and variable binding for linear binders. In *Mathematical Foundations of Computer Science 2000, 25th International Symposium, MFCS 2000, Bratislava, Slovakia, August 28 - September 1, 2000, Proceedings*, pages 670–679, 2000. 34, 55

Christine Tasson and Lionel Vaux. Transport of finiteness structures and applications. *Mathematical Structures in Computer Science*, 28(7):1061–1096, 2018. 35

T. Tsukada, K. Asada, and L. Ong. Generalised species of rigid resource terms. In *LICS 2017*. 34

B. Vallette. Homology of generalized partition posets. *J. Pure Appl. Algebra*, 208(2):699–725, 2007. 33

Chapter 3

Probabilistic Semantics

Publications of the author

Thomas Ehrhard, Michele Pagani, and Christine Tasson. The computational meaning of probabilistic coherence spaces. In *LICS*, pages 87–96. IEEE Computer Society, 2011.

Thomas Ehrhard, Michele Pagani, and Christine Tasson. Probabilistic coherence spaces are fully abstract for probabilistic PCF. In *POPL*, pages 309–320. ACM, 2014.

Thomas Ehrhard, Michele Pagani, and Christine Tasson. Full abstraction for probabilistic PCF. *Journal of the ACM*, 65(4):23:1–23:44, 2018a.

Thomas Ehrhard and Christine Tasson. Probabilistic call by push value. *Logical Methods in Computer Science*, 2018. Accepted for publication, LMCS.

Thomas Ehrhard, Michele Pagani, and Christine Tasson. Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming. In *POPL*, pages 59:1–59:28. ACM, 2018b.

Contents

Introduction

This Chapter 3 is devoted to the denotational semantics of probabilistic programming.

We started this piece of work with Thomas Ehrhard and Michele Pagani with the think of taking benefit of the smoothness of the interpretation of programs in Linear Logic models. Actually, this plan was already hinted at the very beginning of Linear Logic. It was mentioned in the appendix of the seminal paper Girard [1988] and further developed in Girard [2004] which introduced Probabilistic Coherent Spaces. Those spaces were studied in full details in Danos and Ehrhard [2011]. They are shown to be a model of Linear Logic and of probabilistic PCF, a call-by-name[1] probabilistic programming language. One important particularity of this model is that program interpretation can be seen as power series. The reduction is defined through a stochastic matrix Red indexed by terms. The meaning of the coefficient $\mathsf{Red}_{M,M'}$ is the probability that M reduces to M' in one step. We are first interested in the invariance of the interpretation with respect to the reduction:

Soundness: For every term M, $[\![M]\!] = \sum_{M'} \mathsf{Red}_{M,M'}[\![M']\!]$.

After, we are interested in the adequacy lemma: if the type of M is natural numbers, then for any natural number $n \in \mathbb{N}$, $\mathsf{Red}^{\infty}_{M,\underline{n}} = [\![M]\!]_n$. It implies that two terms with the same semantics cannot be distinguished from their behavior in any evaluation context. We say that they are observationally equivalent:

Adequacy: Let M and M' be two closed terms. If $[\![M]\!] = [\![M']\!]$ then $M \sim M'$.

The last theorem we are interested in is Full-Abstraction. It boils down to the converse of Adequacy.

Full Abstraction: Two programs have the same semantics if and only if they behave in the same way in whatever context they are evaluated.

Soundness and Adequacy were proved in Danos and Ehrhard [2011] for probabilistic PCF. Full Abstraction is well-known to be a hard to prove property. Besides, the demonstration (we gave in Ehrhard et al. [2014]) is different from the usual ones which rely on the definabilty of points of the semantics. Our proof relies on a powerful and original tool: namely, programs are interpreted as power series with non-negative coefficients. Let us describe the sketch of proof. We reason by contrapositive. Given two terms with different semantics, we define a context that separates them. This context, when composed with the terms are interpreted as two power series with different coefficients. An argument from real analysis and Adequacy lead to separate the two terms.

Yet, in probabilistic PCF, there is

An expressivity issue: It might be hard to encode probabilistic algorithm. For instance, consider this simple Las Vegas algorithm on a finite array with 0 and 1 cells,

1. randomly choose an index k;
2. test if the content of the cell of index k is 0;
3. if yes, output the index k;
4. if no, start again from step 1.

In a call-by-name language, variables are computed each time they appear, potentially leading to different outputs in a probabilistic setting. Yet, we need to memorize the value of the index k (step 1.) in order to output the same value later (step 3.). So that, we need to have a call-by-value evaluation on ground-types. Natural numbers and other value types are interpreted as coalgebras (introduced in §3.1.4) which is the key notion that is compatible with call-by-value. The language we introduce next incorporate this semantic property.

We first extended in Ehrhard et al. [2018a], the probabilistic PCF with a `let` construction for the ground-type of natural numbers. The resulting language, probabilistic PCF, is denoted as pPCF (see Section 3.3), for which we proved Soundness, Adequacy and Full-Abstraction theorems.

In order to generalize this `let` construct to other types such as `Lists` or `Streams`, we studied in Ehrhard and Tasson [2018], a probabilistic extension of call-by-push-value, where all values enjoy this `let` construction. The resulting language is denoted as $\Lambda^{\mathsf{P}}_{\mathsf{HP}}$ (see Section 3.4). We proved Soundness, Adequacy and Full-Abstraction theorems. The Adequacy Theorem 85 for $\Lambda^{\mathsf{P}}_{\mathsf{HP}}$ is more involved than the one for pPCF, because of recursive types. We adapted the traditional proof that uses logical relations with a method inspired by Pitts [1994] in order to compute fixpoint of logical relations. The Full-Abstraction Theorem 88 needs to introduce new tools based on coalgebras.

1. In the call-by-name evaluation strategy, the argument expression is first passed to the function that will potentially copy the expression, before being evaluated each time it occurs

Notice that pPCF can be encoded into $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$ (as we show §3.4.2). Thus, the adequacy result for pPCF is implied by the one for $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$. Indeed, all contexts for the first language are contexts for the second one. However, the Full-Abstraction results are different because there are more contexts in $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$ than in pPCF.

The first language that we will present in this chapter is pure λ-calculus, denoted as Λ^{+} (see Section 3.2). This allows us to introduce gently the technique we used for proving the Adequacy Theorem for recursive types in $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$.

Finally, we turned recently towards the continuous probability settings in Ehrhard et al. [2018b]. Thomas Ehrhard has built a model of a continuous probabilistic programming language, generalizing the semantics developed for the discrete setting. Although the interpretations of programs are not yet proved to be power series, noteworthy, they enjoy a strong form of monotonicity that we call stability. Indeed, it is deeply related with the standard notion of stability designed for sequential algorithms in Berry [1978]. We introduced pRPCF, a version of pPCF with two main differences. First, natural numbers and successor are replaced by reals and measurable functions. Second, the probabilistic binary choice is replaced by a sampler uniformly choosing a real in the $[0, 1]$ interval. We defined the operational semantics by replacing stochastic matrices by stochastic kernels. Finally, we proved that this model is sound, adequate in §3.5.5 and well-designed to interpret the most standard probabilistic algorithms in §3.5.6.

Context

Since the 80's, formal methods have been applied to probabilistic programming languages. For instance, Kozen [1979] defined a denotational semantics for a first-order while-language extended with a real random number generator. This topic has then been seized by the domain theoretic community with the monadic approach due to Moggi [1989]. This standard approach is based on the use of powerdomains Plotkin [1976]. In Jones and Plotkin [1989], types are interpreted as domains on which acts a probabilistic powerdomain monad $\mathcal{V}$. A probabilistic program of type $A \to B$ is interpreted as a continuous function $[A] \to \mathcal{V}([B])$ which maps a value of $[A]$ to a probability distribution over values of $[B]$. In this way, Jones got computational adequacy results in the typed and untyped case for call-by-value[2] strategy.

To explicit the differences between the domain theoretic approach and our approach based on quantitative semantics, let us take the example of a first-order program M taking integers as inputs and intergers as outputs, that is with type $M : \mathcal{N} \Rightarrow \mathcal{N}$.

In the domain theoretic approach, the type $\mathcal{N}$ is interpreted by the flat domain $\mathbb{N}_{\perp}$ of unrelated natural numbers together with a lowest element $\perp$ representing divergence. The probabilistic monad gives rise to the probabilistic distributions over $\mathbb{N}_{\perp}$. In the quantitative semantics approach, the type $\mathcal{N}$ is interpreted as a set of indices $\mathbb{N}$ together with a set of non-negative sequences $\mathsf{P}\mathcal{N} \subset (\mathbb{R}^{+})^{\mathbb{N}}$ which corresponds to subprobabilistic distributions. Therefore, at this ground type, there is only a difference of presentation between the two approaches: either we weight $\perp$ with the probability of diverging or we consider sub-probabilities.

In the domain theoretic approach, M is denoted as a continuous function $[M] : \mathbb{N}_{\perp} \to \mathcal{V}(\mathbb{N}_{\perp})$. Now if we want to apply M to a probabilistic input, we need to lift its interpretation with a `let` construction: $[\mathsf{let}\ n\ \mathsf{be}\ x\ \mathsf{in}\ M] : \mathcal{V}(\mathbb{N}_{\perp}) \to \mathcal{V}(\mathbb{N}_{\perp})$ which first computes the outcome of the random variable x and then passes it to M. This consists of a call-by-value strategy:

$$[\mathsf{let}\ n\ \mathsf{be}\ x\ \mathsf{in}\ M] : x \mapsto \left(\sum_{n} [\![M]\!]_{n,q} x_n \right)_q .$$

Indeed, alike the Law of total probabilities, the probability to get the output q is the sum over the outcomes n of the random variable x of the probability that M outputs q knowing that the input is n together with the probability that x outcomes n.

In the quantitative semantics approach, the program M is interpreted as a probabilistic operator that preserves subprobability distributions: $[\![M]\!] : \mathsf{P}\mathcal{N} \to \mathsf{P}\mathcal{N}$. Now, the random variable is handled following a call-by-name strategy:

$$[\![M]\!] : x \mapsto \left(\sum_{\mu=[n_1,\dots,n_k]} [\![M]\!]_{\mu,q} \prod_{i=1}^{k} x_{n_i} \right)_q .$$

2. In the call-by-value evaluation strategy, the argument expression is evaluated once for all, and the resulting value is passed to the function that will use it at will.

Indeed, to compute the probability to get output value q: first, we sum over the bags of outcomes n_i of the random variable x (one outcome for each occurrence of x in M); then we multiply the probability $[\![M]\!]_{\mu,q}$ that M outputs q with this bag $\mu = [n_1, \ldots, n_k]$ of inputs and the probability $\prod_{i=1}^{k} x_{n_i}$ that x outcomes successively $n_1, \ldots, n_k$. Notice that we do not take into account the order these outcomes appear (this model is based on the relational model and not on games semantics which distinguished the order the arguments are evaluated). Noteworthy, the interpretation of a program is a power series with potentially infinitely many parameters which are the coefficients of the sequence x. This is the key property of our approach.

The problematic in the domain theoretic approach is to find a full subcategory of continuous domains that is both cartesian closed and closed under the probabilistic monad $\mathcal{V}$ (see Jung and Tix [1998]). In the quantitative semantics approach, we do have a cartesian closed category[3] which is moreover fully abstract for a probabilistic extension of PCF.

More recently, the domain theoretic approach has been renewed. We can mention the cartesian closed category of Kegelspitzen Keimel and Plotkin [2017] and Scott continuous functions, considered in Rennela [2016] as a model for discrete probabilistic programming. Let us mention the model based on event structure and game semantics in Castellan et al. [2018] who obtained a Full abstraction result derived from ours.

In the last years, continuous probabilistic languages have gain an increasing interest. The functional paradigm is particulary relevant in this area (see Park et al. [2008]). Let us mention the probabilistic language WebPPL (see Goodman and Tenenbaum [2014]) which is built on top of a purely functional fragment of `Javascript`, `Church` (see Goodman et al. [2008]) and Anglican (see Wood et al. [2014]).

The formal methods for those languages have inherited from the work of Kozen [1979], where programs are interpreted as stochastic kernels between measurable spaces. The possible configurations of the memory are described by measurable spaces, while kernels define the probabilistic transformation of the memory induced by program execution. Kozen's approach cannot be trivially extended to higher-order types, because there is no clear notion of measurable subset for a functional space. Indeed, we do not know which measurable space can describe values of type, say, $\mathbb{R} \to \mathbb{R}$ (see Aumann [1961] for details). Panangaden [1999] reframed the work by Kozen in a categorical setting, using the category **Kern** of stochastic kernels. This category has been presented as the Kleisli category of the so-called Giry's monad (see Giry [1982]) over the category **Meas** of measurable spaces and measurable functions. One can precisely state the issue for higher-order types in this framework — both **Meas** and **Kern** are cartesian categories but not closed.

The denotational semantics approach to probabilistic programming has been recently relaunched by the increasing importance of continuous distributions and sampling primitives. Indeed, this raises the question of the *measurability of a morphism* as the interpretation of the sampling primitives requires integration. This question has not been investigated yet in the domain theoretic approach and forces to introduce a new line of works which puts the focus on measurability.

The challenge is to define a cartesian closed category in which base types such as reals would be interpreted as measurable spaces. As already mentioned, the category **Meas** of measurable spaces and functions is cartesian but not closed. To overcome this problem, Staton et al. [2016] embed **Meas** in a functor category which is cartesian closed although not well-pointed. Then, to get a more concrete and a well-pointed category, they introduce the category of *quasi-borel spaces* (see Heunen et al. [2017]) which are sets endowed with a set of random variables. Notice that both categories miss the order completeness, and thus the possibility of interpreting higher-order recursion. This is a big difference with our model, presented in §3.5, of measurable cones and stable, measurable functions which is order complete.

Contributions

In Ehrhard et al. [2011], we have proved an adequacy theorem for Probabilistic Coherent Spaces and a probabilistic extension of the pure λ-calculus: the probability that a term reduces to a head normal form is equal to its denotation computed on a suitable set of values.

In Ehrhard et al. [2014], we have proved a Full-Abstraction theorem for Probabilistic Coherent spaces and a probabilistic extension of probabilistic PCF, a well-known simply typed universal functional language. The type hierarchy is based on a single ground type of natural numbers. However, we soon realized that the call-by-value strategy was not expressive enough (see our Las Vegas example in §3.3.2). Thus, we introduced a `let` construct, allowing a call-by-value strategy for ground-type arguments. This extension reflects a denotational property. Indeed, the interpretation of the type of natural numbers is

3. Notice that in this model, objects are domains and morphisms are scott-continuous. However, this category is not a full subcategory of the category of domains so that we do not solve the domain theoretic issue.

equipped with a coalgebraic structure. Thus, it inherits properties of the exponential of Linear Logic: ground-type values can be erased and duplicated at will. That is why they are compatible with the call-by-value strategy. In Ehrhard et al. [2018a], we extended the Full-Abstraction result to this refined language. Then, Ehrhard [2016] proved that this `let` construct can be extended to all coalgebraic types. Using ideas from half-polarized linear logic, he proposed a Call-by-push-value language in which values can be erased or duplicated. In Ehrhard and Tasson [2018], we studied a probabilistic version of this language and adapted the Full-Abstraction result. Although the proof follows the same schema, we had to develop new concepts to adapt it.

In Crubillé et al. [2017], we proved that the exponential comonad of Probabilistic Coherent Spaces is free. The proof consists in showing that the free comonad computed by the formula introduced in MEL [2017] is the same as the exponential of Probabilistic Coherent Spaces.

In Ehrhard et al. [2018b], we moved from discrete probabilities to continuous ones. We proved that the cartesian closed category of measurable cones and stable, measurable functions is adequate for a version of Probabilistic PCF with the single ground type of reals. Moreover, it is well-suited to interpret simple randomized algorithms such as Metropolis-Hasting.

Organization of the Chapter

This chapter is organized as follows. We first give a brief introduction to probabilistic coherent spaces (Section 3.1). We take the opportunity to state that the exponential is free (§3.1.2). We introduce all the concepts needed for interpreting the three languages we consider in the following: Λ^+, the pure λ-calculus (Section 3.2); pPCF, the probabilistic extension of PCF (Section 3.3); and $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$, the probabilistic extension of CBPV (Section 3.4). Finally, we turn to our last piece of work (Section 3.5): After a short presentation of the model of measurable cones and stable, measurable functions, we present pRPCF, the probabilistic extension of real PCF, states the adequacy result and study in details interpretation of probabilistic programs. To finish we describe in §3.5.7 the probability theory that is needed to understand this section.

Notations

Let I and J be countable sets. Given $u, u' \in (\mathbb{R}^+)^I$, we define the pairing

$$\langle u, u' \rangle = \sum_{i \in I} u_i u'_i \in \mathbb{R}^+ \cup \{\infty\}\,.$$

Let $u \in (\mathbb{R}^+)^I$ and $v \in (\mathbb{R}^+)^J$. We denote $(u \otimes v)_{(i,j)} = u_i v_j \in (\mathbb{R}^+)^{I \times J}$ the tensor product of u and v.
Let $t \in (\mathbb{R}^+)^{I \times J}$ be a matrix. We denote $t^\perp \in (\mathbb{R}^+)^{J \times I}$ the transpose of t, that is $(t^\perp)_{j,i} = t_{i,j}$.
A *finite multiset* of elements of I is a function $\mu : I \to \mathbb{N}$ whose *support* $\mathsf{supp}(\mu) = \{a \in I \mid \mu(a) \neq 0\}$ is finite. We denote as $\mathcal{M}_{\mathrm{fin}}(I)$ the set of all finite multisets of elements of I. Given a finite family $a_1, \ldots, a_n$ of elements of I, let $[a_1, \ldots, a_n]$ be the multiset μ such that $\mu(a) = \#\{i \mid a_i = a\}$. We use additive notations for multiset unions: $\sum_{i=1}^k \mu_i$ is the multiset μ such that $\mu(a) = \sum_{i=1}^k \mu_i(a)$. The empty multiset is denoted as 0 or $[\,]$. If $k \in \mathbb{N}$, the multiset $k\mu$ maps a to $k\mu(a)$.
Given terms M and N and given a variable x, we use $M\,[N/x]$ for the term M where x is substituted with N.
The set of head normal forms is denoted as hnf, the letter H will be ranged over hnf.

Symbols

$A, B \ldots$ types of pPCF and pRPCF. 78, 94

$E[\cdot]$ Evaluation contexts of pPCF, $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$ and pRPCF. 79, 87, 96

$M, N, \ldots$ terms of pPCF, $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$ and pRPCF. 74, 78, 86, 94

$V, W, \ldots$ value terms of $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$. 87

$\varphi, \psi, \ldots$ positive types of $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$. 86

$\sigma, \tau, \ldots$ general types of $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$. 86

3.1 Probabilistic coherence spaces

This first part is dedicated to Probabilistic Coherence Spaces which constitute a model of classical Linear Logic introduced by Girard and developed in full details in Danos and Ehrhard [2011] to which we refer. The reader interested in more involved results can refer to Ehrhard et al. [2018a] and Ehrhard and Tasson [2018].

3.1.1 Definition and basic properties of Pcoh

As often in quantitative models of Linear Logic, a probabilistic coherent space is defined by a *web* which represents the possible outcomes of a program and by a subspace of sequences indexed by its web, which represents the different outcomes of probabilistic programs weighted by a coefficient corresponding roughly to the probability that this outcomes occurs. This subspace has to be equal to its biorthogonal for a well-suited biorthogonality. This corresponds to the fact that programs and environments interact correctly.

Let I be a countable set. Given $u, u' \in (\mathbb{R}^+)^I$, we say that u and u' are orthogonal whenever $\langle u, u' \rangle \leq 1$ (recall the pairing notation introduced in §3). The orthogonal of $\mathcal{X} \subseteq (\mathbb{R}^+)^I$, is then given by $\mathcal{X}^\perp = \{u' \in (\mathbb{R}^+)^I \mid \forall u \in \mathcal{X}\ \langle u, u' \rangle \leqslant 1\}$ and satisfies as usual $\mathcal{X} \subseteq \mathcal{Y} \Rightarrow \mathcal{Y}^\perp \subseteq \mathcal{X}^\perp$, $\mathcal{X} \subseteq \mathcal{X}^{\perp\perp}$ and $\mathcal{X}^{\perp\perp\perp} = \mathcal{X}^\perp$.

Definition 39. A *probabilistic coherence space* (PCS) is a pair $X = (|X|, \mathsf{P}X)$ where $|X|$ is a countable set called the web and $\mathsf{P}X \subseteq (\mathbb{R}^+)^{|X|}$ satisfies

- $\mathsf{P}X^{\perp\perp} = \mathsf{P}X$ (equivalently, $\mathsf{P}X^{\perp\perp} \subseteq \mathsf{P}X$),
- for each $a \in |X|$ there exists $u \in \mathsf{P}X$ such that $u_a > 0$,
- for each $a \in |X|$ there exists $A > 0$ such that $\forall u \in \mathsf{P}X\ u_a \leqslant A$.

The second condition ensures that every index of $|X|$ is covered by vector which has a corresponding non zero coefficient on this index. The third condition implies that the set of coefficient on one given index is bounded. The purpose of these two conditions is to prevent infinite coefficients to appear in the semantics. This property in turn will be essential for guaranteeing the morphisms interpreting proofs to be analytic functions, which will be the key property to prove full abstraction.

Actually, $\mathsf{P}X$ has many good properties:

- $\mathsf{P}X$ is unitary with respect to the norm induced by the orthogonality: Given $u \in \mathsf{P}X$, we define the **norm** $\|u\|_X = \sup\{\langle u, u' \rangle \mid u' \in \mathsf{P}X^\perp\}$, so that $\|u\|_X \in [0, 1]$ by definition.
- $\mathsf{P}X$ is a cone: Given $u, v \in \mathsf{P}X$ and $\alpha, \beta \in \mathbb{R}^+$ such that $\alpha + \beta \leqslant 1$, one has $\alpha u + \beta v \in \mathsf{P}X$.
- $\mathsf{P}X$ is equipped with the most obvious **partial order** relation (using the usual order on $\mathbb{R}$):

$$u \leqslant v \text{ iff } \forall a \in |X|\ u_a \leqslant v_a\,. \tag{3.1}$$

Finally, PCS are domains, relating them with usual models of probabilistic programming:

Proposition 40. *$\mathsf{P}X$ is an ω-continuous domain.*

As a consequence, given a family $(u(i))_{i\in\mathbb{N}}$ of elements of $\mathsf{P}X$ and a family $(\alpha_i)_{i\in\mathbb{N}}$ of elements of $\mathbb{R}^+$ such that $\sum_{i\in\mathbb{N}} \alpha_i \leqslant 1$, one has $\sum_{i\in\mathbb{N}} \alpha_i u(i) \in \mathsf{P}X$.

Morphisms of PCSs

A morphism from a PCS X to a PCS Y is a matrix $t \in (\mathbb{R}^+)^{|X|\times|Y|}$ which maps $\mathsf{P}X$ to $\mathsf{P}Y$. To make this statement precise, we define the application $t\,u \in \overline{\mathbb{R}^+}^{|Y|}$ of a matrix t to a vector $u \in \mathsf{P}X$ as[4] $(t\,u)_b = \sum_{a\in|X|} t_{a,b}u_a$.

We say that t is a (linear) **morphism** from X to Y if $u \in \mathsf{P}X$ implies $t\,u \in \mathsf{P}Y$, that is

$$\forall u \in \mathsf{P}X\, \forall v' \in \mathsf{P}Y^\perp,\quad \sum_{(a,b)\in|X|\times|Y|} t_{a,b}u_a v'_b \leqslant 1\,.$$

4. This is an unordered sum, which is potentially infinite. It makes sense since all its terms are non-negaitve.

$$X^\perp = \left(|X|, \mathsf{P}X^\perp\right) \qquad \mathbf{1} = \bot = \left(\{*\}, [0,1]^{\{\star\}}\right)$$

$$X \otimes Y = \left(|X| \times |Y|, \{u \otimes v \mid u \in \mathsf{P}X \text{ and } v \in \mathsf{P}Y\}^{\perp\perp}\right)$$
$$X \multimap Y = \left(|X| \times |Y|, \{t \mid \text{if } u \in \mathsf{P}X \text{ then } t\,u \in \mathsf{P}Y\}\right)$$

$$\mathop{\&}_{i\in I} X_i = \left(\bigcup_{i\in I}\{i\} \times |X_i|, \{u \mid \forall i \in I\ u(i) \in \mathsf{P}X_i, \text{ where } \forall a \in |X_i|\ u(i)_a = u_{(i,a)}\}\right)$$
$$\bigoplus_{i\in I} X_i = \left(\bigcup_{i\in I}\{i\} \times |X_i|, \left\{u \mid \forall i \in I\ u(i) \in \mathsf{P}X_i \text{ and } \sum_{i\in I} \|u(i)\|_{X_i} \leqslant 1\right\}\right)$$

$$!X = \left(\mathcal{M}_{\mathrm{fin}}(|X|), \{u^! \mid u \in \mathsf{P}X\}^{\perp\perp}\right), \quad \text{where } u^!_\mu = \prod_{a\in|X|} u_a^{\mu(a)}$$

Figure 3.1 – Constructions of LL in **Pcoh**

The diagonal matrix $\mathrm{Id} \in (\mathbb{R}^+)^{|X|\times|X|}$, given by $\mathrm{Id}_{a,b} = 1$ if $a = b$ and $\mathrm{Id}_{a,b} = 0$ otherwise, is a morphism. Composition of morphisms is defined as matrix multiplication: let s be a morphism from X to Y and t a morphism from Y to Z. We define $t\,s \in (\mathbb{R}^+)^{|X|\times|Z|}$ by

$$(t\,s)_{a,c} = \sum_{b\in|Y|} s_{a,b}t_{b,c}$$

and a simple computation shows that $t\,s \in \mathbf{Pcoh}(X, Z)$. More precisely, we use the fact that, given $u \in \mathsf{P}X$, one has $(t\,s)\,u = t\,(s\,u)$. Associativity of composition holds because matrix multiplication is associative. Id_X is the identity morphism at X.

PCSs and morphisms of PCSs constitute a category that we denote **Pcoh**.

The partially ordered class of probabilistic coherence spaces

In order to interpret fixpoint of types in **Pcoh**, we need to equip this category with a well-behaved order between PCSs. Intuitively, $X \subseteq Y$ if the web of X is included in the web of Y and if the sequences of X correspond to restrictions of sequences of Y to the web of X.

Given $u \in (\mathbb{R}^+)^{|X|}$ and $I \subseteq |X|$ we use $u_{|I}$ for the element v of $(\mathbb{R}^+)^{|X|}$ such that $v_a = u_a$ if $a \in I$ and $v_a = 0$ otherwise. Of course if $u \in \mathsf{P}X$, then its restriction to I satisfies $u_{|I} \in \mathsf{P}X$.

We define[5] an **order relation**, denoted as $\subseteq$, on probabilistic coherent spaces:

$$X \subseteq Y \Leftrightarrow \begin{cases} |X| \subseteq |Y|\,, \\ \mathsf{P}(X) = \{v_{||X|} \text{ s.t. } v \in \mathsf{P}(Y)\}. \end{cases} \tag{3.2}$$

Notice that the orthogonal is non-decreasing: if $X \subseteq Y$ then $X^\perp \subseteq Y^\perp$.

We define the category $\mathbf{Pcoh}_\subseteq$ as the partially ordered class whose objects are those of **Pcoh** and such that $\mathbf{Pcoh}_\subseteq(X, Y)$ has a unique element $\eta_{X,Y}$ when $X \subseteq Y$ or is empty when $X \not\subseteq Y$.

Proposition 41. $\mathbf{Pcoh}_\subseteq$ *is directed co-complete.*

3.1.2 A model of Linear Logic

Now, we give the description of the category **Pcoh** as a model of Linear Logic. For this purpose, we use the notion of Linear Category introduced by Bierman [1995], as presented in Melliès [2009] which is our main reference for this topic. The constructions are summarized in Figure 3.1. They all define PCSs and the justification can be found in Danos and Ehrhard [2011].

We first present the $*$-autonomous structure which is based on the multiplicative constructs: the tensor product $\otimes$, the linear map $\multimap$ and the dualizaing object $\bot$.

5. This order relation can be equivalently defined as a retraction-embedding pair (which is the usual way of computing fixpoints)

Then we turn to the cartesian structure which is based on the additive constructs: the cartesian product & and the coproduct $\oplus$.

Finally, we present the last but not least ingredient which is the exponential structure based on the exponential comonad ! that allows to duplicate or erase in a model of linear logic.

We refer to Page 64 for notations used in this section.

$*$-autonomous structure

The bifunctor $\otimes : \mathbf{Pcoh} \times \mathbf{Pcoh} \to \mathbf{Pcoh}$ is defined by

$$|X \otimes Y| = |X| \times |Y|\,,$$
$$\mathsf{P}(X \otimes Y) = \{x \otimes y \text{ s.t. } x \in \mathsf{P}(X), y \in \mathsf{P}(Y)\}^{\perp\perp}.$$

The action of $\otimes$ on morphisms $u \in \mathbf{Pcoh}(X, Y)$ and $v \in \mathbf{Pcoh}(X', Y')$ is defined by $(u \otimes v)_{(a,a'),(b,b')} = u_{a,b} v_{a',b'}$, for $(a, a') \in |X \otimes X'|$, $(b, b') \in |Y \otimes Y'|$. The unity of $\otimes$ is given by the singleton web PCS $\mathbf{1} = (\{\star\}, [0,1]^{\{\star\}})$.

The object of linear morphisms $X \multimap Y$ is defined as

$$|X \multimap Y| = |X| \times |Y|\,,$$
$$\mathsf{P}(X \multimap Y) = \mathbf{Pcoh}(X, Y).$$

One proves that $X \multimap Y$ is a PCS by checking $X \multimap Y = (X^\perp \otimes Y)^\perp$.

The definition of PCS morphisms implies the following characterization of elements of $X \multimap Y$:

Lemma 42. *Let X and Y be PCSs. If $t \in (\mathbb{R}^+)^{|X| \times |Y|}$, then $t \in \mathsf{P}(X \multimap Y)$ iff $\forall u \in \mathsf{P}X$, $t\, u \in \mathsf{P}Y$.*

The evaluation morphism $\mathrm{ev} \in \mathbf{Pcoh}(X \otimes (X \multimap Y), Y)$ is given by $\mathrm{ev}_{(a,(a',b)),b'} = \delta_{a,a'}\delta_{b,b'}$. If $t \in \mathbf{Pcoh}(Z \otimes X, Y)$, then the associated linearly curryfied morphism $\mathrm{cur}(t) \in \mathbf{Pcoh}(Z, X \multimap Y)$ is given by $\mathrm{cur}(t)_{(c,(a,b))} = t_{((c,a),b)}$.

Last, the dualizing object $\perp$ is defined as the dual of $\mathbf{1}$ which is indeed equal to $\mathbf{1}$: $\perp = \mathbf{1}^\perp = \mathbf{1}$.

Cartesian structure

Pcoh admits the cartesian product of any countable family $(X_i)_{i \in I}$ of PCSs, defined by

$$|\&_{i \in I} X_i| = \cup_{i \in I}(\{i\} \times |X_i|),$$
$$\mathsf{P}(\&_{i \in I} X_i) = \left\{ x \in (\mathbb{R}^+)^{|\&_{i \in I} X_i|} \text{ s.t. } \forall i \in I,\ \pi_i(x) \in \mathsf{P}(X_i) \right\}.$$

where $\pi_i(x)$ is the vector in $(\mathbb{R}^+)^{|X_i|}$ denoting the i-th component of x, i.e. $\pi_i(x)_a = x_{(i,a)}$. The j-th projection $\mathrm{pr}^j \in \mathbf{Pcoh}(\&_{i \in I} X_i, X_j)$ is defined by $\mathrm{pr}^j_{(i,a),b} = \delta_{i,j}\delta_{a,b}$.

Notice that the empty product yields the terminal object of **Pcoh**. We may write $A_1 \,\&\, A_2$ for the binary product: we can present any $x \in \mathsf{P}(A_1 \,\&\, A_2)$ as the pair $(\pi_1(x), \pi_2(x)) \in \mathsf{P}(A_1) \times \mathsf{P}(A_2)$ of its components. When X_i is equal to X for each $i \in I$, we may write the product $\&_{i \in I} X_i$ by X^I.

Example 43. We introduce the key example of natural numbers and define $\mathbf{N} = \oplus_{i \in \mathbb{N}} \mathbf{1}$, that is $|\mathbf{N}| = \mathbb{N}$ and $u \in (\mathbb{R}^+)^{\mathbb{N}}$ belongs to $\mathsf{P}\mathbf{N}$ if $\sum_{n \in \mathbb{N}} u_n \leqslant 1$.

Exponential structure

The functorial promotion $! : \mathbf{Pcoh} \to \mathbf{Pcoh}$ is defined on objects by

$$|!X| = \mathcal{M}_{\mathrm{fin}}(|X|),$$
$$\mathsf{P}(!X) = \{x^! \text{ s.t. } x \in \mathsf{P}(X)\}^{\perp\perp},$$

where $x^!$ is the vector of $(\mathbb{R}^+)^{\mathcal{M}_{\mathrm{fin}}(|X|)}$ defined by $x^!_\mu = \prod_{a \in |X|} x_a^{\mu(a)}$, for any $\mu \in \mathcal{M}_{\mathrm{fin}}(|X|)$.

The action of ! on a morphism $t \in \mathbf{Pcoh}(X, Y)$ is defined by, for any $\mu \in |!X|$, $\mu' \in |!Y|$,

$$(!t)_{\mu,\mu'} = \sum_{r \in L(\mu,\mu')} \begin{bmatrix} \mu \\ \mu' \end{bmatrix} \left(\prod_{(a,b) \in |X| \times |Y|} t_{a,b}^{r(a,b)} \right)$$

where $L(\mu,\mu')$ is the set of multisets over $|X|\times|Y|$ whose first projection is μ and whose second projection is μ' (occurrences do matter) and where

$$\begin{bmatrix}\mu'\\ \rho\end{bmatrix} = \prod_{b\in|Y|}\frac{\mu'(b)!}{\prod_{a\in|X|}\rho(a,b)!}$$

is the number[6] of ways to associate the elements of μ' to the elements of μ in order to get ρ. Let us underline that such a coefficient introduces scalars greater than 1.

As an example, let $\mathbf{Bool} = (1\ \&\ 1)^\perp$ and call $\mathbf{t}$, $\mathbf{f}$ the only two elements of its web. Notice that $\mathsf{P}(\mathbf{Bool}) = \{x\in(\mathbb{R}^+)^{\{\mathbf{t},\mathbf{f}\}}\text{ s.t. }x_\mathbf{t}+x_\mathbf{f}\leqslant 1\}$. Consider $t\in(\mathbb{R}^+)^{|\mathbf{Bool}|\times|1|}$ defined by $t_{\mathbf{t},*}=t_{\mathbf{f},*}=1$. Notice $t\in\mathbf{Pcoh}(\mathbf{Bool},1)$, hence $!t\in\mathbf{Pcoh}(!\mathbf{Bool},!1)$. We have $!t_{[\mathbf{t},\mathbf{f}],[*,*]} = \left[\begin{smallmatrix}[*,*]\\ [(\mathbf{t},*),(\mathbf{f},*)]\end{smallmatrix}\right]t_{\mathbf{t},*}t_{\mathbf{f},*}=2$. This shows why, in the definition of a PCS, scalars are in $\mathbb{R}^+$ instead of restricting them to $[0,1]$.

Let us underline that adding the coefficient $\left[\begin{smallmatrix}p\\ r\end{smallmatrix}\right]$ in the definition of $!t$ is crucial for ! to be a functor, in fact for having the commutation with the composition (see [Danos and Ehrhard, 2011, Sect. 1.6]).

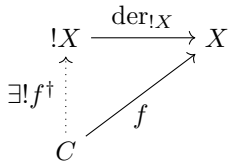
The functorial promotion is equipped with a structure of comonad. The counit (also called *dereliction*) is $\mathrm{der}_X\in\mathbf{Pcoh}(!X,X)$ given by $(\mathrm{der}_X)_{\mu,a}=\delta_{\mu,[a]}$. The comultiplication (also called *digging*) is $\mathrm{dig}_X\in\mathbf{Pcoh}(!X,!!X)$ given by $(\mathrm{dig}_X)_{\mu,M}=\delta_{\mu,\biguplus M}$, where $\biguplus M$ is the multiset in $|!X|$ obtained as the multiset union of the multisets in $M\in|!!X|$.

Free exponential

From the additive structure and the lax symmetric monoidal structure, we can construct a structure of comonoid on $!X$ whose contraction $\mathrm{contr}_X\in\mathbf{Pcoh}(!X,!X\otimes!X)$ and weakening $\mathrm{weak}_X\in\mathbf{Pcoh}(!X,\mathbf{1})$ are given by $(\mathrm{contr}_X)_{\mu,(\mu',\mu'')}=\delta_{\mu,\mu'\uplus\mu''}$, and $(\mathrm{weak}_X)_{\mu,\star}=\delta_{\mu,[]}$ for any $\mu,\mu',\mu''\in\mathcal{M}_{\mathrm{fin}}(|X|)$.

We proved in Crubillé et al. [2017] that it is the free comonoid structure. Thanks to the construction introduced in MEL [2017], the free exponential modality (under some commutativity conditions) can be constructed as the limit of its approximants. We proved that the free exponential and the usual exponential of **Pcoh** are actually the same.

Theorem 44. *For all commutative comonoid C and $f\in\mathbf{Pcoh}(C,X)$, there exists only one commutative comonoid morphism $f^\dagger$ commuting the diagram:*



This means that $!X$ is the terminal object of the category of commutative comonoids over X.

3.1.3 A model of typed and untyped λ-calculus

As usual, we can recover a model of typed λ-calculus from a model of linear logic through the Kleisli category construction. Indeed, the intuitionistic arrow $X\Rightarrow Y$ of typed λ-calculus is encoded by $!X\multimap Y$ using the linear arrow and the exponential of Linear Logic.

Let us consider the Kleisli category $\mathbf{Pcoh}_!$ induced by the comonad associated with the functorial promotion. The objects of $\mathbf{Pcoh}_!$ are the PCSs and the set of morphisms $\mathbf{Pcoh}_!(X,Y)$ is equal to $\mathbf{Pcoh}(!X,Y)$.

The morphisms in the Kleisli category enjoy an alternative description as **analytic functions**. This is the key property of our work, so that we devote it some explanations.

Let $s\in(\mathbb{R}^+)^{|!X\multimap Y|}$. We define a function $\widehat{s}:\mathsf{P}X\to\overline{\mathbb{R}^+}^{|Y|}$ such that, given $u\in\mathsf{P}X$,

$$\widehat{s}(u) = s\,u^! = \left(\sum_{\mu\in|!X|}s_{\mu,b}u^\mu\right)_{b\in|Y|}.$$

6. Note the asymmetry between μ' and μ: if $\mu=[a,b]$, $\mu'=[c,c]$, then $L(\mu,\mu')$ has exactly one element, $[(a,c),(b,c)]$, and the number of ways of getting it is $\left[\begin{smallmatrix}\mu'\\ [(a,c),(b,c)]\end{smallmatrix}\right]=2$, while, inverting μ' and μ, we have that $L(\mu',\mu)$ has always one element, $[(c,a),(c,b)]$, but the number of ways of getting it is $\left[\begin{smallmatrix}m\\ [(c,a),(c,b)]\end{smallmatrix}\right]=1$.

Because, $u^! \in \mathsf{P}(!X)$ and thanks to Lemma 42, we get the characterization:

$$s \in \mathsf{P}(!X \multimap Y) \qquad \text{iff} \qquad \forall u \in \mathsf{P}X,\ \widehat{s}(u) \in \mathsf{P}Y,$$

which can be rephrased in terms of Kleisli morphisms:

Theorem 45. *If s and $s' \in \mathbf{Pcoh}_!(X,Y)$ then, $s = s'$ (as matrices) iff $\widehat{s} = \widehat{s'}$ (as functions).*

This means that any morphism in $\mathbf{Pcoh}_!(X,Y)$ can be identified with the associated analytic function from $\mathsf{P}(X)$ to $\mathsf{P}(Y)$, and this identification is compatible with composition. Thus, in the sequel we will give a morphism in $\mathbf{Pcoh}_!(X,Y)$ either as a matrix in $\mathbf{Pcoh}(!X \multimap Y)$ or as a composition of analytic maps.

Let us describe the counit and the comultiplication of the comonad ! as analytic functions:

$$\forall u \in \mathsf{P}X,\ \widehat{\mathrm{der}_X}(u) = u \quad \text{and} \quad \widehat{\mathrm{dig}_X}(u) = (u^!)^!.$$

Then, we can describe the structure of the Kleisli category. The identity Id^X on X is the dereliction der_X, while the composition $s \circ_! t$ of two morphisms $t \in \mathbf{Pcoh}_!(X,Y)$, $s \in \mathbf{Pcoh}_!(Y,Z)$ is defined by $s \circ_! t = s \circ\, !t \circ \mathrm{dig}_X$ or equivalently by $\widehat{s \circ_! t} = \widehat{s} \circ \widehat{t}$.

As it is known from Girard [1987], the monoidal closedness of **Pcoh** is lifted to a cartesian closedness in $\mathbf{Pcoh}_!$ by the monoidality of the comonad !: there are an isomorphism m^0 between $\mathbf{1}$ and $!\top$ and another one $\mathrm{m}^2_{X,Y}$ between $!(X \,\&\, Y)$ and $!X \otimes !Y$. These isomorphims are defined by their matrices $\mathrm{m}^0 \in (\mathbb{R}^+)^{|\mathbf{1} \multimap !\top|}$ as $\mathrm{m}^0_{*,[]} = 1$ and $\mathrm{m}^2_{X,Y} \in (\mathbb{R}^+)^{|!X \otimes !Y \multimap !(X \& Y)|}$ as $(\mathrm{m}^2_{X,Y})_{\lambda,\rho,\mu} = \delta_{\mu,1\cdot\lambda+2\cdot\rho}$ where $i \cdot [a_1,\dots,a_n] = [(i,a_1),\dots,(i,a_n)]$.

Let us use the analytic function approach to describe the cartesian closed structure of the kleisli category.

The product of a countable family $(X_i)_{i\in I}$ is the PCS $\&_{i\in I} X_i$ endowed with the projections $\mathrm{Pr}^j \in \mathbf{Pcoh}_!(\&_{i\in I} X_i, X_j)$ characterized by the analytic function $\widehat{\mathrm{Pr}}^j$ such that $\widehat{\mathrm{Pr}}^j((x_i)_{i\in I}) = x_j$.

The object of the kleisli morphisms from X to Y is $X \Rightarrow Y = !X \multimap Y$. The evaluation and the curryfication are characterized as composition of analytic functions. The evaluation morphism $\mathsf{Ev} : \mathsf{P}((X \Rightarrow Y) \,\&\, X) \to \mathsf{P}(Y)$ corresponds to the analytic function such that $\widehat{\mathsf{Ev}}(f,x) = f(x)$ for any $(f,x) \in \mathsf{P}(X \Rightarrow Y) \times \mathsf{P}(X) \simeq \mathsf{P}((X \Rightarrow Y) \,\&\, X)$. Given an analytic $f : \mathsf{P}(X \,\&\, Z) \to \mathsf{P}(Y)$, its curryfication $\widehat{\mathsf{Cur}}(f) : \mathsf{P}(Z) \to \mathsf{P}(X \Rightarrow Y)$ is given by $\widehat{\mathsf{Cur}}(f)(z)(x) = f(x,z)$ for $x \in \mathsf{P}(X)$, $z \in \mathsf{P}(Z)$.

Thanks to the description of Kleisli morphisms as analytic functions, we can relate the model of PCS with the domains theoretic approach. Indeed, we have already stated that PCSs are ω-continuous (see Proposition 40). Now, we states that Kleisli morphims are Scott-continuous:

Proposition 46. *Let $s \in \mathbf{Pcoh}_!(X,Y)$. The function $\widehat{s}$ is Scott-continuous.*

Example 47. Take $X = Y = \mathbf{1}$. A morphism in $\mathbf{Pcoh}(!\mathbf{1},\mathbf{1})$ can be seen as a function $f : [0,1] \to [0,1]$ such that $f(u) = \sum_{n=0}^{\infty} s_n u^n$ where the s_n's are non-negative and satisfy $\sum_{n=0}^{\infty} s_n \leqslant 1$. Of course, not all Scott continuous function $[0,1] \to [0,1]$ are of that particular shape! Take for instance the function $f : [0,1] \to [0,1]$ defined by $f(u) = 0$ if $u \leqslant \frac{1}{2}$ and $f(u) = 2u - 1$ if $u > \frac{1}{2}$; this function f is Scott continuous but has no derivative at $u = \frac{1}{2}$ and therefore cannot be expressed as a power series.

In order to interpret recursive programs in pPCF (see Section 3.3, we need that the Kleisli category is cpo-enriched thanks to the monotone convergence theorem. Moreover, least upper bounds are computed pointwise.

Proposition 48. *Let $s, s' \in \mathsf{P}(!X \multimap Y)$. If $s \leqslant s'$, then $\forall u \in \mathsf{P}X\ \widehat{s}(u) \leqslant \widehat{s'}(u)$. Let $(s(i))_{i\in\mathbb{N}}$ be a monotone sequence of elements of $\mathbf{Pcoh}_!(X,Y)$ and let $s = \sup_{i\in\mathbb{N}} s(i)$. Then $\forall u \in \mathsf{P}X\ \widehat{s}(u) = \sup_{i\in I} \widehat{s_i}(u)$.*

Remark. We can have $s, s' \in \mathbf{Pcoh}(!X,Y)$ such that $\forall u \in \mathsf{P}X\ \widehat{s}(u) \leqslant \widehat{s'}(u)$ but without having that $s \leqslant s'$. Take for instance $X = Y = \mathbf{1}$. As in the example above we can see $\widehat{s}$ and $\widehat{s'}$ as functions $[0,1] \to [0,1]$ given by $\widehat{s}(u) = \sum_{n=0}^{\infty} s_n u^n$ and $\widehat{s'}(u) = \sum_{n=0}^{\infty} s'_n u^n$, and $s \leqslant s'$ means that $\forall n \in \mathbb{N}\ s_n \leqslant s'_n$. Then let s be defined by $s_n = 1$ if $n = 2$ and $s_n = 0$ otherwise, and s' be defined by $s'_n = 1$ if $n = 1$ and $s'_n = 0$ otherwise. We have $\widehat{s}(u) = u^2 \leqslant \widehat{s'}(u) = u$ for all $u \in [0,1]$ whereas s and s' are not comparable in $\mathsf{P}(!\mathbf{1} \multimap \mathbf{1})$.

Furthermore, the intepretation of recursive programs can also be computed thanks to the fixpoint operator that can be defined in the Kleisli category.

Proposition 49. *The Kleisli category* $\mathbf{Pcoh}_!$ *has a least fixpoint operator.*

Proof. Let X be a PCS and $\mathcal{F}_0 \in \mathbf{Pcoh}_!(((X \Rightarrow X) \Rightarrow X) \,\&\, (X \Rightarrow X), X)$ be

$$\begin{array}{ccc} ((X \Rightarrow X) \Rightarrow X) \,\&\, (X \Rightarrow X) & & X \\ \downarrow{\scriptstyle \langle \mathrm{pr}_2, \mathrm{pr}_1, \mathrm{pr}_2 \rangle} & & \uparrow{\scriptstyle \mathsf{Ev}} \\ (X \Rightarrow X) \,\&\, ((X \Rightarrow X) \Rightarrow X) \,\&\, (X \Rightarrow X) & \xrightarrow{\langle \mathrm{pr}_1, \mathsf{Ev} \circ \langle \mathrm{pr}_2, \mathrm{pr}_3 \rangle \rangle} & (X \Rightarrow X) \,\&\, X \end{array}$$

We define $\mathcal{F} \in \mathbf{Pcoh}_!((X \Rightarrow X) \Rightarrow X, (X \Rightarrow X) \Rightarrow X)$ as the curryfication $\mathcal{F} = \mathsf{Cur}(\mathcal{F}_0)$. Then, let us consider $F \in \mathsf{P}((X \Rightarrow X) \Rightarrow X)$, that is $F \in \mathbf{Pcoh}(X \Rightarrow X, X)$. Then, $\widehat{\mathcal{F}}(F) = \mathsf{Ev} \circ \langle \mathrm{Id}_{X \Rightarrow X}, F \rangle$ is in $\mathbf{Pcoh}(X \Rightarrow X, X)$. Since $\mathcal{F}$ is a morphism in $\mathbf{Pcoh}$, the function $\widehat{\mathcal{F}}$ is Scott continuous and therefore has a least fixpoint $\mathrm{fix} \in \mathbf{Pcoh}(X \Rightarrow X, X)$, namely $\mathrm{fix} = \sup_{n \in \mathbb{N}} \widehat{\mathcal{F}}^n(0)$ (the sequence $(\widehat{\mathcal{F}}^n(0))_{n \in \mathbb{N}}$ is monotone in the cpo $\mathsf{P}((X \Rightarrow X) \Rightarrow X)$ because $\widehat{\mathcal{F}}$ is monotone).

If we set $\mathrm{fix}_n = \widehat{\mathcal{F}}^n(0) \in \mathbf{Pcoh}(X \Rightarrow X, X)$, we have $\mathrm{fix}_0 = 0$ and $\mathrm{fix}_{n+1} = \mathsf{Ev} \circ \langle \mathrm{Id}, \mathrm{fix}_n \rangle$. Then, given $f \in \mathbf{Pcoh}(X, X)$, we have $\widehat{\mathrm{fix}_n}(f) = \widehat{f}^n(0)$ and $\widehat{\mathrm{fix}}(f) = \sup_{n \in \mathbb{N}} \widehat{f}^n(0)$. So that, fix is the usual least fixpoint operator, and this operation turns out to be a morphism in $\mathbf{Pcoh}_!$, namely $\mathrm{fix} \in \mathbf{Pcoh}_!(X \Rightarrow X, X)$. □

This means that this standard least fixpoint operator can be described as an analytic function, which is not completely obvious at first sight.

In order to interpret pure λ-calculus (see Section 3.2.1), we need a **reflexive object**, that exists in the Kleisli category as proved in [Danos and Ehrhard, 2011, Sect. 2].This means that there is a PCS D together with a pair made of $\lambda \in \mathbf{Pcoh}_!(D \Rightarrow D, D)$ and $\mathsf{app} \in \mathbf{Pcoh}_!(D, D \Rightarrow D)$ which gives rise to an isomorphism between D and $D \Rightarrow D$: $\mathsf{app} \circ_! \lambda = \mathrm{Id}^{D \Rightarrow D}$ and $\lambda \circ_! \mathsf{app} = \mathrm{Id}^D$.

Proposition 50. *The category* $\mathbf{Pcoh}_!$ *has a reflexive object* $(D, \lambda, \mathsf{app})$.

Proof. The PCS D, defined in Figure 3.2, is obtained by iterating the operation $X \mapsto (!X^{\mathbb{N}})^{\perp}$ starting

$$|D_0| = \emptyset \qquad |D_{\ell+1}| = \mathcal{M}_{\mathrm{fin}}(\bigcup_{n \in \mathbb{N}} \{n\} \times |D_\ell|) \qquad |D| = \bigcup_{\ell \in \mathbb{N}} |D_\ell|$$

$$\mathsf{P}(D_0) = \mathbf{0} \qquad \mathsf{P}(D_{\ell+1}) = \left\{ v \in (\mathbb{R}^+)^{|D_{\ell+1}|} \text{ s.t. } \forall u \in \mathsf{P}(D_\ell^{\mathbb{N}}), \langle v, u^! \rangle \leqslant 1 \right\}$$

$$\mathsf{P}(D) = \left\{ v \in (\mathbb{R}^+)^{|D|} \text{ s.t. } \forall \ell \in \mathbb{N}, \forall u \in \mathsf{P}(D_\ell^{\mathbb{N}}), \langle v_{||D_{\ell+1}|}, u^! \rangle \leqslant 1 \right\}$$

Figure 3.2 – Definition of D as the least fixpoint of the operation $X \mapsto (!X^{\mathbb{N}})^{\perp}$. Recall that $v_{||D_{\ell+1}|} \in (\mathbb{R}^+)^{|D_{\ell+1}|}$ is obtained by restricting $v \in (\mathbb{R}^+)^{|D|}$ to the indexes $|D_{\ell+1}| \subseteq |D|$.

from the empty-web PCS $D_0 = (\emptyset, \mathbf{0})$. Notice that $D_{\ell+1} = (!D_\ell^{\mathbb{N}})^{\perp}$.

D is the least upper bound of $\{D_\ell\}_{\ell \in \mathbb{N}}$, an increasing chain with respect to the order $X \subseteq Y$ defined in Paragraph 3.1.1. Moreover, the operation $X \mapsto (!X^{\mathbb{N}})^{\perp}$ is Scott continuous (i.e. monotone and preserving directed lowest upper bound), hence D is its least fixed point by Kleene-Tarski Theorem (see Tarski [1955]).

Let $\mu \in |!D_\ell| = \mathcal{M}_{\mathrm{fin}}(|D_\ell|)$ and $d \in |D_{\ell+1}| = \mathcal{M}_{\mathrm{fin}}(\cup_{n \in \mathbb{N}} \{n\} \times |D_\ell|)$, we denote:

$$\mu :: d = [(0, c) \text{ s.t. } c \in \mu] \uplus [(n+1, c) \text{ s.t. } (n, c) \in d] \tag{3.3}$$

Notice that $\mu :: d \in D_{\ell+1}$. Conversely, for any point in $d \in D_{\ell+1}$ there is a unique $\mu \in |!D_\ell|$, potentially empty, and $d' \in |D_{\ell+1}|$ such that $d = \mu :: d'$.

The empty multiset is a remarkable element of $|D|$ that we denote $\star$. In particular, we have $\star = [\,] :: \star$. This notation underlines an isomorphism between the webs of D and $D \Rightarrow D$, which is equal to $\mathcal{M}_{\mathrm{fin}}(|D|) \times |D|$.

We set $\lambda \in \mathbf{Pcoh}_!(D \Rightarrow D, D)$ and $\mathsf{app} \in \mathbf{Pcoh}_!(D, D \Rightarrow D)$ as follows: for any $\mu \in \mathcal{M}_{\mathrm{fin}}(|D \Rightarrow D|)$, $\mu', \mu'' \in \mathcal{M}_{\mathrm{fin}}(|D|)$, and $d \in |D|$,

$$\lambda_{\mu, \mu' :: c} = \delta_{\mu, [(\mu', d)]}, \qquad \mathsf{app}_{\mu'', (\mu', d)} = \delta_{\mu'', [\mu' :: d]}.$$

An easy computation shows that $\mathsf{app} \circ_! \lambda = \mathrm{Id}^{D \Rightarrow D}$ and $\lambda \circ_! \mathsf{app} = \mathrm{Id}^D$, so that $(D, \lambda, \mathsf{app})$ yields an extensional model of pure λ-calculus. □

It is significant that D satisfies two different recursive equations: $X = (!X^{\mathbb{N}})^{\perp}$ and $X = X \Rightarrow X$. The first gives the construction of D and, in fact, D is its minimal solution (with respect to $\subseteq$). The second equation is needed to interpret the pure λ-calculus. However D is not its minimal solution, since the empty-web PCS D_0 trivially satisfies $D_0 = D_0 \Rightarrow D_0$.

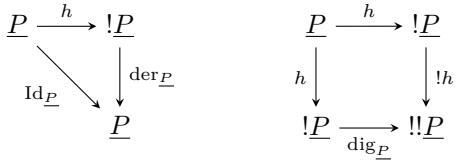
Remark that D is isomorphic to $D^{\mathbb{N}} \Rightarrow \perp$. In fact, if $\mathsf{P}(D)$ is meant to contain the denotations of terms, the vectors in $\mathsf{P}(D^{\mathbb{N}})$ represent infinite stacks of terms, whose promotion play the role of the environments.

3.1.4 Coalgebras to model call-by-value

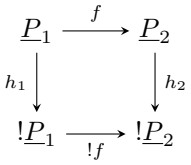
We introduce coalgebras as they are of particular interest in what follows. Indeed, as $!X$, they are equipped with a commutative comonoid that allows to duplicate and to erase through the contraction and weakening that are defined below.

Eilenberg-Moore category

By definition, a **coalgebra** of the ! comonad is a pair $(\underline{P}, h)$ where $\underline{P}$ is a PCS and $h \in \mathbf{Pcoh}(\underline{P}, !\underline{P})$ satisfying the following commutations



A **morphism of coalgebras** from $(\underline{P}_1, h_1)$ to $(\underline{P}_2, h_2)$ is an $f \in \mathbf{Pcoh}(\underline{P}_1, \underline{P}_2)$ such that the following diagram commutes



Coalgebras and their morphisms constitute the **Eilenberg-Moore** category $\mathbf{Pcoh}^{!}$ of !-coalgebras. The functor ! can be seen as a functor from **Pcoh** to $\mathbf{Pcoh}^{!}$ mapping a PCS X to a coalgebra $(!X, \mathrm{dig}_X)$ and a morphism $f \in \mathbf{Pcoh}(X, Y)$ to a coalgebra $!f$ (since dig_X is the comultiplication of the comonad !. It is right adjoint to the forgetful functor $\mathsf{U} : \mathbf{Pcoh}^{!} \to \mathbf{Pcoh}$. This adjunction maps every morphism $f \in \mathbf{Pcoh}(\underline{P}, X)$, to a morphism $f^{!} \in \mathbf{Pcoh}^{!}(P, !X)$ such that $f^{!} = !f\, \mathrm{h}_P$. Moreover, if $g \in \mathbf{Pcoh}^{!}(Q, P)$, we have $f^{!}\, g = (f\, g)^{!}$.

The Eilenberg-Moore category $\mathbf{Pcoh}^{!}$ is cartesian (with product of shape $P \otimes Q = (\underline{P} \otimes \underline{Q}, \mathrm{h}_{P \otimes Q})$ and terminal object $(\mathbf{1}, \mathrm{h}_{\mathbf{1}})$, still denoted as $\mathbf{1}$). This category is also co-cartesian with coproduct of shape $P \oplus Q = (\underline{P} \oplus \underline{Q}, \mathrm{h}_{P \oplus Q})$ and initial object $(0, \mathrm{h}_0)$ still denoted as 0. The complete definitions can be found in Ehrhard [2016]. We use $\mathrm{c}_P \in \mathbf{Pcoh}^{!}(P, P \otimes P)$ (**contraction**) for the diagonal and $\mathrm{w}_P \in \mathbf{Pcoh}^{!}(P, \mathbf{1})$ (**weakening**) for the unique morphism to the terminal object.

Dense coalgebras

Given an object P of $\mathbf{Pcoh}^{!}$, we use $\mathsf{P}^{!}(P)$ for the set of **coalgebraic elements** of $\mathsf{P}(\underline{P})$, where an element $u \in \mathsf{P}(\underline{P})$ is coalgebraic if, considered as a morphism from $\mathbf{1}$ to $\underline{P}$, u belongs to $\mathbf{Pcoh}^{!}(\mathbf{1}, P)$. This is equivalent to $u^{!} = \mathrm{h}_P\, u$.

Definition 51. An object P of $\mathbf{Pcoh}^{!}$ is *dense* if, for any object Y of **Pcoh** and any two morphisms $t, t' \in \mathbf{Pcoh}(\underline{P}, Y)$, if $t\, u = t'\, u$ for all $u \in \mathsf{P}^{!}(P)$, then $t = t'$.

The following lemma is useful in the sequel and holds in any model of Linear Logic.

Lemma 52. *Let X be a probabilistic coherence space. Then, $\mathsf{P}^{!}(!X) = \{u^{!} \mid u \in \mathsf{P}X\}$.*

Let P_ℓ and P_r be coalgebras.

Then, $P_\ell \otimes P_r$ is the cartesian product of P_ℓ and P_r in $\mathbf{Pcoh}^!$. The function $\mathsf{P}^!(P_\ell) \times \mathsf{P}^!(P_r) \to \mathsf{P}^!(P_\ell \otimes P_r)$ which maps (u, v) to $u \otimes v$ is a bijection. The projections $\mathrm{pr}_i^\otimes \in \mathbf{Pcoh}^!(P_\ell \otimes P_r, P_i)$ are characterized by $\mathrm{pr}_i^\otimes(u_\ell \otimes u_r) = u_i$.

Moreover, the function $\{\ell\} \times \mathsf{P}^!(P_\ell) \cup \{r\} \times \mathsf{P}^!(P_r) \to \mathsf{P}^!(P_\ell \oplus P_r)$ which maps (i, u) to $\mathrm{in}_i(u)$ is a bijection. The injection $u \mapsto \mathrm{in}_i(u)$ has a left inverse $\mathrm{pr}_i \in \mathbf{Pcoh}(\underline{P_\ell} \oplus \underline{P_r}, \underline{P_i})$ defined by $(\mathrm{pr}_i)_{(j,a),b} = \delta_{i,j}\delta_{a,b}$, which is not a coalgebra morphism in general.

Proof. Let $v \in \mathsf{P}^!(!X)$. We have $v^! = \mathrm{h}_{!X}\, v = \mathrm{dig}_X\, v$ hence $(\mathrm{der}_X\, v)^! = !\mathrm{der}_X\, v^! = !\mathrm{der}_X\, \mathrm{dig}_X\, v = v$. The other properties result from the fact that the Eilenberg-Moore category $\mathbf{Pcoh}^!$ is cartesian and co-cartesian with $\otimes$ and $\oplus$ as product and co-product, see Melliès [2009] for more details. □

Because of these properties we write sometimes (u_ℓ, u_r) instead of $u_\ell \otimes u_r$ when $u_i \in \mathsf{P}^!(P_i)$ for $i \in \{\ell, r\}$.

Theorem 53. *For any probabilistic coherence space X, $!X$ is a dense coalgebra. If P_ℓ and P_r are dense coalgebras then $P_\ell \otimes P_r$ and $P_\ell \oplus P_r$ are dense.*

Proof. Let X be an object of $\mathbf{Pcoh}$. By the preceding Lemma 52, $\mathsf{P}^!(!X) = \{u^! \mid u \in \mathsf{P}X\}$. It follows that $!X$ is a dense coalgebra by Theorem 45. Assume that P_ℓ and P_r are dense coalgebras. Let $t, t' \in \mathbf{Pcoh}(\underline{P_\ell} \otimes \underline{P_r}, Y)$ be such that $t\, w = t'\, w$ for all $w \in \mathsf{P}^!(P_\ell \otimes P_r)$. We have $\mathrm{cur}\,(t)$ and $\mathrm{cur}\,(t') \in \mathbf{Pcoh}(\underline{P_\ell}, \underline{P_r} \multimap Y)$. Therefore, using the density of P_ℓ, it suffices to prove that $\mathrm{cur}\,(t)\, u_\ell = \mathrm{cur}\,(t')\, u_\ell$ for each $u_\ell \in \mathsf{P}^!(P_\ell)$. Let $u_\ell \in \mathsf{P}^!(P_\ell)$, let $s = \mathrm{cur}\,(t)\, u_\ell$, $s' = \mathrm{cur}\,(t')\, u_\ell$ and let $u_r \in \mathsf{P}^!(P_r)$. We have $s\, u_r = t\,(u_\ell \otimes u_r) = t'\,(u_\ell \otimes u_r) = s'\, u_r$ since $u_\ell \otimes u_r \in \mathsf{P}^!(P_\ell \otimes P_r)$ and therefore $s = s'$ since P_r is dense. Let now $t, t' \in \mathbf{Pcoh}(\underline{P_\ell} \oplus \underline{P_r}, Y)$ be such that $t\, w = t'\, w$ for all $w \in \mathsf{P}^!(P_\ell \oplus P_r)$. To prove that $t = t'$, it suffices to prove that $t\, \mathrm{in}_i = t'\, \mathrm{in}_i$ for $i \in \{\ell, r\}$. Since P_i is dense, it suffices to prove that $t\, \mathrm{in}_i\, u = t'\, \mathrm{in}_i\, u$ for each $u \in \mathsf{P}^!(P_i)$ which follows from the fact that $\mathrm{in}_i\, u \in \mathsf{P}^! P_i$. □

The sub-category $\mathbf{Pcoh}^!$ of dense coalgebras is cartesian and co-cartesian and is well-pointed by Theorem 53. We use $\mathbf{Pcoh}^!_{\mathsf{den}}$ for this sub-category.

We end this section by an important example. Indeed, it shows that natural numbers can be duplicated and erased even in a call-by-name setting. This will be useful for interpreting probabilistic programs where it is necessary to sample a random variable and to use its outcome several times.

Example 54 (The natural numbers example)**.** Observe that $\mathbf{1}$ has a natural structure of !-coalgebra $\upsilon \in \mathbf{Pcoh}(\mathbf{1}, !\mathbf{1})$ which is obtained as the following composition of morphisms

$$\mathbf{1} \xrightarrow{\mathrm{m}^0} !\top \xrightarrow{\mathrm{dig}_\top} !!\top \xrightarrow{!(\mathrm{m}^0)^{-1}} !\mathbf{1}$$

Checking that $(\mathbf{1}, \upsilon)$ is indeed a !-coalgebra boils down to a simple diagrammatic computation using the general axioms satisfied by the comonadic and monoidal structure of the ! functor.

A simple computation shows that $\upsilon_{*,n} = 1$ for all $n \in |!\mathbf{1}|$ (remember that $|!\mathbf{1}| = \mathbb{N}$).

Let $(X_i, h_i)_{i \in I}$ be a countable family of coalgebras. Then we can endow $X = \bigoplus_{i \in I} X_i$ with a structure of coalgebra $h \in \mathbf{Pcoh}(X, !X)$. By the universal property of the coproduct, it suffices to define for each $i \in I$ a morphism $h'_i : X_i \to !X$. We set $h'_i = !\mathrm{in}_i\, h_i$ where we record that $\mathrm{in}_i : X_i \to X$ is the ith canonical injection into the coproduct. It is then quite easy to check that (X, h) is a coalgebra using the fact that each (X_i, h_i) is a coalgebra.

Consider the case where $I = \mathbb{N}$, $X_i = \mathbf{1}$ and $h_i = \upsilon$ for each $i \in \mathbb{N}$. Then we use N to denote the corresponding object X and $h_{\mathbf{N}}$ for the corresponding coalgebra structure, $h_{\mathbf{N}} \in \mathbf{Pcoh}(\mathbf{N}, !\mathbf{N})$. Notice that N is the PCS introduced in Example 43. We use $\overline{n} \in \mathbf{Pcoh}(\mathbf{1}, \mathbf{N})$ for the nth injection that we consider also as the element of $\mathsf{P}\mathbf{N}$ defined by $\overline{n}_k = \delta_{n,k}$.

An easy computation shows that

$$(h_{\mathbf{N}})_{n,\mu} = \begin{cases} 1 & \text{if } \mu = k[n] \text{ for some } k \in \mathbb{N}, \\ 0 & \text{otherwise.} \end{cases}$$

Let $t \in \mathbf{Pcoh}_!(\mathbf{N}, X)$ for some object X of $\mathbf{Pcoh}$. Then $t\, h_{\mathbf{N}} \in \mathbf{Pcoh}(\mathbf{N}, X)$ is a linearized[7] version of t. Given $u \in \mathsf{P}\mathbf{N}$, an easy computation shows that

$$t\, h_{\mathbf{N}}\, u = \sum_{n=0}^{\infty} u_n \widehat{t}(\overline{n})\,.$$

7. This is not at all the same kind of linearization as the one introduced by Differential Linear Logic Ehrhard [2018].

The objects $\mathbf{N}$ and $\mathbf{1} \oplus \mathbf{N}$ are obviously isomorphic, through the morphisms $p \in \mathbf{Pcoh}(\mathbf{N}, \mathbf{1} \oplus \mathbf{N})$ and $s \in \mathbf{Pcoh}(\mathbf{1} \oplus \mathbf{N}, \mathbf{N})$ given by

$$p_{n,(1,*)} = s_{(1,*),n} = \delta_{n,0} \text{ and } p_{n,(2,n')} = s_{(2,n'),n} = \delta_{n,n'+1}.$$

We set $\overline{\mathsf{succ}} = s \,\mathrm{in}_2 \in \mathbf{Pcoh}(\mathbf{N}, \mathbf{N})$, so that $\overline{\mathsf{succ}}_{n,n'} = \delta_{n+1,n'}$ represents the successor function.

We can now define the morphim corresponding to the **conditional**.

Given an object X of **Pcoh**, we define a morphism

$$\overline{\mathsf{if}} \in \mathbf{Pcoh}(\mathbf{N} \otimes !X \otimes !(!\mathbf{N} \multimap X), X)\,.$$

For this, we define first $\overline{\mathsf{if}}_0 \in \mathbf{Pcoh}(\mathbf{1} \otimes !X \otimes !(!\mathbf{N} \multimap X), X)$ as the following composition of morphisms (without mentioning the isomorphisms associated with the monoidality of $\otimes$)

$$!X \otimes !(!\mathbf{N} \multimap X) \xrightarrow{!X \otimes \mathrm{w}} !X \xrightarrow{\mathrm{der}_X} X$$

and next $\overline{\mathsf{if}}_+ \in \mathbf{Pcoh}(\mathbf{N} \otimes !X \otimes !(!\mathbf{N} \multimap X), X)$ (with the same conventions as above)

$$\mathbf{N} \otimes !X \otimes !(!\mathbf{N} \multimap X) \xrightarrow{h_{\mathbf{N}} \otimes \mathrm{w} \otimes \mathrm{der}} !\mathbf{N} \otimes (!\mathbf{N} \multimap X) \xrightarrow{\mathrm{ev}\ \sigma} X$$

where σ is the isomorphism associated with the symmetry of the functor $\otimes$.

The universal property of $\oplus$ and the fact that $\_ \otimes Y$ is a left adjoint for each object Y allows therefore to define $\overline{\mathsf{if}}' \in \mathbf{Pcoh}((\mathbf{1} \oplus \mathbf{N}) \otimes !X \otimes !(!\mathbf{N} \multimap X), X)$. Finally, our conditional morphism is $\overline{\mathsf{if}} = \overline{\mathsf{if}}'\,(p \otimes !X \otimes !(!\mathbf{N} \multimap X)) \in \mathbf{Pcoh}(\mathbf{N} \otimes !X \otimes !(!\mathbf{N} \multimap X), X)$. The isomorphism $p \in \mathbf{Pcoh}(\mathbf{N}, \mathbf{1} \oplus \mathbf{N})$ is defined at the end of Section 3.1.4.

Notice that the two following diagrams commute

$$\begin{array}{ccc}
\mathbf{1} \otimes !X \otimes !(!\mathbf{N} \multimap X) & \xrightarrow{\overline{0} \otimes \mathrm{Id}} & \mathbf{N} \otimes !X \otimes !(!\mathbf{N} \multimap X) \\
& \searrow^{\mathrm{der} \otimes \mathrm{w}} & \downarrow{\scriptstyle \overline{\mathsf{if}}} \\
& & X
\end{array}$$

$$\begin{array}{ccc}
\mathbf{1} \otimes !X \otimes !(!\mathbf{N} \multimap X) & \xrightarrow{\overline{n+1} \otimes \mathrm{Id}} & \mathbf{N} \otimes !X \otimes !(!\mathbf{N} \multimap X) \\
{\scriptstyle \overline{n}^! \otimes \mathrm{w}} \downarrow & & \downarrow{\scriptstyle \overline{\mathsf{if}}} \\
!\mathbf{N} \otimes (!\mathbf{N} \multimap X) & \xrightarrow{\mathrm{ev}\ \sigma} & X
\end{array}$$

This second commutation boils down to the following simple property: $\forall n \in \mathbb{N}\ h_{\mathbf{N}}\,\overline{n} = \overline{n}^!$, meaning that $h_{\mathbf{N}}$ promote $\overline{n}$. Observe that it is not true however that $\forall u \in \mathsf{PN}\ h_{\mathbf{N}}\,u = u^!$. This means that $h_{\mathbf{N}}$ allows to duplicate and erase "true" natural numbers $\overline{n}$ but not general elements of PN which can be considered as "computations" and not as "values".

3.1.5 Type fixpoints

Linear Logic

All the operations of LL define monotone continuous functionals on $\mathbf{Pcoh}_{\subseteq}$[8] which moreover commute with the functor E. As a consequence, if $X_i \subseteq Y_i$ for $i = 1, 2$ then $X_1 \multimap X_2 \subseteq Y_1 \multimap Y_2$ (recall that $X_1^\perp \subseteq X_2^\perp$), and $\multimap$ commutes with directed colimits in $\mathbf{Pcoh}_{\subseteq}$.

Coalgebras

We will need to interpret fixpoint of coalgebras in the interpretation of Λ_{HP}.

This notion of inclusion on probabilistic coherence spaces extends to coalgebras (again, we refer to Ehrhard [2016] for more details) and we can define an order $P \subseteq Q$ in $\mathbf{Pcoh}^!_{\subseteq}$ where P and Q are objects of $\mathbf{Pcoh}^!$.

Notice that the colimit in $(\mathbf{Pcoh}^!)_{\subseteq}$ of a directed family of dense coalgebras is dense. We use $(\mathbf{Pcoh}^!_{\mathsf{den}})_{\subseteq}$ for the sub-class of $\mathbf{Pcoh}^!_{\subseteq}$ whose objects are the dense coalgebras (with the same order relation).

8. The order $\subseteq$ on PCSs has been defined in Paragraph 3.1.1.

3.2 An adequate model of pure probabilistic λ-calculus

This section presents results of Ehrhard et al. [2011]. The model of probabilistic coherent spaces was already known to be a sound model of a probabilistic extension of pure λ-calculus (see Danos and Ehrhard [2011]). We prove the Adequacy Theorem and give a computational meaning to this model: the probability that a term reduces to a head normal form is equal to its denotation computed on a suitable set of values.

3.2.1 Syntax and operational semantics of Λ^+

$$\Lambda^+ : M, N, \ldots := * \mid x \mid \lambda x\, M \mid (M)\, N \mid M +_p N,\ p \in [0,1] \cap \mathbb{Q}$$

(a) The Syntax of Λ^+

$$\frac{}{(\lambda x^\sigma\, M)\, N \xrightarrow{1} M\,[N/x]} \qquad \frac{}{M +_p N \xrightarrow{p} M} \qquad \frac{}{M +_p N \xrightarrow{1-p} N}$$

$$\frac{M \xrightarrow{p} M'}{\lambda x\, M \xrightarrow{p} \lambda x\, M'} \qquad \frac{M \xrightarrow{p} M'}{(M)\, N \xrightarrow{p} (M')\, N}\ M \text{ not abstraction}$$

(b) Probabilistc one-step head-reduction of $\xrightarrow{p}$

$$\mathsf{Red}_{M,M'} = \begin{cases} \sum_\pi p_\pi, & \text{where } \pi \text{ range over derivations of } M \xrightarrow{\mathtt{p}_\pi} M' \\ 1 & \text{if } M = M' \text{ is head-normal} \\ 0 & \text{otherwise.} \end{cases} \qquad \mathsf{Red}^\infty_{M,M'} = \bigvee_n \mathsf{Red}^n_{M,M'}$$

$$M \sim M', \text{ if for all closed term } C,\ \sum_{H \in \mathrm{hnf}} \mathsf{Red}^\infty_{(C)M,H} = \sum_{H \in \mathrm{hnf}} \mathsf{Red}^\infty_{(C)M',H}.$$

(c) "Transitive closure" of $\xrightarrow{p}$ and observational equivalence of closed terms.

Figure 3.3 – Syntax and operational semantics of Λ^+

The syntax of Λ^+, the pure probabilistic lambda-calculus is given in Figure 3.3a. The term $*$ is a constant, considered as a closed term. Although it is used in the proofs, the adequacy Theorem 64 still holds without having $*$ in the syntax. We will consider only the head-reduction, i.e. the small step operational semantics defined in Figure 3.3. The notation $M \xrightarrow{\mathtt{p}} N$ means that the term M reduces in one step to the term N with probability $p \in [0,1]$. As expected, the normal forms of this strategy are the *head normal forms*, i.e. the terms of the shape $\lambda x_1 \ldots x_\ell . M N_1 \ldots N_n$, with M either a variable or $*$. The set of head normal forms is denoted as hnf, the letter H will be ranged over hnf.

As in the preceding sections, we present the head-reduction as a Markov process over the set Λ^+, following Danos and Ehrhard [2011]. We consider the set Λ^+ as a set of states and the transition matrix $\mathrm{Red} \in [0,1]^{\Lambda^+ \times \Lambda^+}$ given in Figure 3.3c. Notice that, if M and N are fixed, then $\Pi_{M,N}$ has at most two elements, as for example in the case $M = N +_p N$ that flips a biased coin and goes on with M or N according to the outcome. Red is a stochastic matrix (i.e. for all terms M, $\sum_{N \in \Lambda^+} \mathrm{Red}_{M,N} = 1$), the value of $\mathrm{Red}_{M,N}$ intuitively describes the probability of evolving from the state M to the state N in one step. A term M is *absorbing* whenever $\mathrm{Red}_{M,M} = 1$: the absorbing states are those which are invariant under the transition matrix. Notice that the head normal forms are all absorbing, but there are absorbing terms which are not hnf, such as the ever-looping term $\mathbf{\Omega}$. The n-th power Red^n of the matrix Red is a stochastic matrix on Λ^+ (in case $n = 0$, we have the identity matrix on Λ^+). Intuitively, the value of $\mathrm{Red}^n_{M,N}$ is the probability of evolving from the state M to the state N in exactly n steps. Let $M \in \Lambda^+$ and H absorbing. $\mathrm{Red}^\infty_{M,H}$ is defined in Figure 3.3c as the lowest upper bound of the monotonic sequence $\{\mathrm{Red}^n_{M,H}\}_{n \in \mathbb{N}}$. Intuitively, $\mathrm{Red}^\infty_{M,H}$ defines the probability that M reaches a head normal form H in an arbitrary number of steps.

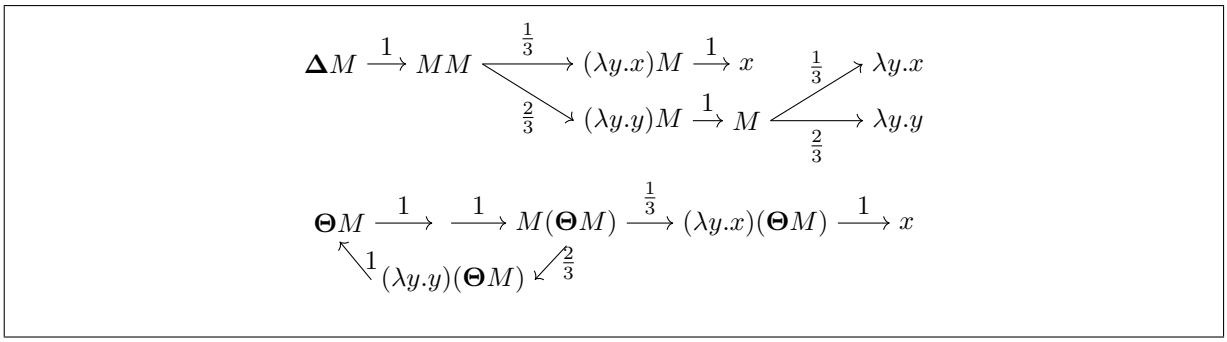


Figure 3.4 – Reduction trees of the terms $\mathbf{\Delta}M$ and $\mathbf{\Theta}M$, with $M = \lambda y.x +_{\frac{1}{3}} \lambda y.y$.

$$\begin{aligned}
[\![*]\!]^\Gamma(v) &= e_\star & [\![x]\!]^\Gamma(v) &= \pi_x(v)\\
[\![\lambda x.M]\!]^\Gamma(v) &= \lambda\big(u \mapsto [\![M]\!]^{x,\Gamma}(u :: v)\big) & [\![MN]\!]^\Gamma(v) &= \mathsf{app}\big([\![M]\!]^\Gamma(v)\big)([\![N]\!]^\Gamma(v))\\
[\![M +_p N]\!]^\Gamma(v) &= p[\![M]\!]^\Gamma(v) + (1-p)[\![N]\!]^\Gamma(v)
\end{aligned}$$

Figure 3.5 – Interpretation of a term in Λ^+ as an entire function from $\mathsf{P}(D^\Gamma)$ to $\mathsf{P}(D)$.

3.2.2 Examples

We use the following notations for terms useful to build examples:

$$\mathbf{\Theta} = (\lambda xy.y(xxy))(\lambda xy.y(xxy)), \qquad \mathbf{\Delta} = \lambda x.xx, \qquad \mathbf{\Omega} = \Delta\Delta.$$

Figure 3.4 gives two examples of reduction tree. The reduction is non-deterministic since there are two rules associated with the random constructor. Remark that $L +_p N$ intuitively expresses a superposition between L and N, rather than an uncertain knowledge whether the term is L or N. Figure 3.4, for example, shows that $\mathbf{\Delta}(\lambda y.x +_{\frac{1}{3}} \lambda y.y)$ reduces to $\lambda y.x$ (with probability $\frac{2}{9}$), in which case the random term $\lambda y.x +_{\frac{1}{3}} \lambda y.y$ behaves behaves sometimes as $\lambda y.y$ and sometimes as $\lambda y.x$.

We are interested in the probability that a given term reduces to a given head normal form after an arbitrary large (but finite) number of reduction steps. Computing such a probability is not trivial because of the presence of non-normalizing terms. For example, the probability that $\mathbf{\Theta}(\lambda y.x +_{\frac{1}{3}} \lambda y.y)$ reduces to x must be, intuitively, the limit of $\frac{1}{3}\sum_{n=0}^{\infty}\frac{2^n}{3^n} = 1$ (see Figure 3.4). In particular, recalling the first example, we have $\mathrm{Red}^n_{\mathbf{\Delta}M,x} = \frac{1}{3}$ if $n \geqslant 3$, otherwise it is 0. Thus, $\mathrm{Red}^\infty_{\mathbf{\Delta}M,x} = \frac{1}{3}$ and, as for the second example, $\mathrm{Red}^n_{\mathbf{\Theta}M,x} = \frac{1}{3}\sum_{i=0}^{k}\left(\frac{2}{3}\right)^i$, for $4(k+1) \leqslant n < 4(k+2)$. Hence, $\mathrm{Red}^\infty_{\mathbf{\Theta}M,x} = 1$.

3.2.3 Interpretation of Λ^+ in Pcoh

Recall from Proposition 50 that $\mathbf{Pcoh}_!$ has a reflexive object $(D, \lambda, \mathsf{app})$. We recall the notation introduced in its proof: if $r \in \mathsf{P}(D)$ and $u \in \mathsf{P}(D^{\mathbb{N}})$, $r :: u$ is the vector in $\mathsf{P}(D^{\mathbb{N}})$ defined by $\pi_0(r :: u) = r$ and $\pi_{n+1}(r :: u) = \pi_n(u)$.

The closed terms of Λ^+ are interpreted as vectors in $\mathsf{P}(D)$. In the general case, given a term M and a list Γ of pairwise different variables containing all the free variables of M, the interpretation of M is a morphism $[\![M]\!]^\Gamma \in \mathbf{Pcoh}_!(D^\Gamma, D)$, which can be seen as an entire function:

$$[\![M]\!]^\Gamma : \mathsf{P}(D^\Gamma) \to \mathsf{P}(D).$$

The definition of $[\![M]\!]^\Gamma$ is given in Figure 3.5, by structural induction on M. Using the notation of that figure, we recall that $\pi_x(v) \in \mathsf{P}(D)$ is the x-th component of $v \in \mathsf{P}(D^\Gamma)$, for $x \in \Gamma$. Also, recall the writing $u :: v$ denotes the vector in $\mathsf{P}(D^{x,\Gamma})$ whose x-th component is $u \in \mathsf{P}(D)$ and whose components in Γ are given by $v \in \mathsf{P}(D^\Gamma)$. Finally, $u \mapsto [\![M]\!]^{x,\Gamma}(u :: v)$ denotes the entire function mapping any $u \in \mathsf{P}(D)$ to $[\![M]\!]^{x,\Gamma}(u :: v) \in \mathsf{P}(D)$. We will simply write $[\![M]\!]$ in case M is a closed term.

Notice that $*$ is interpreted by the basis vector $e_\star$ in the direction of the empty multiset $\star \in |D|$, and $+_p$ is interpreted by the p-weighted sum. Apart from these, the interpretation follows the one determined by the categorical model of the pure λ-calculus given by the cartesian closed structure of the category $\mathbf{Pcoh}_!$ and the reflexive object $(D, \lambda, \mathsf{app})$. More precisely, $[\![x]\!]^\Gamma$ is the x-th projection of the product D^Γ, $[\![\lambda x.M]\!]^\Gamma(v) = \lambda \circ_! \mathsf{Cur}\left([\![M]\!]^{x,\Gamma}\right) \circ_! v$ and $[\![MN]\!]^\Gamma(v) = \mathsf{Ev} \circ_! \left(\mathsf{app} \circ_! \left([\![M]\!]^\Gamma \circ_! v\right), [\![N]\!]^\Gamma \circ_! v\right)$.

Next Lemma 55 precises the intuition about pairing and orthogonal. Recall that the interpretation of a closed term M is a vector $[\![M]\!]$ in $\mathsf{P}(D)$ and the interpretation of an environment intended as an infinite stack of terms is a vector u in $\mathsf{P}(D^{\mathbb{N}})$. Then, the interaction of a term and an environment is given by pairing and promotion: $\langle [\![M]\!], u^! \rangle$.

Lemma 55. *For every $v, r \in \mathsf{P}(D)$ and $u \in \mathsf{P}(D^{\mathbb{N}})$, $\langle \mathsf{app}(v)(r), u^! \rangle = \langle v, (r :: u)^! \rangle$.*

3.2.4 Soundness and Adequacy

The next proposition precises the invariance of the interpretation along the reduction.:

Proposition 56 (Soundness)**.** *For every term $M \in \Lambda^+$, and sequence $\Gamma \supseteq \mathrm{FV}(M)$,*

$$[\![M]\!]^\Gamma = \sum_{N \in \Lambda^+} \mathrm{Red}_{M,N} [\![N]\!]^\Gamma .$$

Proof. It is a standard structural induction on M. The case $M = (\lambda x.N)L$ is achieved by means of the substitution lemma $[\![N[L/x]]\!]^\Gamma(v) = [\![N]\!]^{x,\Gamma}([\![L]\!]^\Gamma(v), v)$, inferred also by a structural induction on N. □

Our main result (Adequacy Theorem 64) will give a computational meaning to the values of the vectors in $\mathsf{P}(D)$ associated with the indexes in $|D_2|$. We have $|D_1| = \{\star\}$ and

$$|D_2| = \{m_0 :: \cdots :: m_k :: \star \text{ s.t. } k \in \mathbb{N} \text{ and } \forall i \leqslant k,\ m_i = [\star, \ldots, \star] \in \mathcal{M}_{\mathrm{fin}}(|D_1|)\} .$$

Our goal is to achieve the Adequacy Theorem 64 stating: $\sum_{d \in |D_2|} [\![M]\!]_d = \sum_{H \in \mathrm{hnf}} \mathrm{Red}^\infty_{M,H}$, that is the probability that a closed term M reaches a head normal form is equal to the sum of the values of $[\![M]\!]$ on the points of $|D_2|$. We refer to Ehrhard et al. [2011] for a detailed proof of this result and give here a sketch of the reasoning.

The first inequality $\sum_{d \in |D_2|} [\![M]\!]_d \geqslant \sum_{H \in \mathrm{hnf}} \mathrm{Red}^\infty_{M,H}$ (Proposition 58) is an easy consequence of the invariance of the interpretation under head-reduction (Proposition 56) and of the next Lemma 57 that ensures that $\sum_{d \in |D_2|} [\![M]\!]_d = 1$ whenever M is a head normal form. Let us first notice that $\sum_{d \in |D_2|} [\![M]\!]_d$ estimates the behavior of M when applied to an infinite stack made of $*$ terms. We denote $e_{\vec{\star}}$ the vector of $\mathsf{P}(D^{\mathbb{N}})$ that embodied this stack defined as, $\forall i \in \mathbb{N}$, $\pi_i(e_{\vec{\star}}) = e_\star \in \mathsf{P}(D)$. It satisfies $e_{\vec{\star}} = e_\star :: e_{\vec{\star}}$, $(e_{\vec{\star}})^!$ is in $\mathsf{P}(D^\perp)$ and

$$\forall v \in \mathsf{P}(D),\ \sum_{d \in |D_2|} v_d = \langle v, (e_{\vec{\star}})^! \rangle \leqslant 1.$$

Lemma 57. *Let $M \in \Lambda_0^+$. If M is a head normal form, then $\sum_{d \in |D_2|} [\![M]\!]_d = 1$.*

Proof. In order to prove that $\langle [\![M]\!], (e_{\vec{\star}})^! \rangle = 1$, for a closed head normal form M, we use that M is of the shape $\lambda x_1 \ldots \lambda x_\ell . H M_1 \ldots M_m$, where H is either $*$ or a variable in $\{x_1, \ldots, x_\ell\}$. Since $e_{\vec{\star}} = e_\star :: e_{\vec{\star}}$, we can apply the equation of Lemma 55 from right-to-left and conclude by using the interpretation of Figure 3.5, the retraction property $\mathsf{app} \circ_! \lambda = \mathrm{Id}$, and the fact that $\mathsf{app}(e_\star)(u) = e_\star$ for any $u \in D$. □

Proposition 58. *Let $M \in \Lambda_0^+$. Then, $\sum_{d \in |D_2|} [\![M]\!]_d \geqslant \sum_{H \in \mathrm{hnf}} \mathrm{Red}^\infty_{M,H}$.*

Proof. The proof is made in two steps: first, we prove that $\sum_{d \in |D_2|} [\![M]\!]_d \geqslant \sum_{H \in \mathrm{hnf}} \mathrm{Red}^n_{M,H}$ by induction on n. Then, the result follows from $\sum_{H \in \mathrm{hnf}} \mathrm{Red}^\infty_{M,H} = \sup_{n=0}^\infty \big(\sum_{H \in \mathrm{hnf}} \mathrm{Red}^n_{M,H} \big)$ that holds since $\mathrm{Red}^n_{M,H}$ is an increasing positive sequence.The base case of the induction results from Lemma 57 and the induction case from the invariance of the semantics under reduction Proposition 56.

□

Then, we turn to the converse inequality, $\sum_{d \in |D_2|} [\![M]\!]_d \leqslant \sum_{H \in \mathrm{hnf}} \mathrm{Red}^\infty_{M,H}$. Its proof is by far more delicate. In fact it corresponds to a quantitative version of the sensibility of Scott's model with respect to the standard λ-calculus Hyland [1976]: a λ-term with no head normal form is interpreted by the bottom element of the model. The inequality will be proved using a notion of formal approximation relating the syntactical behavior of the closed terms in Λ^+ with their denotations in D.

The converse of the soundness Proposition 58 is a consequence of the adequacy Lemma 63. We adapt the technique of logical relations (see e.g. Plotkin [1977]; Reynolds) to our quantitive framework. The idea is to find a relation $\lhd$ between vectors and terms relating the values of the firsts to the computational behavior of the seconds. Basically, one extends the operation on PCSs defining D to an operation Φ

acting on the relations in $\mathsf{P}(D) \times \Lambda_0^+$ (Definition 59). Then, $\lhd$ is the result of the closure by Φ of the relation between v and M defined by $\sum_{d\in|D_2|} v_d \leqslant \sum_{H\in\mathrm{hnf}} \mathrm{Red}^\infty_{M,H}$. However, the operation Φ is not monotonic, hence finding its closure is not trivial. We then use a technique due to Pitts [1994], consisting in deriving $\lhd$ from a fixed point of a monotonic operation Ψ (Definition 60, Proposition 62) associated with Φ.

We will use this technique again for the Adequacy proof for $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$ (see §3.4.4).

Definition 59. For any relation $R \subseteq \mathsf{P}(D) \times \Lambda_0^+$, we define the relation $\Phi(R) \subseteq \mathsf{P}(D) \times \Lambda_0^+$ as follows:

$$\Phi(R) = \left\{ \begin{array}{ll} (v, M) \text{ s.t.} & \forall u \in \mathsf{P}(D^{\mathbb{N}}), \forall n \in \mathbb{N},\ \forall N_0, \dots, N_{n-1} \in \Lambda_0^+ \\ & \text{if } \forall i < n, (\pi_i(u), N_i) \in R \text{ and if } \forall i \geqslant n, \pi_i(u) = e_\star, \\ & \text{then } \langle v, u^! \rangle \leqslant \sum_{H\in\mathrm{hnf}} \mathrm{Red}^\infty_{(MN_0\dots N_{n-1}),H} \end{array} \right\}.$$

Let us underline the analogy between Φ and the operation $X \mapsto (!X^{\mathbb{N}})^\perp$ defining our reflexive object D. Indeed, $\mathsf{P}((!X^{\mathbb{N}})^\perp)$ is the set of those vectors $v \in (\mathbb{R}^+)^{|!X^{\mathbb{N}}|}$ such that for all $u \in (\mathbb{R}^+)^{|X^{\mathbb{N}}|}$, if for every $i \in \mathbb{N}$, $\pi_i(u) \in \mathsf{P}(X)$, then $\langle v, u^! \rangle \leqslant 1$.

By $e_{\vec{\star}} \in \mathsf{P}(D^{\mathbb{N}})$, $(v, M) \in \Phi(R)$ entails $\sum_{d\in|D_2|} v_d = \langle v, (e_{\vec{\star}})^! \rangle \leqslant \sum_{H\in\mathrm{hnf}} \mathrm{Red}^\infty_{M,H}$. Indeed, we will define a relation $\lhd$ such that $([\![M]\!], M) \in \lhd$ for any closed term $M \in \Lambda_0^+$. The relation $\lhd$ is in fact a fixed point for Φ.

Definition 60. Given a pair $(R^+, R^-) \in \left(\mathcal{P}(D \times \Lambda_0^+)\right)^2$, let $\Psi(R^+, R^-) = (\Phi(R^-), \Phi(R^+)) \in \left(\mathcal{P}(D \times \Lambda_0^+)\right)^2$. The order $\sqsubseteq$ on $\left(\mathcal{P}(D\times\Lambda_0^+)\right)^2$ is $(R_1, R_2) \sqsubseteq (R_3, R_4)$ iff $R_1 \subseteq R_3$ and $R_2 \supseteq R_4$. Clearly, $\sqsubseteq$ defines a complete lattice on $\left(\mathcal{P}(D\times\Lambda_0^+)\right)^2$. We denote by $\bigsqcap$ its greatest lower bound and we consider the greatest lower bound of the set of the pre-fixed points of Ψ: $(\lhd^+, \lhd^-) = \bigsqcap \{(R^+, R^-) \text{ s.t. } \Psi(R^+, R^-) \sqsubseteq (R^+, R^-)\}$

Lemma 61. *We have $\Psi(\lhd^+, \lhd^-) = (\lhd^+, \lhd^-)$. In particular, $\lhd^+ = \Phi(\lhd^-)$ and $\lhd^- = \Phi(\lhd^+)$.*

Proof. As $R \subseteq R'$ entails $\Phi(R') \subseteq \Phi(R)$, Ψ is monotone increasing with respect to $\sqsubseteq$. Hence, by Tarski's Theorem on fixed points Tarski [1955] the greatest lower bound of the set of the pre-fixed points of Ψ is the least fixed point of Ψ. In particular, by definition of Ψ, $\Phi(\lhd^+) = \lhd^-$ and $\Phi(\lhd^-) = \lhd^+$. □

Proposition 62 shows that actually $\lhd^+ = \lhd^-$, so this is a fixed point for Φ. The inclusion $\lhd^+ \subseteq \lhd^-$ follows easily from the previous lemma, while the proof of the converse $\lhd^- \subseteq \lhd^+$ uses the approximations of the vectors in $\mathsf{P}(D)$ given by the chain $\{D_\ell\}_{\ell\in\mathbb{N}}$ of which D is the limit.

Proposition 62. *We have $\lhd^+ = \lhd^-$, which is then a fixed point of Φ. From now on, we denote it simply by $\lhd$.*

We now state that $([\![M]\!], M) \in \lhd = \Phi(\lhd)$ for every closed term M. The proof, which is quite involved, is achieved by structural induction on M.

Lemma 63. *Let $M \in \Lambda^+$, $\Gamma = (y_0, \dots, y_{n-1}) \supseteq \mathrm{FV}(M)$, $(u_0, N_0), \dots, (u_{n-1}, N_{n-1}) \in \lhd$. We have:*

$$[\![M]\!]^\Gamma(u_0, \dots, u_{n-1}) \lhd M\{N_0/y_0, \dots, N_{n-1}/y_{n-1}\}.$$

From the Adequacy Lemma 63, one can easily derive a number of results giving a computational meaning to the probabilistic coherence spaces interpretation of terms.

Theorem 64. *For any $M \in \Lambda_0^+$, we have*

$$\sum_{d\in|D_2|} [\![M]\!]_d = \sum_{H\in\mathrm{hnf}} \mathrm{Red}^\infty_{M,H}.$$

Proof. Proposition 58 gives one inequality. For the converse, Lemma 63 gives $([\![M]\!], M) \in \lhd$, so by Proposition 62, $([\![M]\!], M) \in \Phi(\lhd)$. Hence, by definition of Φ, $\langle [\![M]\!], e_{\vec{\star}}^! \rangle \leqslant \sum_{H\in\mathrm{hnf}} \mathrm{Red}^\infty_{M,H}$. We conclude by recalling that $\langle [\![M]\!], e_{\vec{\star}}^! \rangle = \sum_{d\in|D_2|} [\![M]\!]_d$ (see Equation (3.2.4)). □

A corollary of the adequacy Theorem 64 is the soundness of the order $\leqslant$ on vectors (Equation (3.1)) with respect to the following operational pre-order $\preccurlyeq$ on terms.

Definition 65. For any terms M, N, we define $M \preccurlyeq N$ whenever for every context $C[\cdot]$,

$$\sum_{H\in\mathrm{hnf}} \mathrm{Red}^\infty_{C[M],H} \leqslant \sum_{H\in\mathrm{hnf}} \mathrm{Red}^\infty_{C[N],H}.$$

We prove by a structural induction on the contexts that the interpretation $[\![\cdot]\!]$ is context closed:

Lemma 66. *If $[\![M]\!]^\Gamma \leqslant [\![N]\!]^\Gamma$, then for every context $C[\cdot]$, $[\![C[M]]\!]^\Gamma \leqslant [\![C[N]]\!]^\Gamma$.*

This entails, thanks to the Adequacy Theorem 64, that

Corollary 67. *Let $M, N \in \Lambda^+$ and $\Gamma \supseteq \mathrm{FV}(M) \cup \mathrm{FV}(N)$. Then $[\![M]\!]^\Gamma \leqslant [\![N]\!]^\Gamma$ entails $M \preccurlyeq N$.*

3.3 A fully abstract model of pPCF

This section presents results from Ehrhard et al. [2018a]. We define an extension of the probabilistic PCF studied in Danos and Ehrhard [2011], with a `let` construction. More precisely, this construction is encoded from a more primitive one: $\mathsf{ifzm}(n, M, z \cdot N)$ that tests if n is equal to 0, then it goes on with M, else it goes on with N by passing the value $n-1$ to the variable z in N. This conditional reflects the coalgebraic structure of natural numbers described in Example 54. Actually, this coalgebraic structure was introduced in order to allow the `let` construction (see §3.3.2) which reflects a call-by-value handling of natural numbers. Moreover, it allows to encode the Las Vegas example (see §3.3.2) presented in the introduction of this chapter (see Page 61).

3.3.1 Syntax and operational semantics of pPCF

$$A, B \ldots := \mathcal{N} \mid A \Rightarrow B$$

(a) Types of pPCF

$$M, N, \ldots := \underline{n} \mid \mathsf{succ}(M) \mid x \mid \lambda x^A\, M \mid (M)\, N \mid \mathsf{fix}(M) \mid \mathsf{ifzm}(L, M, z \cdot N) \mid \mathsf{coin}(p), p \in [0,1] \cap \mathbb{Q}$$

(b) Terms of $\Lambda^p_{\mathcal{N}}$

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{n \in \mathbb{N}}{\Gamma \vdash \underline{n} : \mathcal{N}} \qquad \frac{\Gamma \vdash M : \mathcal{N}}{\Gamma \vdash \mathsf{succ}(M) : \mathcal{N}}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A\, M : A \Rightarrow B} \qquad \frac{\Gamma \vdash M : A \Rightarrow B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M)\, N : B} \qquad \frac{\Gamma \vdash M : A \Rightarrow A}{\Gamma \vdash \mathsf{fix}(M) : A}$$

$$\frac{\Gamma \vdash L : \mathcal{N} \qquad \Gamma \vdash M : A \qquad \Gamma, z : \mathcal{N} \vdash N : A}{\Gamma \vdash \mathsf{ifzm}(L, M, z \cdot N) : A} \qquad \frac{p \in [0,1] \cap \mathbb{Q}}{\Gamma \vdash \mathsf{coin}(p) : \mathcal{N}}$$

A typing context is a sequence $\Gamma = (x_1 : A_1, \ldots, x_n : A_n)$ where the x_i's are pairwise distinct variables.

(c) Typing system of pPCF

Figure 3.6 – Syntax of pPCF

The syntax of $\Lambda^p_{\mathcal{N}}$ of our probabilistic extension pPCF of PCF is given in Figure 3.6. Notice that there is only one ground type $\mathcal{N}$, the type of natural numbers. The booleans true and false are represented by $\underline{1}$ or $\underline{0}$. The probabilistic choice is encoded by the term $\mathsf{coin}(p)$ that reduces to $\underline{0}$ with probability p and to $\underline{1}$ with probability $1-p$.

The term $\mathsf{ifzm}(M, z, P \cdot R)$ deserves a comment. It relies on the fact that $\mathcal{N}$ is interpreted by $\mathbf{N}$ which is recursively defined as a coproduct $\mathbf{N} = \mathbf{1} \oplus \mathbf{N}$. The canonical coalgebra structure of the type $\mathcal{N}$ (see Example 54) allows to discard or duplicate a value of type $\mathcal{N}$. This explains why it can be passed to the term of type R through the variable z whose linear type, in a standard call-by-name translation of PCF into Linear Logic, would be $!\mathcal{N}$ and not $\mathcal{N}$. Indeed, the coalgebra structure is precisely a linear morphism $\mathbf{N} \multimap !\mathbf{N}$.

Proposition 68. *Let M be a term and Γ be a typing context. There is at most one type A such that $\Gamma \vdash M : A$.*

Lemma 69. *If $\Gamma, x : A \vdash M : B$ and $\Gamma \vdash N : A$, then $\Gamma \vdash M\,[N/x] : B$.*

The reduction rules of pPCF are given in Figure 3.7 p.79. Given two terms M, M' and a real number $p \in [0,1]$, we define $M \xrightarrow{p} M'$, meaning that M reduces in one step to M' with probability p, by the following deduction system.

$$\frac{}{(\lambda x^A\, M)\, N \to_{\mathsf{w}} M\,[N/x]} \qquad \frac{}{\mathsf{fix}(M) \to_{\mathsf{w}} (M)\,\mathsf{fix}(M)}$$

$$\frac{}{\mathsf{succ}(\underline{n}) \to_{\mathsf{w}} \underline{n+1}} \qquad \frac{}{\mathsf{ifzm}(\underline{0}, P, z \cdot R) \to_{\mathsf{w}} P} \qquad \frac{}{\mathsf{ifzm}(\underline{n+1}, P, z \cdot R) \to_{\mathsf{w}} R\,[\underline{n}/z]}$$

(a) Deterministic one-step reduction $\to_{\mathsf{w}}$

$$\frac{M \to_{\mathsf{w}} M'}{M \xrightarrow{1} M'} \qquad \frac{}{\mathsf{coin}(p) \xrightarrow{p} \underline{0}} \qquad \frac{}{\mathsf{coin}(p) \xrightarrow{1-p} \underline{1}}$$

(b) Probabilistic one-step reduction $\xrightarrow{p}$

$$E[\cdot] := (E[\,])\,M \mid \mathsf{ifzm}(E[\,], M, z \cdot N) \mid \mathsf{succ}(E[\,])$$

$$E[M] \xrightarrow{p} E[N], \text{ whenever } M \xrightarrow{p} N$$

(c) Evaluation contexts and context closure of reduction $\xrightarrow{p}$.

$$\mathsf{Red}_{M,M'} = \begin{cases} p & \text{if } M \xrightarrow{p} M' \\ 1 & \text{if } M \text{ is weak-normal and } M' = M \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathsf{Red}^n_{M,M'} = (\mathsf{Red} \cdot \mathsf{Red}^n)_{M,M'} = \sum_{M_0} \mathsf{Red}_{M,M_0}\,\mathsf{Red}^n_{M_0,M'}$$

$$\mathsf{Red}^\infty_{M,M'} = \bigvee_n \mathsf{Red}^n_{M,M'}$$

(d) Stochastic matrix of reduction

$M \sim M'$, if $\mathsf{Red}^\infty_{(C)M,\underline{0}} = \mathsf{Red}^\infty_{(C)M',\underline{0}}$ for all closed term C of type $A \Rightarrow \mathcal{N}$.

(e) Observational equivalence of closed terms.

Figure 3.7 – Operational semantics of pPCF

We define first a deterministic *weak* reduction relation $\to_{\mathsf{w}}$ (see Figure 3.7a) and then we define the probabilistic reduction relation $\xrightarrow{p}$, for $p \in [0,1] \cap \mathbb{Q}$ (see figure 3.7b). This reduction can be called *weak-head reduction* (or simply weak reduction) since it always reduces the leftmost outermost redex and never reduces redexes under abstractions. We say that M is *weak-normal* if there is no reduction $M \xrightarrow{p} M'$.

In order to define observational equivalence, we need to represent the probability of convergence of a term to a normal form. As in Danos and Ehrhard [2011], we consider the reduction as a discrete time Markov chain whose states are terms and stationary states are weak normal terms. We then define a stochastic matrix indexed by terms $\mathsf{Red} \in [0,1]^{\mathsf{pPCF} \times \mathsf{pPCF}}$ (see Figure 3.7d). Saying that Red is stochastic means that the coefficients of Red belong to $[0,1]$ and that, for any given term M, one has $\sum_{M'} \mathsf{Red}_{M,M'} = 1$ (actually there are at most two terms M' such that $\mathsf{Red}_{M,M'} \neq 0$).

For all $M, M' \in \mathsf{pPCF}$, if M' is weak-normal then the sequence $(\mathsf{Red}^n_{M,M'})_{n=1}^\infty$ is monotone and included in $[0,1]$, and therefore has a lowest upper bound that we denote as $\mathsf{Red}^\infty_{M,M'}$ which defines a sub-stochastic matrix (taking $\mathsf{Red}^\infty_{M,M'} = 0$ when M' is not weak-normal). When M' is weak-normal, the number $p = \mathsf{Red}^\infty_{M,M'}$ is the probability that M reduces to M' after a finite number of steps.

For two closed terms M_1, M_2 of type A, the observational equivalence is given in Figure 3.7e. For simplicity we consider only closed terms M_1 and M_2. We could also define an observational equivalence

on non closed terms, replacing the term C with a context $C[\cdot]$ which could bind free variables of the M_i's. This would not change the Full-Abstraction result. The choice of testing the probability of reducing to $\underline{0}$ in the definition of observational equivalence is arbitrary: we would get the same equivalence by replacing $\underline{0}$ by $\underline{n}$.

3.3.2 Examples

We give a series of terms written in pPCF which implement natural simple algorithms to illustrate the expressive power of the language. We explain intuitively the behavior of these programs, and one can also have a look at §3.3.3 where the denotational interpretations of these terms in **Pcoh** are given, presented as functions.
Given a type A, we set $\Omega_A = \mathsf{fix}(\lambda x^A\, x)$ so that $\vdash \Omega_A : A$, which is the ever-looping term of type A.

Arithmetics

The predecessor function, which is usually a basic construction of PCF, is defined as:

$$\mathsf{pred} = \lambda x^{\mathcal{N}}\, \mathsf{ifzm}(x, \underline{0}, z \cdot z)$$

it is clear then that $(\mathsf{pred})\,\underline{0} \to_{\mathsf{w}}{}^* \underline{0}$ and that $(\mathsf{pred})\,\underline{n+1} \to_{\mathsf{w}}{}^* \underline{n}$.

The addition function can be defined as:

$$\mathsf{add} = \lambda x^{\mathcal{N}}\, \mathsf{fix}(\lambda a^{\mathcal{N}\Rightarrow\mathcal{N}}\, \lambda y^{\mathcal{N}}\, \mathsf{ifzm}(y, x, z \cdot \mathsf{succ}((a)\,z)))$$

and it is easily checked that $\vdash \mathsf{add} : \mathcal{N} \Rightarrow \mathcal{N} \Rightarrow \mathcal{N}$.

The exponential function can be defined as follows and satisfies $(\mathsf{exp}_2)\,\underline{n} \to_{\mathsf{w}}{}^* \underline{2^n}$.

$$\mathsf{exp}_2 = \mathsf{fix}(\lambda e^{\mathcal{N}\Rightarrow\mathcal{N}}\, \lambda x^{\mathcal{N}}\, \mathsf{ifzm}(x, \mathbf{1}, z \cdot (\mathsf{add})\,(e)\,z\ (e)\,z))$$

In the same line, one defines a comparison function cmp

$$\mathsf{cmp} = \mathsf{fix}(\lambda c^{\mathcal{N}\Rightarrow\mathcal{N}\Rightarrow\mathcal{N}}\, \lambda x^{\mathcal{N}}\, \lambda y^{\mathcal{N}}\, \mathsf{ifzm}(x, \underline{0}, z \cdot \mathsf{ifzm}(y, \underline{1}, z' \cdot (c)\,z\,z')))$$

such that $(\mathsf{cmp})\,\underline{n}\,\underline{m}$ reduces to $\underline{0}$ if $n \leqslant m$ and to $\underline{1}$ otherwise.

The let construction

This version of pPCF, which is globally call-by-name, offers however the possibility of handling integers in a CBV way. For instance, we can set

$$\mathsf{let}\ x\ \mathsf{be}\ M\ \mathsf{in}\ N = \mathsf{ifzm}(M, N\,[\underline{0}/x]\,, z \cdot N\,[\mathsf{succ}(z)/x])$$

and this construction can be typed as:

$$\frac{\Gamma \vdash M : \mathcal{N} \qquad \Gamma, x : \mathcal{N} \vdash N : A}{\Gamma \vdash \mathsf{let}\ x\ \mathsf{be}\ M\ \mathsf{in}\ N : A}$$

One can also check that the following reduction inference holds

$$\frac{M \xrightarrow{p} M'}{\mathsf{let}\ x\ \mathsf{be}\ M\ \mathsf{in}\ N \xrightarrow{p} \mathsf{let}\ x\ \mathsf{be}\ M'\ \mathsf{in}\ N}$$

whereas *it is no true* that

$$\frac{M \xrightarrow{p} M'}{N\,[M/x] \xrightarrow{p} N\,[M'/x]}$$

(consider cases where x does not occur in N, or occurs twice...). We have of course

$$\frac{}{\mathsf{let}\ x\ \mathsf{be}\ \underline{n}\ \mathsf{in}\ N \to_{\mathsf{w}} N\,[\theta(n)/x]}$$

where $\theta(0) = \underline{0}$ and $\theta(n+1) = \mathsf{succ}(\underline{n})$ (which reduces to $\underline{n+1}$ in one deterministic step) by definition of this construction.

Random generators

Using these constructions, we can define a closed term unif_2 of type $\mathcal{N} \Rightarrow \mathcal{N}$ which, given an integer n, yields a uniform probability distribution on the integers $0, \dots, 2^n - 1$:

$$\mathsf{unif}_2 = \mathsf{fix}(\lambda u^{\mathcal{N}\Rightarrow\mathcal{N}}\, \lambda x^{\mathcal{N}}\, \mathsf{ifzm}(x, \underline{0}, z \cdot \mathsf{ifzm}(\mathsf{coin}(1/2), (u)\, z, z' \cdot (\mathsf{add})\, (\mathsf{exp}_2)\, z\, (u)\, z)))\,.$$

Observe that, when evaluating $(\mathsf{unif}_2)\, M$ (where $\vdash M : \mathcal{N}$), the term M is evaluated only once thanks to the CBV feature of the conditional construct. Indeed, we do not want the upper bound of the interval on which we produce a probability distribution to change during the computation (the result would be unpredictable!).

We use a rejection sampling, we can define a function unif which, given an integer n, yields a uniform probability distribution on the integers $0, \dots, n$:

$$\mathsf{unif} = \lambda x^{\mathcal{N}}\, \mathsf{let}\ y\ \mathsf{be}\ x\ \mathsf{in}\ \mathsf{fix}(\lambda u^{\mathcal{N}}\, \mathsf{let}\ z\ \mathsf{be}\ (\mathsf{unif}_2)\, y\ \mathsf{in}\ \mathsf{ifzm}((\mathsf{cmp})\, z\, y, z, w \cdot (u)\, y))$$

One checks easily that $\vdash \mathsf{unif} : \mathcal{N} \Rightarrow \mathcal{N}$. Given $n \in \mathbb{N}$, this function applies iteratively unif_2 until the result is $\leqslant n$. It is not hard to check that the resulting distribution is uniform (with probability $\frac{1}{n+1}$ for each possible result).

Last, let $n \in \mathbb{N}$ and let $\vec{p} = (p_0, \dots, p_n)$ be such that $p_i \in [0,1] \cap \mathbb{Q}$ and $p_0 + \cdots + p_n \leqslant 1$. Then one defines a closed term $\mathsf{ran}(\vec{p})$ which reduces to $\underline{i}$ with probability p_i for each $i \in \{0, \dots, n\}$. The definition is by induction on n.

$$\mathsf{ran}(p_0, \dots, p_n) = \begin{cases} \underline{0} & \text{if } p_0 = 1 \text{ whatever be the value of } n \\ \mathsf{ifzm}(\mathsf{coin}(p_0), \underline{0}, z \cdot \Omega^{\mathcal{N}}) & \text{if } n = 0 \\ \mathsf{ifzm}(\mathsf{coin}(p_0), \underline{0}, z \cdot \mathsf{succ}(\mathsf{ran}(\frac{p_1}{1-p_0}, \dots, \frac{p_n}{1-p_0}))) & \text{otherwise} \end{cases}$$

Observe indeed that in the first case we must have $p_1 = \cdots = p_n = 0$.

A simple Las Vegas program

We present the example that is the starting point of the coalgebra point of view. Indeed, we were stuck in encoding this basic probabilistic program, because we did not know how to duplicate the outcomes of a random variable without sampling it again. The introduction of the `let` construct solved this issue elegantly. This construct is available because natural numbers are interpreted by a coalgebra. Therefore, they can be discarded or duplicated.

Given a function $f : \mathbb{N} \to \mathbb{N}$ and $n \in \mathbb{N}$, such that for half of the $k \in \{0, \dots, n\}$, $(f)\, k = 0$ and for the other half $(f)\, k = 1$, find a $k \in \{0, \dots, n\}$ such that $f(k) = 0$.

This can be done by iterating random choices of k until we get a value such that $f(k) = 0$: this is probably the simplest example of a Las Vegas algorithm.

The following function does the job:

$$\mathsf{LV\,f\,n} = \mathsf{fix}(\lambda r^{\mathcal{N}}\, \mathsf{let}\ y\ \mathsf{be}\ (\mathsf{unif})\, \underline{n}\ \mathsf{in}\ \mathsf{ifzm}((f)\, y, y, z \cdot r))$$

with $\vdash \mathsf{LV\,f\,n} : (\mathcal{N} \Rightarrow \mathcal{N}) \Rightarrow \mathcal{N} \Rightarrow \mathcal{N}$. Our CBV integers are crucial here since without our version of the conditional, it would not be possible to get a random integer and use this value y both as an argument for f and as a result if the expected condition holds.

Let us describe the stochastic matrix associated to the reduction of $\mathsf{LV\,f\,n}$ (see its definition in Figure 3.7d) $\mathsf{Red}^1_{\mathsf{LV\,f\,n},\underline{k}} = \frac{1}{n+1}$.

If $(f)\, \underline{k} = 0$, then $\mathsf{Red}^1_{\mathsf{LV\,f\,n},\underline{k}} = \frac{1}{n+1}$. Yet, if $(f)\, \underline{k} = 1$, which happens with probability $\frac{1}{2}$, then $\mathsf{LV\,f\,n}$ reduces to itself but is not a weak normal form, so that: $\mathsf{Red}^1_{\mathsf{LV\,f\,n},\mathsf{LV\,f\,n}} = \frac{1}{2}$.

We iterate this process and get a recursive equation:

$$\begin{aligned} \forall j,\ \forall k \in \{0, \dots, n\} \text{ s.t. } (f)\, \underline{k} = 0,\ \mathsf{Red}^{j+1}_{\mathsf{LV\,f\,n},\underline{k}} &= \mathsf{Red}^1_{\mathsf{LV\,f\,n},\mathsf{LV\,f\,n}} \mathsf{Red}^j_{\mathsf{LV\,f\,n},\underline{k}} + \mathsf{Red}^1_{\mathsf{LV\,f\,n},\underline{k}} \mathsf{Red}^j_{\underline{k},\underline{k}} \\ &= \tfrac{1}{2} \mathsf{Red}^j_{\mathsf{LV\,f\,n},\underline{k}} + \tfrac{1}{n+1} \\ \mathsf{Red}^\infty_{\mathsf{LV\,f\,n},\underline{k}} &= \tfrac{1}{n+1} \sum_{j=0}^{\infty} \tfrac{1}{2^j} = \tfrac{2}{n+1}. \end{aligned}$$

Now, by summing over all the k such that $(f)\, \underline{k} = 0$, we get that $\mathsf{LV\,f\,n}$ will output the index of a cell that contains 0 with probability 1.

Similarly, we can compute that $\forall k \in \{0, \dots, n\}$ s.t. $(f)\, \underline{k} = 1$, $\mathsf{Red}^{j+1}_{\mathsf{LV\,f\,n},\underline{k}} = 0$.

3.3.3 Interpretation of pPCF in Pcoh

We first give the interpretation of types. Given a type A, we define an object $[A]$ of **Pcoh** as follows: $[\mathcal{N}] = \mathbf{N}$ and $[A \Rightarrow B] = [A] \Rightarrow [B]$.

Then we give the interpretation of terms. Given a context $\Gamma = (x_1 : A_1, \dots, x_k : A_k)$, a type A and a term M such that $\Gamma \vdash M : A$, we define a morphism $[\![M]\!]_\Gamma \in \mathbf{Pcoh}_!([\Gamma], [A])$ where $[\Gamma] = [A_1] \,\&\, \cdots \,\&\, [A_k]$. Equivalently, we can see $[\![M]\!]_\Gamma$ as a morphism in $\mathbf{Pcoh}([\![\Gamma]\!]^!, [A])$ where $[\![\Gamma]\!]^! = ![A_1] \otimes \cdots \otimes ![A_k]$. By Theorem 45, this morphism can be fully described as a function $\widehat{[\![M]\!]_\Gamma} : \prod_{i=1}^k \mathsf{P}[A_i] \to \mathsf{P}[A]$.

The interpretation of terms, given in Figure 3.8 are defined by induction on the typing derivation of $\Gamma \vdash M : A$, or, equivalently, on M.

- If $M = x_i$, then $[\![M]\!]_\Gamma = \mathrm{pr}_i$, that is $\widehat{[\![M]\!]_\Gamma}(u_1, \dots, u_k) = u_i$.
- If $M = \underline{n}$, then $[\![M]\!]_\Gamma = \overline{n} \circ B$ where B is the unique morphism in $\mathbf{Pcoh}_!([\Gamma], \top)$. That is $\widehat{[\![M]\!]_\Gamma}(\vec{u}) = \overline{n}$.
- If $M = \mathsf{coin}(p)$, for some $p \in [0,1] \cap \mathbb{Q}$, then $[\![M]\!]_\Gamma = p\overline{0} + (1-p)\overline{1}$.
- If $M = \mathsf{succ}(P)$ with $\Gamma \vdash P : \mathcal{N}$, then $[\![P]\!]_\Gamma \in \mathbf{Pcoh}_!([\Gamma], \mathbf{N})$. We set
$$\widehat{[\![M]\!]_\Gamma}(\vec{u}) = \sum_{n=0}^{\infty} (\widehat{[\![P]\!]_\Gamma}(\vec{u}))_n \overline{n+1}.$$
- If $M = \mathsf{ifzm}(P, Q, z \cdot R)$, $\Gamma \vdash P : \mathcal{N}$, $\Gamma \vdash Q : A$ and $\Gamma, z : \mathcal{N} \vdash R : A$ then by inductive hypothesis $[\![P]\!]_\Gamma \in \mathbf{Pcoh}_!([\Gamma], \mathbf{N})$, $[\![Q]\!]_\Gamma \in \mathbf{Pcoh}_!([\Gamma], [A])$ and $[\![R]\!]_{\Gamma, z:\mathcal{N}} \in \mathbf{Pcoh}_!([\Gamma] \otimes !\mathbf{N}, [A])$. We set
$$\widehat{[\![M]\!]_\Gamma}(\vec{u}) = (\widehat{[\![P]\!]_\Gamma}(\vec{u}))_0 \widehat{[\![Q]\!]_\Gamma}(\vec{u}) + \sum_{n=0}^{\infty} (\widehat{[\![P]\!]_\Gamma}(\vec{u}))_{n+1} \widehat{[\![R]\!]_{\Gamma, z:\mathcal{N}}}(\vec{u}, \overline{n})\,.$$
- If $M = (P)\,Q$ with $\Gamma \vdash P : A \Rightarrow B$ and $\Gamma \vdash Q : A$ then we have $[\![P]\!]_\Gamma \in \mathbf{Pcoh}_!([\Gamma], ![A] \multimap [B])$ and $[\![Q]\!]_\Gamma \in \mathbf{Pcoh}_!([\Gamma], [A])$. We set $\widehat{[\![M]\!]_\Gamma}(\vec{u}) = \widehat{[\![P]\!]_\Gamma}(\vec{u})(\widehat{[\![Q]\!]_\Gamma}(\vec{u}))$.
- If $M = \lambda x^A\, P$ with $\Gamma, x : A \vdash P : B$, then $[\![P]\!]_{\Gamma, x:A} \in \mathbf{Pcoh}_!([\Gamma] \otimes ![A], [B])$. We set $\widehat{[\![M]\!]_\Gamma}(\vec{u})(u) = \widehat{[\![P]\!]_{\Gamma, x:A}}(\vec{u}, u)$.
- If $M = \mathsf{fix}(P)$ with $\Gamma \vdash P : A \Rightarrow A$, then $[\![P]\!]_\Gamma \in \mathbf{Pcoh}_!([\Gamma], ![A] \multimap [A])$. For $u \in \mathsf{P}([\![A]\!])$, we define $f(u) = \widehat{[\![P]\!]_\Gamma}(\vec{u})(u)$. Then, we set $\widehat{[\![M]\!]_\Gamma}(\vec{u}) = \sup_{n \in \mathbb{N}} f^n(0)$.

Figure 3.8 – Interpretation of pPCF in **Pcoh**. The terms are supposed typed as in Figure 3.6c, and $\vec{u} \in [\![\Gamma]\!]$.

Examples

We refer to the various terms introduced in Section 3.3.2 and describe their interpretations as functions.

For instance, We have $\vdash \mathsf{pred} : \mathcal{N} \Rightarrow \mathcal{N}$ so $[\![\mathsf{pred}]\!] \in \mathsf{P}(\mathsf{N} \Rightarrow \mathsf{N})$, and one checks easily that:

$$\widehat{[\![\mathsf{pred}]\!]}(u) = (u_0 + u_1)\overline{0} + \sum_{n=1}^{\infty} u_{n+1}\overline{n}, \qquad \widehat{[\![\mathsf{add}]\!]}(u)(v) = \sum_{n=0}^{\infty} \Big(\sum_{i=0}^{n} u_i v_{n-i} \Big) \overline{n},$$

$$\widehat{[\![\mathsf{exp}_2]\!]}(u) = \sum_{n=0}^{\infty} u_n \overline{2^n}, \qquad \widehat{[\![\mathsf{cmp}]\!]}(u)(v) = \Big(\sum_{i \leqslant j} u_i v_j \Big) \overline{0} + \Big(\sum_{i > j} u_i v_j \Big) \overline{1}$$

$$[\![\mathsf{ran}(\vec{p})]\!] = \sum_{i=0}^{n} p_i \bar{i}, \qquad \widehat{[\![\mathsf{unif}]\!]}(u) = \sum_{n=0}^{\infty} \frac{u_n}{n+1} \Big(\sum_{i=0}^{n} \bar{i} \Big) = \sum_{i=0}^{\infty} \Big(\sum_{n=i}^{\infty} \frac{u_n}{n+1} \Big) \bar{i}$$

For the let construction $\mathsf{let}\ x\ \mathsf{be}\ P\ \mathsf{in}\ R$ where $\Gamma \vdash P : \mathcal{N}$ and $\Gamma, z : \mathcal{N} \vdash R : A$,

$$[\![\mathsf{let}\ x\ \mathsf{be}\ P\ \mathsf{in}\ R]\!]_\Gamma(\vec{u}) = \sum_{n=0}^{\infty} (\widehat{[\![P]\!]_\Gamma}(\vec{u}))_n \widehat{[\![R]\!]_{\Gamma, z:\mathcal{N}}}(\vec{u}, \overline{n})\,.$$

We postpone the computation of the semantics of our simple Las Vegas algorithm presented in page 84 as it will be an easy consequence of the Adequacy Lemma 76.

3.3.4 Soundness and Adequacy

We begin by the Substitution Lemma that relates syntactical substitution with semantical composition:

Lemma 70. *If* $\Gamma, x : A \vdash M : B$ *and* $\Gamma \vdash P : A$*, then* $[\![M\,[P/x]]\!]_\Gamma = [\![M]\!]_{\Gamma,x:A} \circ \langle \mathrm{Id}_{[\Gamma]}, [\![P]\!]_\Gamma \rangle$ *in* $\mathbf{Pcoh}_!$*. In other words, for any* $\vec{u} \in \mathsf{P}[\Gamma]$*, we have* $\widehat{[\![M\,[P/x]]\!]_\Gamma}(\vec{u}) = \widehat{[\![M]\!]_{\Gamma,x:A}}(\vec{u}, \widehat{[\![P]\!]_\Gamma}(\vec{u}))$*.*

The proof is a simple induction on M. The simplest way to write it is to use the functional characterization of the semantics.

Let Λ^A_Γ be the set of all terms M such that $\Gamma \vdash M : A$. In the case where Γ is empty, and so the elements of Λ^A_Γ are closed, we use Λ^A_0 to denote that set.

We formulate the **invariance of the interpretation** of terms under weak-reduction, using the stochastic reduction matrix introduced in §3.3.1. It results from a simple case analysis on the shape of M and by using the Substitution Lemma.

Theorem 71. *Let* $\Gamma \vdash M : A$*. Then,*

$$[\![M]\!]_\Gamma = \sum_{M' \in \Lambda^A_\Gamma} \mathsf{Red}_{M,M'} [\![M']\!]_\Gamma.$$

As it is often the case, the Adequacy Theorem 77 that relates denotational and observational equivalence is a consequence of the Adequacy Lemma 76, which relates the denotation of a close term $\vdash M : \mathcal{N}$ with its operational semantics:

$$\forall n \in \mathbb{N},\ \mathsf{Red}^\infty_{M,\underline{n}} = ([\![M]\!])_n.$$

The left-to-right inequality results from the invariance of the semantics:

Theorem 72. *Let* M *be such that* $\vdash M : \mathcal{N}$*. Then,*

$$\forall n \in \mathbb{N},\ \mathsf{Red}^\infty_{M,\underline{n}} \leqslant [\![M]\!]_n .$$

Proof. Iterating Theorem 71, we get $\forall k \in \mathbb{N},\ [\![M]\!] = \sum_{M' \in \Lambda^{\mathcal{N}}_0} \mathsf{Red}^k_{M,M'} [\![M']\!]$. Therefore, for all $k \in \mathbb{N}$, $[\![M]\!]_n \geqslant \mathsf{Red}^k_{M,\underline{n}}$ and the result follows, since $\underline{n}$ is weak-normal. □

For the converse inequality, we use a logical relation that encodes this inequality for ground types and that transposes it to higher type through the logical relation machinery, following the method introduced in Amadio and Curien [1998], simplifying the technique of Plotkin [1977].

For any type A we define a logical relation $\mathcal{R}^A \subseteq \Lambda^A_0 \times \mathsf{P}[A]$ by induction on types as follows:

- $M\ \mathcal{R}^{\mathcal{N}}\ u$ if $\forall n \in \mathbb{N}\ u_n \leqslant \mathsf{Red}^\infty_{M,\underline{n}}$
- $M\ \mathcal{R}^{A \Rightarrow B}\ t$ if $\forall P \in \Lambda^A_0\, \forall u \in \mathsf{P}[A]\ P\ \mathcal{R}^A\ u \Rightarrow (M)\, P\ \mathcal{R}^B\ \widehat{t}(u)$. Here, $t \in \mathsf{P}[A \Rightarrow B]$ and hence $\widehat{t} : \mathsf{P}[A] \to \mathsf{P}[B]$

The following Lemma 73, shows that the logical relation follows the definition of Red^∞ as the lowest upper bound of Red^n (see Figure 3.7). It is a consequence of Proposition 48 using an induction on types.

Lemma 73. *Let* $M \in \Lambda^\sigma_0$*. Then* $M\ \mathcal{R}^\sigma\ 0$*. Moreover, if* $(u(i))_{i \in \mathbb{N}}$ *is an increasing sequence in* $\mathsf{P}[\sigma]$ *such that* $\forall i \in \mathbb{N}\ M\ \mathcal{R}^\sigma\ u(i)$*, then* $M\ \mathcal{R}^\sigma\ \sup_{i \in \mathbb{N}} u(i)$*.*

The next lemma, proved by induction on types, describes the behavior of the logical relation with respect to antireduction.

Lemma 74. *Let* σ *be a type,* $M, M' \in \Lambda^\sigma_0$ *and* $u \in \mathsf{P}[\sigma]$*. Then*

$$M'\ \mathcal{R}^\sigma\ u \Rightarrow M\ \mathcal{R}^\sigma\ \mathsf{Red}_{M,M'} u .$$

The logical relation Lemma 75 is proved by induction on the typing derivation of a term M and by inspection of cases.

Lemma 75. *Assume that* $\Gamma \vdash M : \sigma$ *where* $\Gamma = (x_1 : \sigma_1, \dots, x_\ell : \sigma_\ell)$. *For all families* $(P_i)_{i=1}^\ell$ *and* $(u_i)_{i=1}^\ell$ *one has*

$$(\forall i\ P_i\ \mathcal{R}^{\sigma_i}\ u_i) \Rightarrow M\,[P_1/x_1, \dots, P_\ell/x_\ell]\ \mathcal{R}^\sigma\ [\![M]\!]_\Gamma(u_1, \dots, u_\ell)$$

A direct consequence of the definition of the logical relation at ground type and of this theorem is that if $\vdash M : \mathcal{N}$ we have $\forall n \in \mathbb{N}$, $\mathsf{Red}^\infty_{M,\underline{n}} \geqslant [\![M]\!]_n$. By Theorem 72 we have therefore the following operational interpretation of the semantics of ground type closed terms.

Theorem 76. *If* $\vdash M : \mathcal{N}$ *then, for all* $n \in \mathbb{N}$, $\mathsf{Red}^\infty_{M,\underline{n}} = [\![M]\!]_n$.

As a consequence, we get the semantics of the Las Vegas algorithm presented in Page 84

$$[\![\mathsf{LV\,f\,n}]\!]_k = \begin{cases} \frac{2}{n+1}, & \text{if } (f)\,k = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

As usual, the Adequacy Theorem 77 follows straightforwardly from Theorem 76. The observational equivalence relation on terms is defined in Figure 3.7e.

Theorem 77 (Adequacy)**.** *Let* M *and* M' *be closed terms of* pPCF *with same type.*
If $[\![M]\!] = [\![M']\!]$ *then* $M \sim M'$.

Proof. Let M and M' closed terms with type A. Assume that $[\![M]\!] = [\![M']\!]$. Let C be a closed term of type $A \Rightarrow \mathcal{N}$. Theorem 76 implies:

$$\mathsf{Red}^\infty_{(C)M,\underline{0}} = [\![(C)M]\!]_0 = (\widehat{[\![C]\!]}([\![M]\!]))_0 = (\widehat{[\![C]\!]}([\![M']\!]))_0 = [\![(C)M']\!]_0 = \mathsf{Red}^\infty_{(C)M',\underline{0}}\,.$$

□

3.3.5 Full Abstraction

We want to prove that **Pcoh** provides an equationally fully abstract model of pPCF, that is: two terms have the same denotation if and only if they are observationally equivalent. Thanks to Adequacy Theorem 77, we already proved the direct implication. This paragraph is devoted to the converse implication, that is: given two terms M and M' such that $\Gamma \vdash M : A$ and $\Gamma \vdash M' : A$, if $M \sim M'$ then $[\![M]\!]_\Gamma = [\![M']\!]_\Gamma$.

Let us first convey some intuitions about our approach to Full Abstraction. The first thing to say is that the usual method, which consists in proving that the model contains a collection of definable elements which is dense in a topological sense, does not apply here because definable elements are very sparse in **Pcoh**. For instance, in $\mathsf{P}[\mathcal{N} \Rightarrow \mathcal{N}]$, there is an element t which is characterized by $\widehat{t}(u) = 4u_0u_1\overline{0}$. We have $t \in \mathsf{P}[\mathcal{N} \Rightarrow \mathcal{N}]$ because, for any $u \in \mathsf{PN}$ we have $u_0 + u_1 \leqslant 1$ and hence $u_0u_1 \leqslant u_0(1-u_0) \leqslant 1/4$, and therefore $\widehat{t}(u) \in [0,1]$. Actually, t is not definable in pPCF and the best definable approximation of t is obtained by means of the term $\lambda x\,\mathtt{ifzm}(x, \mathtt{ifzm}(x, \Omega^{\mathcal{N}}, z' \cdot \mathtt{ifzm}(z', \underline{0}, z'' \cdot \Omega^{\mathcal{N}})), z \cdot \Omega^{\mathcal{N}})$ whose interpretation s satisfies $\widehat{s}(u) = 2u_0u_1\overline{0}$.

Our approach lies on a reasoning by contradiction. Let M and M' be terms (that we suppose closed for simplifying and without loss of generality) such that $\vdash M : A$ and $\vdash M' : A$. Assume that $[\![M]\!] \neq [\![M']\!]$. We have to prove that $M \not\sim M'$. Let us choose any point of the web, $a \in |[A]|$, which is an index where the matrices $[\![M]\!]$ and $[\![M]\!]'$ differ: $[\![M]\!]_a \neq [\![M']\!]_a$. This point a of the web encodes the structure of a context C such that $\vdash C : A \Rightarrow \mathcal{N}$ that will separate observationally the two terms M and M'. Indeed, we will have $[\![(C)\,M]\!]_0 \neq [\![(C)\,M']\!]_0$, so that by Adequacy Lemma 76, we get $\mathsf{Red}^\infty_{(C)M,\underline{0}} \neq \mathsf{Red}^\infty_{(C)M',\underline{0}}$ which contradicts observational equivalence (see Figure 3.7e).

Following Bucciarelli et al. [2011], in order to define C, *independently of* M *and* M', we associate with a a testing term a^- such that $\vdash a^- : \mathcal{N} \Rightarrow A \Rightarrow \mathcal{N}$ and which has the following essential property:

> There is an $n \in \mathbb{N}$ – depending only on a – such that, given $w, w' \in \mathsf{P}[A]$ such that $w_a \neq w'_a$ there are rational numbers $p_0, \dots, p_{n-1} \in [0,1]$ such that $\widehat{[\![a^-]\!]}(u)(w)_0 \neq \widehat{[\![a^-]\!]}(u)(w')_0$ where $u = p_0\overline{0} + \cdots + p_{n-1}\overline{n-1}$.

Applying this property to $w = [\![M]\!]$ and $w' = [\![M']\!]$, we obtain the required term F by setting $F = (a^-)\,\mathsf{ran}(p_0, \dots, p_{n-1})$.

In order to prove this crucial property of a^-, we consider the map $\varphi_w : u \mapsto \widehat{[\![a^-]\!]}(u)(w)_0$ which is an entire function depending only on the n first components $u_0, \dots, u_{n-1}$ of $u \in \mathsf{PN}$ (again, n is a non-negative integer which depends only on a).

We prove that the coefficient in φ_w of the monomial $u_0 u_1 \dots u_{n-1}$ is w_a.

It follows that the functions φ_w and $\varphi_{w'}$ are different, and therefore take different values on an argument of shape $p_0\overline{0} + \dots + p_{n-1}\overline{n-1}$ where all p_is are rational, because φ_w and $\varphi_{w'}$ are continuous functions.

Let us now give the exact statement of the theorems that are proved in full details in Ehrhard and Tasson [2018].

We first state a separation theorem which expresses that our testing terms a^-, when fed with suitable rational probability distributions, are able to separate any two distinct elements of the interpretation of a type.

Theorem 78 (Separation)**.** *Let A be a type and let $a \in |[A]|$. Let $w, w' \in \mathsf{P}[A]$ be such that $w_a \neq w'_a$. Let $n = |a|^-$. There is a sequence $(q_i)_{i=0}^{n-1}$ of rational numbers such that the element $u \in \mathsf{PN}$, defined by $(u)_i = q_i$ if $0 \leqslant i < n$ and $(u)_i = 0$ otherwise, satisfies $[\![a^-]\!](u)(w) \neq [\![a^-]\!](u)(w')$.*

We have now all the ingredients to prove the Full Abstraction theorem:

Theorem 79 (Full Abstraction)**.** *Let A be a type, Γ be a typing context and let M and M' be terms such that $\Gamma \vdash M : A$ and $\Gamma \vdash M' : A$. If $M \sim M'$. Then, $[\![M]\!]\Gamma = [\![M']\!]\Gamma$.*

Proof. Assume that $[\![M]\!]\Gamma \neq [\![M']\!]\Gamma$.

Let $(x_1 : A_1, \dots, x_k : A_k)$ be the typing context Γ. Let $N = \lambda {x_1}^{A_1} \cdots \lambda {x_k}^{A_k}\, M$ and $N' = \lambda {x_1}^{A_1} \cdots \lambda {x_k}^{A_k}\, M'$ be closures of M and M'. Let $\tau = A_1 \Rightarrow \cdots A_k \Rightarrow A$.

Let $w = [\![N]\!]$ and $w' = [\![N']\!]$. Since $w \neq w'$, there is $a \in |[\tau]|$ such that $w_a \neq w'_a$. By Separation Theorem 78, we can find a sequence $(q_i)_{i=0}^{n-1}$ of rational numbers such that for all $i \in \{0, \dots, n-1\}$ one has $q_i \geqslant 0$ and $\sum_{i=0}^{n-1} q_i \leqslant 1$. Then, $u = \sum_{i=0}^{n-1} q_i \mathsf{e}_i \in \mathsf{PN}$ satisfies $[\![a^-]\!](u)(w)_0 \neq [\![a^-]\!](u)(w')_0$.

Observe that $u = [\![\mathsf{ran}(q_0, \dots, q_{n-1})]\!]$.

Let C be the following term:

$$C = \left(a^-\right) \mathsf{ran}(q_0, \dots, q_{n-1})$$

which satisfies $\vdash C : A \Rightarrow \mathcal{N}$, $[\![(C)\, M]\!] = [\![a^-]\!](u)(w)$ and $[\![(C)\, M']\!] = [\![a^-]\!](u)(w')$.

Applying Theorem 76, we get

$$\mathsf{Red}(\mathcal{N})^{\infty}_{(C)M,\underline{0}} \neq \mathsf{Red}(\mathcal{N})^{\infty}_{(C)M',\underline{0}}$$

which shows that $M \not\sim M'$. □

3.4 A fully abstract model of pCBPV

This section presents results from Ehrhard and Tasson [2018]. In the preceding Section 3.3, we presented an extension of the probabilistic PCF of Danos and Ehrhard [2011]. Although it is globally call-by-name, we introduced a `let` constructor that allows to handle values of ground-type natural numbers as in a call-by-value. Now, we turn to $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$, a probabilistic extension of call-by-push-value described in Figure 3.9. We generalize the call-by-value strategy from natural numbers to all ground-types that are *values*. This notion of value is defined syntactically in Figure 3.9a and corresponds, in the semantics to coalgebras described in §3.1.4. We decompose the language pPCF of the preceding paragraph with respect to the value type and the general types, and get a language which is closed to polarized Linear Logic. Then, in §3.4.2, we encode the constructions of pPCF (among others) in $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$.

The operational semantics (see Figure 3.10f) and the Soundness Theorem 81 are straightforward generalizations. However, the Adequacy Theorem 84 for $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$ is more involved and implies the Adequacy Theorem 77 for pPCF. Indeed, the terms of pPCF can be encoded in our new language. The Full-abstraction Theorem 88 for $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$ holds and is proved along the same sketch of proof as the Full-abstraction Theorem 79 for pPCF. Yet, it necessitates to use the dense property that we introduced for coalgebras on Page 71.

3.4.1 Syntax and operational semantics of $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$

We introduce in Figure 3.9 the syntax of $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$ of call-by-push-value (where HP stands for "half polarized").

There are two kinds of types: positive types and general types which are defined by mutual induction. The type $\mathbf{1}$ is the neutral element of $\otimes$. Observe also that there are no restriction on the variance of types

$$\begin{array}{ll} \text{Positive types} & \varphi, \psi, \ldots := \mathbf{1} \mid !\sigma \mid \varphi \otimes \psi \mid \varphi \oplus \psi \mid \zeta \mid \mathsf{Rec}\, \zeta \cdot \varphi \\ \text{General types} & \sigma, \tau, \ldots := \varphi \mid \varphi \multimap \sigma \end{array}$$

(a) Types of $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$

$$\begin{aligned} M, N, \ldots := {} & x \mid () \mid M^! \mid \mathsf{der}\, M \mid (M, N) \mid \mathsf{in}_\ell M \mid \mathsf{in}_r M \mid \lambda x^\varphi M \mid \langle M \rangle N \mid \mathsf{fix}\, x^{!\sigma} M \\ & \mid \mathsf{case}(M, x_\ell \cdot N_\ell, x_r \cdot N_r) \mid \mathsf{pr}_\ell M \mid \mathsf{pr}_r M \mid \mathsf{coin}(p),\ p \in [0,1] \cap \mathbb{Q} \\ & \mid \mathsf{fold}(M) \mid \mathsf{unfold}(M) \end{aligned}$$

(b) Terms of $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$

$$\frac{}{\mathcal{P} \vdash () : \mathbf{1}} \qquad \frac{}{\mathcal{P}, x : \varphi \vdash x : \varphi} \qquad \frac{\mathcal{P} \vdash M : \sigma}{\mathcal{P} \vdash M^! : !\sigma} \qquad \frac{\mathcal{P} \vdash M : !\sigma}{\mathcal{P} \vdash \mathsf{der}\, M : \sigma}$$

$$\frac{\mathcal{P}, x : \varphi \vdash M : \sigma}{\mathcal{P} \vdash \lambda x^\varphi M : \varphi \multimap \sigma} \qquad \frac{\mathcal{P} \vdash M : \varphi \multimap \sigma \qquad \mathcal{P} \vdash N : \varphi}{\mathcal{P} \vdash \langle M \rangle N : \sigma} \qquad \frac{\mathcal{P}, x : !\sigma \vdash M : \sigma}{\mathcal{P} \vdash \mathsf{fix}\, x^{!\sigma} M : \sigma}$$

$$\frac{\mathcal{P} \vdash M : \varphi_\ell \otimes \varphi_r \quad i \in \{\ell, r\}}{\mathcal{P} \vdash \mathsf{pr}_i M : \varphi_i} \qquad \frac{\mathcal{P} \vdash M_\ell : \varphi_\ell \qquad \mathcal{P} \vdash M_r : \varphi_r}{\mathcal{P} \vdash (M_\ell, M_r) : \varphi_\ell \otimes \varphi_r}$$

$$\frac{\mathcal{P} \vdash M : \varphi_i \quad i \in \{\ell, r\}}{\mathcal{P} \vdash \mathsf{in}_i M : \varphi_\ell \oplus \varphi_r} \qquad \frac{}{\mathcal{P} \vdash \mathsf{coin}(p) : \mathbf{1} \oplus \mathbf{1}}$$

$$\frac{\mathcal{P} \vdash M : \varphi_\ell \oplus \varphi_r \qquad \mathcal{P}, x_\ell : \varphi_\ell \vdash M_\ell : \sigma \qquad \mathcal{P}, x_r : \varphi_r \vdash M_r : \sigma}{\mathcal{P} \vdash \mathsf{case}(M, x_\ell \cdot M_\ell, x_r \cdot M_r) : \sigma}$$

$$\frac{\mathcal{P} \vdash M : \psi\,[\mathsf{Rec}\, \zeta \cdot \psi / \zeta]}{\mathcal{P} \vdash \mathsf{fold}(M) : \mathsf{Rec}\, \zeta \cdot \psi} \qquad \frac{\mathcal{P} \vdash M : \mathsf{Rec}\, \zeta \cdot \psi}{\mathcal{P} \vdash \mathsf{unfold}(M) : \psi\,[\mathsf{Rec}\, \zeta \cdot \psi / \zeta]}$$

A typing context is an expression $\mathcal{P} = (x_1 : \varphi_1, \ldots, x_k : \varphi_k)$ where all types are positive and the x_is are pairwise distinct variables.

(c) Typing system of $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$.

Figure 3.9 – Syntax of $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$.

in the recursive type construction: for instance, $\mathsf{Rec}\, \zeta \cdot \varphi$ is a well-formed positive type if $\varphi = !(\zeta \multimap \zeta)$, where ζ has a negative and a positive occurrence. Notice that our *positive types* are positive in the sense of logical polarities, and *not* of the variance of type variables!

Values are particular $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$ terms (they are not a new syntactic category) defined in Figure 3.10a. It is easy to check that they are all typed with positive types.

Figure 3.10 defines a deterministic *weak* reduction relation $\to_{\mathsf{w}}$ and a probabilistic reduction relation $\xrightarrow{p}$. This reduction is weak in the sense that we never reduce within a "box" $M^!$ or under a λ.

The distinguishing feature of this reduction system is the role played by values in the definition of $\to_{\mathsf{w}}$. Consider for instance the case of the term $\mathsf{pr}_\ell\,(M_\ell, M_r)$; one might expect this term to reduce directly to M_ℓ but this is not the case. One needs first to reduce M_ℓ *and* M_r to values before extracting the first component of the pair (the terms $\mathsf{pr}_\ell\,(M_\ell, M_r)$ and M_ℓ have not the same denotational interpretation in general). Of course, replacing M_i with $M_i^!$ allows a lazy behavior. Similarly, in the $\to_{\mathsf{w}}$ rule for case, the term on which the test is made must be reduced to a value (necessarily of shape $\mathsf{in}_\ell V$ or $\mathsf{in}_r V$ if the expression is well typed) before the reduction is performed. This allows to "memoize" the value V for further usage: the value is passed to the relevant branch of the case through the variable x_i.

Given two terms M, M' and a real number $p \in [0,1]$, $M \xrightarrow{p} M'$ means that M reduces in one step to M' with probability p.

We say that M is *weak normal* if there is no p and no reduction $M \xrightarrow{p} M'$. It is clear that any value is weak normal. When M is closed, M is weak normal iff it is a value or an abstraction.

$$V, W, \ldots := x \mid () \mid M^! \mid (V, W) \mid \mathsf{in}_\ell V \mid \mathsf{in}_r V \mid \mathsf{fold}(V)\,.$$

(a) Values of $\Lambda^{\mathsf{P}}_{\mathsf{HP}}$

$$\frac{}{\mathsf{der}\, M^! \to_{\mathsf{w}} M} \qquad \frac{}{\langle \lambda x^{\varphi}\, M\rangle V \to_{\mathsf{w}} M\,[V/x]} \qquad \frac{}{\mathsf{pr}_i\,(V_\ell, V_r) \to_{\mathsf{w}} V_i}\ {\scriptstyle i \in \{\ell, r\}}$$

$$\frac{}{\mathsf{fix}\, x^{!\sigma}\, M \to_{\mathsf{w}} M\,\left[(\mathsf{fix}\, x^{!\sigma}\, M)^!/x\right]} \qquad \frac{}{\mathsf{case}(\mathsf{in}_i V, x_\ell \cdot M_\ell, x_r \cdot M_r) \to_{\mathsf{w}} M_i\,[V/x_i]}\ {\scriptstyle i \in \{\ell, r\}}$$

$$\frac{}{\mathsf{unfold}(\mathsf{fold}(V)) \to_{\mathsf{w}} V}$$

(b) Deterministic one-step reduction $\to_{\mathsf{w}}$

$$\frac{M \to_{\mathsf{w}} M'}{M \xrightarrow{1} M'} \qquad \frac{}{\mathsf{coin}(p) \xrightarrow{p} \mathsf{in}_\ell()} \qquad \frac{}{\mathsf{coin}(p) \xrightarrow{1-p} \mathsf{in}_r()}$$

(c) Probabilistic one-step reduction $\xrightarrow{p}$

$$E[\cdot] := \mathsf{der}\, E[\,] \mid \langle E[\,]\rangle V \mid \langle M\rangle E[\,] \mid \ \mid \mathsf{pr}_\ell E[\,] \mid \mathsf{pr}_r E[\,] \mid (E[\,], M_r) \mid (V, E[\,])$$
$$\mid \mathsf{in}_\ell E[\,] \mid \mathsf{in}_r E[\,] \mid \mathsf{case}(E[\,], x_\ell \cdot M_\ell, x_r \cdot M_r) \mid \mathsf{fold}(E[\,]) \mid \mathsf{unfold}(E[\,])$$

$$E[M] \xrightarrow{p} E[N], \text{ whenever } M \xrightarrow{p} N$$

(d) Evaluation contexts and context closure of reduction $\xrightarrow{p}$.

$$\mathsf{Red}_{M,M'} = \begin{cases} p & \text{if } M \xrightarrow{p} M' \\ 1 & \text{if } M \text{ is weak-normal and } M' = M \\ 0 & \text{otherwise.} \end{cases}$$
$$\mathsf{Red}^n_{M,M'} = (\mathsf{Red} \cdot \mathsf{Red}^n)_{M,M'} = \sum_{M_0} \mathsf{Red}_{M,M_0} \mathsf{Red}^n_{M_0,M'}$$
$$\mathsf{Red}^\infty_{M,M'} = \bigvee_n \mathsf{Red}^n_{M,M'}$$

(e) Stochastic matrix of reduction

$M \sim M'$, if $\mathsf{Red}^\infty_{\langle C\rangle M^!,()} = \mathsf{Red}^\infty_{\langle C\rangle M'^!,()}$ for all closed term C of type $!\sigma \multimap \mathbf{1}$.

(f) Observational equivalence of closed terms

Figure 3.10 – Operational semantics of $\Lambda^{\mathsf{P}}_{\mathsf{HP}}$

In order to simplify the presentation, we *choose* in Figure 3.10 a reduction strategy. For instance, we decide that, for reducing (M_ℓ, M_r) to a value, one needs first to reduce M_ℓ to a value, and then M_r; this choice is of course completely arbitrary. A similar choice is made for reducing terms of shape $\langle M\rangle N$, where we require the argument to be reduced first.

As for pPCF, in Section 3.3.1 we consider the reduction as a discrete time Markov chain whose states are terms and stationary states are weak normal terms. We define a stochastic matrix $\mathsf{Red} \in [0,1]^{\Lambda^{\mathsf{P}}_{\mathsf{HP}} \times \Lambda^{\mathsf{P}}_{\mathsf{HP}}}$ in 3.10e representing the probability to reduce in one step and get by iteration the substochastic matrix $\mathsf{Red}^\infty \in [0,1]^{\Lambda^{\mathsf{P}}_{\mathsf{HP}} \times \Lambda^{\mathsf{P}}_{\mathsf{HP}}}$ representing the probability to reduce to a weak normal term in finitely many steps.

For two closed terms M, M' of type A, the observational equivalence is given in Figure 3.10e.

3.4.2 Examples

In this section, we show how to encode usual types and terms in programming language theory. Noteworthy, in the example of natural numbers, we encode the constructions of pPCF, the language that we described in Section 3.3. Thanks to recursive types, we also encode `Lists` and `Streams`.

For the sake of readability, we drop the fold/unfold constructs associated with recursive types definitions; they can easily be inserted at the right places.

Ever-looping program

Given any type σ, we define $\Omega^\sigma = \mathsf{fix}\, x^{!\sigma}\, \mathsf{der}\, x$ which satisfies $\vdash \Omega^\sigma : \sigma$. Then, $\Omega^\sigma \to_{\mathsf{w}} \mathsf{der}\,(\Omega^\sigma)^! \to_{\mathsf{w}} \Omega^\sigma$. Thus, we can consider Ω^σ as the ever-looping program of type σ.

Booleans

Let type $\mathcal{B} = \mathbf{1} \oplus \mathbf{1}$. Then, $\mathcal{P} \vdash \mathsf{coin}(p) : \mathcal{B}$. We define the "true" constant as $\mathbf{t} = \mathsf{in}_\ell()$ and the "false" constant as $\mathbf{f} = \mathsf{in}_r()$. The corresponding eliminator is defined as follows. Given terms M, N_ℓ and N_r, we set $\mathsf{if}(M, N_\ell, N_r) = \mathsf{case}(M, x_\ell \cdot N_\ell, x_r \cdot N_r)$ where x_i is not free in N_i for $i \in \{\ell, r\}$. Then,

$$\frac{\mathcal{P} \vdash M : \mathcal{B} \qquad \mathcal{P} \vdash N_\ell : \sigma \qquad \mathcal{P} \vdash N_r : \sigma}{\mathcal{P} \vdash \mathsf{if}(M, N_\ell, N_r) : \sigma}\,.$$

We have the following weak and probabilistic reduction rules, derived from Figure 3.10:

$$\frac{}{\mathsf{if}(\mathbf{t}, N_\ell, N_r) \to_{\mathsf{w}} N_\ell} \qquad \frac{}{\mathsf{if}(\mathbf{f}, N_\ell, N_r) \to_{\mathsf{w}} N_r} \qquad \frac{M \xrightarrow{p} M'}{\mathsf{if}(M, N_\ell, N_r) \xrightarrow{p} \mathsf{if}(M', N_\ell, N_r)}\,.$$

Natural numbers

We recover natural numbers constructions of pPCF presented in §3.1. They will have the same operational and denotational semantics.

We define the type $\mathcal{N}$ of unary natural numbers by $\mathcal{N} = \mathbf{1} \oplus \mathcal{N}$ (i.e. $\mathcal{N} = \mathsf{Rec}\,\zeta \cdot (\mathbf{1} \oplus \zeta)$). We define $\underline{0} = \mathsf{in}_\ell()$ and $\underline{n+1} = \mathsf{in}_r \underline{n}$ so that we have $\mathcal{P} \vdash \underline{n} : \mathcal{N}$ for each $n \in \mathbb{N}$.

Then, given a term M, we define the term $\mathsf{succ}(M) = \mathsf{in}_r M$, so that

$$\frac{\mathcal{P} \vdash M : \mathcal{N}}{\mathcal{P} \vdash \mathsf{succ}(M) : \mathcal{N}}\,.$$

Finally, given terms M, N_ℓ and N_r and a variable x not free in N_ℓ, we define an "ifz" conditional by $\mathsf{ifzm}(M, N_\ell, x \cdot N_r) = \mathsf{case}(M, z \cdot N_\ell, x \cdot N_r)$, so that

$$\frac{\mathcal{P} \vdash M : \mathcal{N} \qquad \mathcal{P} \vdash N_\ell : \sigma \qquad \mathcal{P}, x : \mathcal{N} \vdash N_r : \sigma}{\mathcal{P} \vdash \mathsf{ifzm}(M, N_\ell, x \cdot N_r) : \sigma}$$

We have the following weak and probabilistic reduction rules, derived from Figure 3.10:

$$\frac{}{\mathsf{ifzm}(\mathsf{in}_i V, M_\ell, x \cdot M_r) \to_{\mathsf{w}} M_i\,[V/x]}\; i \in \{\ell, r\} \qquad \frac{M \xrightarrow{p} M'}{\mathsf{ifzm}(M, N_\ell, x \cdot N_r) \xrightarrow{p} \mathsf{ifzm}(M', N_\ell, x \cdot N_r)}$$

These conditionals will be used in the examples below.

Streams

Let φ be a positive type and $\mathcal{S}_\varphi$ be the positive type defined by $\mathcal{S}_\varphi = !(\varphi \otimes \mathcal{S}_\varphi)$, that is $\mathcal{S}_\varphi = \mathsf{Rec}\,\zeta \cdot !(\varphi \otimes \zeta)$. We can define a term M such that $\vdash M : \mathcal{S}_\varphi \multimap \mathcal{N} \multimap \varphi$ which computes the n-th element of a stream:

$$M = \mathsf{fix}\, f^{!(\mathcal{S}_\varphi \multimap \mathcal{N} \multimap \varphi)}\, \lambda x^{\mathcal{S}_\varphi}\, \lambda y^{\mathcal{N}}\, \mathsf{ifzm}(y, \mathsf{pr}_\ell(\mathsf{der}\, x), z \cdot \langle \mathsf{der}\, f \rangle \mathsf{pr}_r(\mathsf{der}\, x)\, z).$$

Let $O = (\Omega^{\varphi \otimes \mathcal{S}_\varphi})^!$ be a term which represents the "undefined stream". More precisely, it is a stream which is a value, but which contains nothing. It should not be confused with $\Omega^{\mathcal{S}_\varphi}$ which has the same type but which is not a value. We have $\vdash O : \mathcal{S}_\varphi$, and observe that the reduction of $\langle M \rangle O$ converges (to an abstraction) and that $\langle M \rangle O\, \underline{0}$ diverges.

Conversely, we can define a term N such that $\vdash N : !(\mathcal{N} \multimap \varphi) \multimap \mathcal{S}_\varphi$ which turns a function into the stream of its successive applications to an integer.

$$N = \mathsf{fix}\, F^{!(!(\mathcal{N}\multimap\varphi)\multimap\mathcal{S}_\varphi)}\, \lambda f^{!(\mathcal{N}\multimap\varphi)}\, \left(\langle \mathsf{der}\, f\rangle \underline{0}, \langle \mathsf{der}\, F\rangle(\lambda x^{\mathcal{N}}\, \langle \mathsf{der}\, f\rangle \mathsf{succ}(x))^!\right)^! .$$

Observe that the recursive call of F is encapsulated into a box, which makes the construction lazy. As a last example, consider the term P such that $\vdash P : (\mathcal{S}_\varphi \otimes \mathcal{S}_\varphi) \multimap (\mathcal{N} \oplus \mathcal{N}) \multimap \varphi$ is given by

$$P = \lambda y^{\mathcal{S}_\varphi \otimes \mathcal{S}_\varphi}\, \lambda c^{\mathcal{N}\oplus\mathcal{N}}\, \mathsf{case}(c, x \cdot \langle M\rangle x\, \mathsf{pr}_\ell y, x \cdot \langle M\rangle x\, \mathsf{pr}_r y).$$

Take $\varphi = \mathbf{1}$ and consider the term $Q = \Big(((), O)^!, O\Big)$. Then, we have $\vdash Q : \mathcal{S}_{\mathbf{1}} \otimes \mathcal{S}_{\mathbf{1}}$, and observe that $\langle P\rangle Q\, \mathsf{in}_\ell \underline{0}$ converges to $()$ whereas $\langle P\rangle Q\, \mathsf{in}_r \underline{0}$ diverges.

These examples suggest that $\mathcal{S}_\varphi$ behaves as should behave a type of streams of elements of type φ.

Lists

There are various possibilities for defining a type of lists of elements of a positive type φ. The simplest definition is $\mathcal{L}_0 = \mathbf{1} \oplus (\varphi \otimes \mathcal{L}_0)$. This corresponds to the ordinary ML type of lists. But we can also define $\mathcal{L}_1 = \mathbf{1} \oplus !(\varphi \otimes \mathcal{L}_1)$ and then we have a type of lazy lists (or terminable streams) where the tail of the list is computed only when required. Here is an example of a term L such that $\vdash L : \mathcal{L}_1$, with $\varphi = \mathcal{B} = \mathbf{1} \oplus \mathbf{1}$ which is a list of random length containing random booleans:

$$L = \mathsf{fix}\, x^{!\mathcal{L}_1}\, \mathsf{if}(\mathsf{coin}(1/4), \mathsf{in}_\ell(), \mathsf{in}_r(\mathsf{coin}(1/2) \otimes \mathsf{der}\, x)^!).$$

Then, L will reduce with probability $\frac{1}{4}$ to the empty list $\mathsf{in}_\ell()$, and with probability $\frac{3}{4}$ to the list $L' = \mathsf{in}_r(\mathsf{coin}(1/2) \otimes L)^!$ (up to the equivalence of $\mathsf{der}\, L^!$ with L) which is a value. One can access this value by evaluating the term $\mathsf{case}(L', z \cdot \_, y \cdot \mathsf{der}\, y)$ where $\_$ is any term of type $\mathcal{B} \otimes \mathcal{L}_1$ (we know that this branch will not be evaluated) and this term reduces to $(\mathbf{t}, L)$ or $(\mathbf{f}, L)$ with probability $\frac{1}{2}$. In turn, each of these terms (b, L) will reduce to $(b, \mathsf{in}_\ell())$ with probability $\frac{1}{4}$ and to (b, L') with probability $\frac{3}{4}$.

We can iterate this process, defining a term R of type $\mathcal{L}_1 \multimap \mathcal{L}_0$ which evaluates completely a terminable stream to a list:

$$R = \mathsf{fix}\, f^{!(\mathcal{L}_1\multimap\mathcal{L}_0)}\, \lambda x^{\mathcal{L}_1}\, \mathsf{case}(x, z \cdot \mathsf{in}_\ell(), z \cdot \langle \lambda y^{\mathbf{Bool}\otimes\mathcal{L}_1}\, (\mathsf{pr}_\ell y, \langle \mathsf{der}\, f\rangle \mathsf{pr}_r y)\rangle \mathsf{der}\, z)\,.$$

Then $\langle R\rangle L$, which is a closed term of type $\mathcal{L}_0$, terminates with probability 1. The expectation of the length of this "random list" is $\sum_{n=0}^{\infty} n(\frac{3}{4})^n = 12$.

We could also consider $\mathcal{L}_2 = \mathbf{1} \oplus (!\sigma \otimes \mathcal{L}_2)$ which allows to manipulate lists of objects of type σ (which can be a general type) without accessing their elements.

Probabilistic tests

We encode a program that flips a biased coin and goes on with M_1 or M_2 according to the outcome.

If $\mathcal{P} \vdash M_i : \sigma$ for $i = 1, 2$, we set $\mathsf{flip}_p(M_1, M_2) = \mathsf{if}(\mathsf{coin}(p), M_1, M_2)$ and this term satisfies $\mathcal{P} \vdash \mathsf{flip}_p(M_1, M_2) : \sigma$. If M_i reduces to a value V_i with probability q_i, then $\mathsf{flip}_p(M_1, M_2)$ reduces to V_1 with probability $p\, q_1$ and to V_2 with probability $(1-p)q_2$.

Let $n \in \mathbb{N}$ and let $\vec{p} = (p_0, \ldots, p_n)$ be such that $p_i \in [0,1] \cap \mathbb{Q}$ and $p_0 + \cdots + p_n \leqslant 1$. Then one defines a closed term $\mathsf{ran}(\vec{p})$, such that $\vdash \mathsf{ran}(\vec{p}) : \mathcal{N}$, which reduces to $\underline{i}$ with probability p_i for each $i \in \{0, \ldots, n\}$. The definition is by induction on n.

$$\mathsf{ran}(\vec{p}) = \begin{cases} \underline{0} & \text{if } p_0 = 1 \text{ whatever be the value of } n \\ \mathsf{if}(\mathsf{coin}(p_0), \underline{0}, \Omega^{\mathcal{N}}) & \text{if } n = 0 \\ \mathsf{if}(\mathsf{coin}(p_0), \underline{0}, \mathsf{succ}(\mathsf{ran}(\frac{p_1}{1-p_0}, \ldots, \frac{p_n}{1-p_0}))) & \text{otherwise} \end{cases}$$

As an example of use of the test to zero conditional, we define, by induction on k, a family of terms eq_k such that $\vdash \mathsf{eq}_k : \mathcal{N} \multimap \mathbf{1}$ and that tests the equality to k:

$$\mathsf{eq}_0 = \lambda x^{\mathcal{N}}\, \mathsf{ifzm}(x, (), z \cdot \Omega^{\mathbf{1}}) \qquad \mathsf{eq}_{k+1} = \lambda x^{\mathcal{N}}\, \mathsf{ifzm}(x, \Omega^{\mathbf{1}}, z \cdot \langle \mathsf{eq}_k\rangle z)$$

For M such that $\vdash M : \mathcal{N}$, the term $\langle \mathsf{eq}_k\rangle M$ reduces to $()$ with a probability equal to the probability of M to reduce to $\underline{k}$.

3.4.3 Interpretation of $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$ in Pcoh

The important point in the interpretation of $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$ is that positive types are interpreted as coalgebras (in the Eilenberg-Moore category $\mathbf{Pcoh}^!$, see §3.1.4) and general types as standard probabilistic coherent spaces (in **Pcoh**, see §3.1.2). Moreover, contexts are always interpreted as coalgebras so that the variables can be handled as in a call-by-value.

Given a type σ with free type variables contained in the repetition-free list $\vec{\zeta}$, and given a sequence $\vec{P}$ of length n of objects of $\mathbf{Pcoh}^!$, we define $[\sigma]_{\vec{\zeta}}(\vec{P})$ as an object of **Pcoh** and when φ is a positive type (whose free variables are contained in $\vec{\zeta}$) we define $[\varphi]^!_{\vec{\zeta}}(\vec{P})$ as an object of $\mathbf{Pcoh}^!$. These operations are continuous and their definitions follow the general pattern described in Ehrhard [2016].

Theorem 80. *Let φ be a positive type and let $\vec{\zeta} = (\zeta_1, \dots, \zeta_n)$ be a repetition-free list of type variables which contains all the free variables of φ. Let $\vec{P}$ be a sequence of n dense coalgebras. Then $[\varphi]^!_{\vec{\zeta}}(\vec{P})$ is a dense coalgebra. In particular, when φ is closed, the coalgebra $[\varphi]^!$ is dense.*

This is an immediate consequence of the definition of $[\varphi]^!$ and of Theorem 53.

Given a typing context $\mathcal{P} = (x_1 : \varphi_1, \dots, x_k : \varphi_k)$, a type σ and a term M such that $\mathcal{P} \vdash M : \sigma$, then M is interpreted as a morphism $[\![M]\!]_{\mathcal{P}} \in \mathbf{Pcoh}([\mathcal{P}], [\sigma])$. For all constructs of the language but probabilistic choice, this interpretation uses the generic structures of the model described in Section 3.1. The description of this interpretation can be found in Ehrhard [2016].

If $x_1 : \varphi_1, \dots, x_k : \varphi_k \vdash M : \sigma$, the morphism $[\![M]\!]_{\mathcal{P}}$ is completely characterized by its values on $(u_1, \dots, u_k) \in \mathsf{P}^!([\mathcal{P}]^!)$. Using this observation, we describe the interpretation of terms in Figure 3.11.

- $[\![\mathsf{coin}(p)]\!] = pe_{(\ell,*)} + (1-p)e_{(r,*)}$.
- $[\![()]\!] = 1 \in \mathsf{P}\mathbf{1} = [0,1]$.
- $[\![x_i]\!]_{\mathcal{P}}(\vec{u}) = u_i$.
- $[\![N^!]\!]_{\mathcal{P}}(\vec{u}) = ([\![N]\!]_{\mathcal{P}}(\vec{u}))^!$.
- $[\![(M_\ell, M_r)]\!]_{\mathcal{P}}(\vec{u}) = [\![M_\ell]\!]_{\mathcal{P}}(\vec{u}) \otimes [\![M_r]\!]_{\mathcal{P}}(\vec{u})$.
- $[\![\mathsf{in}_i N]\!]_{\mathcal{P}}(\vec{u}) = \mathrm{in}_i([\![N]\!]_{\mathcal{P}}(\vec{u}))$, $i \in \{\ell, r\}$.
- $[\![\mathsf{der}\, N]\!]_{\mathcal{P}}(\vec{u}) = \mathrm{der}_{[\sigma]}([\![N]\!]_{\mathcal{P}}(\vec{u}))$, assuming that $\mathcal{P} \vdash N : !\sigma$.
- If $\mathcal{P} \vdash N : \varphi \multimap \sigma$ and $\mathcal{P} \vdash R : \varphi$ then $[\![N]\!]_{\mathcal{P}}(\vec{u}) \in \mathsf{P}([\varphi] \multimap [\sigma])$, and $[\![R]\!]_{\mathcal{P}}(\vec{u}) \in \mathsf{P}([\varphi])$. Using the application of a matrix to a vector, $[\![\langle N\rangle R]\!]_{\mathcal{P}}(\vec{u}) = [\![N]\!]_{\mathcal{P}}(\vec{u})\,[\![R]\!]_{\mathcal{P}}(\vec{u})$.
- If $\mathcal{P}, x : \varphi \vdash N : \sigma$ then $[\![\lambda x^{\varphi} N]\!]_{\mathcal{P}}(\vec{u}) \in \mathsf{P}([\varphi] \multimap [\sigma])$ is completely described by the fact that, for all $u \in \mathsf{P}^!([\varphi]^!)$, one has $[\![\lambda x^{\varphi} N]\!]_{\mathcal{P}}(\vec{u})\, u = [\![N]\!]_{\mathcal{P},x:\varphi}(\vec{u}, u)$. This is a complete characterization of this interpretation by Theorem 80.
- If $\mathcal{P} \vdash N : \varphi_\ell \oplus \varphi_r$ and $\mathcal{P}, y_i : \varphi_i \vdash R_i : \sigma$ for $i \in \{\ell, r\}$, then $[\![\mathsf{case}(N, y_\ell \cdot R_\ell, y_r \cdot R_r)]\!]_{\mathcal{P}}(\vec{u}) = [\![R_\ell]\!]_{\mathcal{P},y_\ell:\varphi_\ell}(\vec{u}, \mathrm{pr}_\ell([\![N]\!]_{\mathcal{P}}(\vec{u}))) + [\![R_r]\!]_{\mathcal{P},y_r:\varphi_r}(\vec{u}, \mathrm{pr}_r([\![N]\!]_{\mathcal{P}}(\vec{u})))$ where $\mathrm{pr}_i \in \mathbf{Pcoh}(P_\ell \oplus P_r, P_i)$ is the ith "projection" introduced in §3.1.4, left inverse for in_i.
- If $\mathcal{P}, x : !\sigma \vdash N : \sigma$ then $[\![N]\!]_{\mathcal{P},x:!\sigma} \in \mathbf{Pcoh}(\underline{[\mathcal{P}]} \otimes ![\sigma], [\sigma])$ and $[\![\mathsf{fix}\, x^{!\sigma} N]\!]_{\mathcal{P}}(\vec{u}) = \sup_{n=0}^{\infty} f^n(0)$ where $f : \mathsf{P}[\sigma] \to \mathsf{P}[\sigma]$ is the Scott-continuous function given by $f(u) = [\![N]\!]_{\mathcal{P},x:!\sigma}(\vec{u}, u^!)$.
- If $\mathcal{P} \vdash N : \psi\,[\mathsf{Rec}\,\zeta \cdot \psi/\zeta]$ then $[\![\mathsf{fold}(N)]\!]_{\mathcal{P}} = [\![N]\!]_{\mathcal{P}}$, indeed $[\psi\,[\mathsf{Rec}\,\zeta \cdot \psi/\zeta]] = [\mathsf{Rec}\,\zeta \cdot \psi]$.
- If $\mathcal{P} \vdash N : \mathsf{Rec}\,\zeta \cdot \psi$ then $[\![\mathsf{unfold}(N)]\!]_{\mathcal{P}} = [\![N]\!]_{\mathcal{P}}$.

Figure 3.11 – Interpretation of $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$ in **Pcoh**. The terms are supposed typed as in Figure 3.9c, and $\vec{u} \in [\![\mathcal{P}]\!]$.

Examples of term interpretations

Let us now give interpretation of types and terms given as examples in §3.4.2.

$\mathcal{B} = \mathbf{1} \oplus \mathbf{1}$ satisfies $|[\mathcal{B}]| = \{(\ell,*), (r,*)\}$ and $u \in (\mathbb{R}^+)^{|[\mathcal{B}]|}$ satisfies $u \in \mathsf{P}[\mathcal{B}]$ iff $u_{(\ell,*)} + u_{(r,*)} \leqslant 1$. The

coalgebraic structure of this object is given by

$$(\mathrm{h}_{\mathcal{B}})_{(j,*),[(j_1,*),\dots,(j_k,*)]} = \begin{cases} 1 & \text{if } j = j_1 = \cdots = j_k \\ 0 & \text{otherwise.} \end{cases}$$

The object $\mathsf{N} = [\mathcal{N}]$ satisfies $\mathsf{N} = \mathbf{1} \oplus \mathsf{N}$ so that $|\mathsf{N}| = \{(\ell,*),(r,(\ell,*)),(r,(r,(\ell,*))),\dots\}$ and we use $\overline{n}$ for the element of $|\mathsf{N}|$ which has n occurrences of r. Given $u \in (\mathbb{R}^+)^{|\mathsf{N}|}$, we use $\ell(u)$ for the element of $(\mathbb{R}^+)^{|\mathsf{N}|}$ defined by $l(u)_{\overline{n}} = u_{\overline{n+1}}$. By definition of N, we have $u \in \mathsf{PN}$ iff $u_{\overline{0}} + \|\ell(u)\|_{\mathsf{N}} \leqslant 1$, and then $\|u\|_{\mathsf{N}} = u_{\overline{0}} + \|\ell(u)\|_{\mathsf{N}}$. It follows that $u \in \mathsf{PN}$ iff $\sum_{n=0}^{\infty} u_{\overline{n}} \leqslant 1$ and of course $\|u\|_{\mathsf{N}} = \sum_{n=0}^{\infty} u_{\overline{n}}$. Then the coalgebraic structure $\mathrm{h}_{\mathcal{N}}$ is defined exactly as $\mathrm{h}_{\mathcal{B}}$ above. In the sequel, we identify $|\mathsf{N}|$ with $\mathbb{N}$.

Let us give the interpretation of the terms defined in §3.4.2:

- $[\![\Omega^{\sigma}]\!] = [\![\mathsf{fix}\, x^{!\sigma}\, \mathsf{der}\, x]\!] = 0$
- $[\![\mathbf{t}]\!] = e_{(\ell,*)}$ and $[\![\mathbf{f}]\!] = e_{(r,*)}$
- $[\![\mathsf{if}(M, N_\ell, N_r)]\!]_{\mathcal{P}}(\vec{u}) = ([\![M]\!]_{\mathcal{P}})_{(\ell,*)}(\vec{u})[\![N_\ell]\!]_{\mathcal{P}}(\vec{u}) + ([\![M]\!]_{\mathcal{P}})_{(r,*)}(\vec{u})[\![N_r]\!]_{\mathcal{P}}(\vec{u})$
- $[\![\mathsf{flip}_p(M_\ell, M_r)]\!]_{\mathcal{P}}(\vec{u}) = p[\![M_\ell]\!]_{\mathcal{P}}(\vec{u}) + (1-p)[\![M_r]\!]_{\mathcal{P}}(\vec{u})$
- $[\![\underline{n}]\!] = \overline{n}$ for $n \in \mathbb{N}$
- $([\![\mathsf{succ}(M)]\!]_{\mathcal{P}})_{n+1}(\vec{u}) = [\![M]\!]_n(\vec{u})$
- $[\![\mathsf{ifzm}(M, N_\ell, x \cdot N_r)]\!]_{\mathcal{P}}(\vec{u}) = ([\![M]\!]_{\mathcal{P}})_0(\vec{u})[\![N_\ell]\!]_{\mathcal{P}}(\vec{u}) + \sum_{n=0}^{\infty} ([\![M]\!]_{\mathcal{P}})_{n+1}(\vec{u})[\![N_r]\!]_{\mathcal{P}}(\vec{u})(\overline{n})$
- $[\![\mathsf{ran}(\vec{p})]\!] = \sum_{i=1}^{n} p_i e_{\overline{i}}$
- $[\![\langle \mathsf{eq}_\ell \rangle M]\!]_{\mathcal{P}}(\vec{u}) = [\![M]\!]_{\mathcal{P}}(\vec{u})_\ell e_*$

3.4.4 Soundness and Adequacy

We state the main theorems and give the sketch of the proof of Adequacy which involves the Pitt's trick that we already used this trick in a simpler setting for probabilistic pure λ-calculus (see Page 76).

The invariance of the interpretation is proved by a structural induction on M:

Theorem 81. *If M satisfies $\mathcal{P} \vdash M : \sigma$ then*

$$[\![M]\!]_{\mathcal{P}} = \sum_{\mathcal{P} \vdash M' : \sigma} \mathsf{Red}_{M,M'} [\![M']\!]_{\mathcal{P}}$$

As usual, the Adequacy Theorem 84 is a consequence of the Adequacy Lemma 84 which relates operational and denotational semantics for ground type terms: if $\vdash M : \mathbf{1}$, then $[\![M]\!] = \mathsf{Red}^{\infty}_{M,()}$.

The left-to-right inequality is an immediate consequence of Theorem 81 and of the definition of Red^{∞} (see §3.4.1):

Theorem 82. *Let M be a term such that $\vdash M : \mathbf{1}$ so that $[\![M]\!] \in [0,1]$. Then $[\![M]\!] \geqslant \mathsf{Red}^{\infty}_{M,()}$.*

In spite of its very simple statement, the proof of the converse inequality is rather long. The difficulty comes from the recursive type definitions allowed by our syntax. The proof uses a combination of the techniques introduced to prove the Adequacy Lemma for pPCF (see §3.3.4) and Λ^+ (see §3.2.4).

We describe the thread of this proof which is due to Thomas Ehrhard, but do not give the involved technical details which can be found in Ehrhard and Tasson [2018].

As usual, the proof is based on the definition of logical relations between terms and elements of the model (more precisely, given any type σ, we have to define a relation between closed terms of types σ and elements of $\mathsf{P}[\sigma]$; let us call such a relation a *σ-relation*).

We have no positivity restrictions on the occurrence of type variables with respect to which recursive types are defined. Besides, types are neither covariant nor contravariant with respect to these type variables. Thus, we use a very powerful technique introduced in Pitts [1994] for defining this logical relation.

Indeed, a type variable ζ can have positive and negative occurrences in a positive[9] type φ. Consider for instance the case $\varphi = !(\zeta \multimap \zeta)$, where the type variable ζ has a positive (on the right of the $\multimap$) and a negative occurrence (on the left). To define the logical relation associated with $\mathsf{Rec}\, \zeta \cdot \varphi$, we have

9. Warning: the word "positive" has two different meanings here!

to find a fixpoint for the operation which maps a $(\mathsf{Rec}\,\zeta\cdot\varphi)$-relation $\mathcal{R}$ to the relation $\Phi(\mathcal{R}) = !(\mathcal{R} \multimap \mathcal{R})$ (which can be defined using $\mathcal{R}$ as a "logical relation" in a fairly standard way). Relations are naturally ordered by inclusion, and this strongly suggests to define the above fixpoint using this order relation by Tarski's Fixpoint Theorem. The problem however is that Φ is neither a monotone nor an anti-monotone operation on relations. Indeed, ζ has a positive and a negative occurrence in φ.

It is here that Pitts's trick comes in. We replace the relations $\mathcal{R}$ with pairs of relations $\mathcal{R} = (\mathcal{R}^-, \mathcal{R}^+)$ ordered as follows: $\mathcal{R} \sqsubseteq \mathcal{S}$ if $\mathcal{R}^+ \subseteq \mathcal{S}^+$ and $\mathcal{S}^- \subseteq \mathcal{R}^-$. Then we define accordingly $\Phi(\mathcal{R})$ as a pair of relations by $\Phi(\mathcal{R})^- = !(\mathcal{R}^+ \multimap \mathcal{R}^-)$ and $\Phi(\mathcal{R})^+ = !(\mathcal{R}^- \multimap \mathcal{R}^+)$. Now the operation Φ is monotone with respect to the $\sqsubseteq$ relation and it becomes possible to apply Tarski's Fixpoint Theorem to Φ and get a pair of relations $\mathcal{R}$ such that $\mathcal{R} = \Phi(\mathcal{R})$. The next step consists in proving that $\mathcal{R}^- = \mathcal{R}^+$. This is obtained by means of an analysis of the definition of the interpretation of fixpoints of types as colimits in the category $\mathbf{Pcoh}_{\subseteq}$. One is finally in position of proving a fairly standard "Logical Relation Lemma" from which adequacy follows straightforwardly.

Theorem 83 (Logical Relation Lemma)**.** *Let $x_1 : \varphi_1, \dots, x_k : \varphi_k \vdash M : \sigma$, $(V_i, v_i) \in \mathcal{R}(\varphi_i)$ (where V_i is a value and $v_i \in \mathsf{P}^!|\varphi_i|$) for $i \in \{1, \dots, k\}$ and $\vec{v} = (v_1, \dots, v_k)$. Then,*

$$(M\,[V_1/x_1, \dots, V_k/x_k]\,, [\![M]\!]^{x_1,\dots,x_k}\,\vec{v}) \in \mathcal{R}(\sigma).$$

There are other techniques that are needed for this proof. The most important is that values are handled in a special way so that we actually consider two kinds of pairs of relations. Also, a kind of "biorthogonality closure" plays an essential role in the handling of positive types, no surprise for the readers acquainted with Linear Logic, see for instance the proof of normalization in Girard [1987].

From Theorem 82 and Theorem 83, we deduce the Adequacy Lemma:

Lemma 84. *Let M be a closed term such that $\vdash M : \mathbf{1}$. Then $[\![M]\!] = \mathsf{Red}^{\infty}_{M,()}$.*

We finally get the Adequacy Theorem which relates denotational and observational equivalence (see Figure 3.10f) in the same way as for pPCF (see Theorem 77):

Theorem 85 (Adequacy)**.** *Let M, M' be closed terms of $\Lambda^{\mathsf{p}}_{\mathsf{HP}}$ such that $\vdash M : \sigma$ and $\vdash M' : \sigma$.*
If $[\![M]\!] = [\![M']\!]$ then $M \sim M'$.

Proof. Assume that $[\![M]\!] = [\![M']\!]$. Let C be a closed term of type $!\sigma \multimap \mathbf{1}$. Then, we apply Lemma 84:

$$\mathsf{Red}^{\infty}_{\langle C\rangle M^!,()} = [\![\langle C\rangle M^!]\!] = [\![C]\!][\![M]\!]^! = [\![C]\!][\![M']\!]^! = \mathsf{Red}^{\infty}_{\langle C\rangle M'^!,()}\,.$$

□

3.4.5 Full Abstraction

We prove now the Full Abstraction Theorem 88, that is the converse of the Adequacy Theorem. The proof follows the same pattern as for pPCF (see §3.3.5). However, it is not simply an adaptation to this more granular setting. Indeed, we need to take into account values but also, we need the ingredient of dense coalgebras (see the paragraph on Dense Coalgebras Page 71).

We give in this paragraph the sketch of reasoning and underline the changes with respect to the proof for pPCF (see §3.3.5).

We reason by contrapositive and assume that two closed terms M_1 and M_2 have different semantics. Remember from §3.4.3 and §3.1.1, that a closed term of type σ is interpreted as a vector with indices in the web $|[\![\sigma]\!]|$, so that there is $a \in |[\![\sigma]\!]|$ such that $[\![M_1]\!]_a \neq [\![M_2]\!]_a$. We want to design a term that will separate M_1 and M_2 observationally (see Figure 3.10f for the definition of observational equivalence).

We define a *testing term* $\vdash a^- : !\mathcal{N} \multimap (!\sigma \multimap \mathbf{1})$ that will depend only on the structure of the element a of the web. We then use properties of the semantics (namely that terms of type $!\mathcal{N} \multimap \tau$ can be seen as power series as explained in Theorem 45 to find reals $\vec{p}$ such that the context $C = \langle a^-\rangle\mathsf{ran}(\vec{p})^!$ separates M_1 and M_2:

$$\mathsf{Red}^{\infty}_{\langle\langle a^-\rangle\mathsf{ran}(\vec{p})^!\rangle M_1^!,()} \neq \mathsf{Red}^{\infty}_{\langle\langle a^-\rangle\mathsf{ran}(\vec{p})^!\rangle M_2^!,()}$$

Although we spare the reader of the technical definition of the testing terms (they can be found in Ehrhard and Tasson [2018]), we describe their properties. The testing term a^- is defined by induction on the structure of the point a. Actually, we need three kinds of *testing terms*:

- Given a positive type φ and $a \in |[\varphi]|$, we define a term a^0 such that

$$\vdash a^0 : !\mathcal{N} \multimap \varphi \multimap \mathbf{1}.$$

- Given a general type σ and $a \in |[\sigma]|$, we define terms a^+ and a^- such that

$$\vdash a^+ : !\mathcal{N} \multimap \sigma \qquad\qquad \vdash a^- : !\mathcal{N} \multimap !\sigma \multimap \mathbf{1}.$$

We also introduce natural numbers $|a|^0$, $|a|^-$ and $|a|^+$ depending only on a. They represent the finite numbers of parameters on which the power series $\widehat{[\![a^0]\!]}$, $\widehat{[\![a^-]\!]}$ and $\widehat{[\![a^+]\!]}$ depend respectively.

We denote as $\mathfrak{m}^0(a)$, $\mathfrak{m}^-(a)$ and $\mathfrak{m}^+(a)$ natural numbers depending only on a and that will appear respectively as the coefficient of the unitary monomial $\prod_{k=0}^{|a|^0} \zeta_k$, $\prod_{k=0}^{|a|^-} \zeta_k$ and $\prod_{k=0}^{|a|^+} \zeta_k$ of the corresponding power series. These numbers are all non negative.

Lemma 86 states a key observation: the semantics of a^- is a power series with finitely many parameters. The coefficient of the unitary monomial can be seen as a morphism in $\mathsf{P}[!\sigma \multimap \mathbf{1}]$.

Lemma 86. *Let σ be a general type and $t \in \mathsf{P}[!\mathcal{N} \multimap \sigma]$.*

1. *Assume that there is $k \in \mathbb{N}$ such that for any $c \in |[\sigma]|$, the power series $\widehat{t}_c$ over $\mathsf{P}[\mathcal{N}]$ depends on the k first parameters. For any $c \in |[\sigma]|$, let us denote as $\mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}\left(\widehat{t}\right)_c$ the coefficient of the monomial $\zeta_0 \dots \zeta_{k-1}$ of $\widehat{t}_c$. Then, $k^{-k}\, \mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}\left(\widehat{t}\right) \in \mathsf{P}[\sigma]$.*

2. *Assume moreover that $\sigma = \varphi \multimap \tau$ where φ is a positive type and τ a general type. Let $m \in \mathsf{P}[\tau]$ and $a \in |[\varphi]|$. If $\forall u \in \mathsf{P}^!{}[\varphi]^!\ \mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}\left(\widehat{t}\right) u = m u_a$ then $\forall u \in \mathsf{P}[\varphi]\ \mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}\left(\widehat{t}\right) u = m u_a$.*

We are now ready to prove the central point in the proof of Full Abstraction. Namely, the coefficient of the unitary monomial of a testing term associated with a point a of the web allows to extract the a-coefficient of an argument, up to a non-zero coefficient depending only on a.

Lemma 87. *Let σ be a type and $a \in |[\sigma]|$.*

1. *Assume that $\sigma = \varphi$ is positive. If $a' \in |[\varphi]|$, then $\widehat{[\![a^0]\!]}_{(a',*)}$ is a power series over $\mathsf{P}[\mathcal{N}]$ depending on $|a|^0$ parameters. We define $\mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}\left([\![a^0]\!]\right)_{(a',*)} = \mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}\left(\widehat{[\![a^0]\!]}_{(a',*)}\right)$. It satisfies $\mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}\left([\![a^0]\!]\right) \in \mathsf{P}[\varphi \multimap \mathbf{1}]$ and for any $u \in \mathsf{P}([\varphi])$, $\mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}\left([\![a^0]\!]\right) u = \mathfrak{m}^0(a)\, u_a$.*

2. *Assume that σ is a general type. For any $a' \in |[\sigma]|$, $\widehat{[\![a^+]\!]}_{a'}$ is a power series over $\mathsf{P}[\mathcal{N}]$ depending on $|a|^+$ parameters. We define $\mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}([\![a^+]\!])_{a'} = \mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}\left(\widehat{[\![a^+]\!]}_{a'}\right)$. It satisfies $\mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}([\![a^+]\!]) \in \mathsf{P}[\sigma]$ and $\mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}([\![a^+]\!]) = \mathfrak{m}^+(a)\, e_a$ where e_a is the base vector such that $(e_a)_{a'} = \delta_{a',a}$ for $a' \in |[\sigma]|$.*

3. *Let σ be a general type. For any $a' \in |[!\sigma]|$, $\widehat{[\![a^-]\!]}_{(a',*)}$ is a power series over $\mathsf{P}[\mathcal{N}]$ depending on $|a|^-$ parameters. We define $\mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}([\![a^-]\!])_{(a',*)} = \mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}\left(\widehat{[\![a^-]\!]}_{(a',*)}\right)$. It satisfies $\mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}([\![a^-]\!]) \in \mathsf{P}[!\sigma \multimap \mathbf{1}]$ and for any $u \in \mathsf{P}[\sigma]$, $\mathfrak{c}^{\mathbf{1}}_{\vec{\zeta}}([\![a^-]\!])\, u^! = \mathfrak{m}^-(a)\, u_a$.*

Ingredients of the proof. The proof is by mutual induction on the size of a and the structure of φ. It uses crucially Lemma 86 and Theorem 80 on the characterization of dense types that allows to compute functions over dense types only by computing them on coalgebric points. □

Theorem 88 (Full Abstraction)**.** *If $\vdash M_1 : \sigma$ and $\vdash M_2 : \sigma$ satisfy $M_1 \sim M_2$ then $[\![M_1]\!] = [\![M_2]\!]$.*

Proof. Reason by contrapositive and assume that $[\![M_1]\!] \neq [\![M_2]\!]$.

There is $a \in |[\sigma]|$ such that $[\![M_1]\!]_a \neq [\![M_2]\!]_a$. Then by Lemma 87, $[\![\lambda x^{!\mathcal{N}}\, \langle\langle a^-\rangle x\rangle M_i^!]\!]$, for $i \in \{1,2\}$, are real power series with finitely parameters and different coefficients. Indeed, because $[\![\lambda x^{!\mathcal{N}}\, \langle\langle a^-\rangle x\rangle M_i^!]\!]\, \vec{\zeta}^{\,!} = [\![a^-]\!]\, \vec{\zeta}^{\,!} [\![M_i]\!]^!$, the coefficients of their unitary monomial $\zeta_0 \dots \zeta_{|a|^- - 1}$ are

$$\mathfrak{m}^-(a)\, [\![M_1]\!]_a \neq \mathfrak{m}^-(a)\, [\![M_2]\!]_a.$$

By real analysis for multivariate power series, there is $\vec{p} = (p_0, \dots, p_{|a|^- - 1}) \in \mathsf{P}[\mathcal{N}]$ with $p_i \in \mathbb{Q} \cap [0,1]$ such that the two power series are different: $[\![a^-]\!]\, \vec{p}^{\,!}\, [\![M_1]\!]^! \neq [\![a^-]\!]\, \vec{p}^{\,!}\, [\![M_2]\!]^!$.

We compute $[\![\langle\langle a^-\rangle \mathsf{ran}(\vec{p})^!\rangle M_i^!]\!] = [\![a^-]\!]\, \vec{p}^{\,!}\, ([\![M_i]\!])$. By the Adequacy Lemma 84, $\langle\langle a^-\rangle \mathsf{ran}(\vec{p})^!\rangle M_i^!$ converges to $()$ with probability $[\![\langle\langle a^-\rangle \mathsf{ran}(\vec{p})^!\rangle M_i^!]\!]$.

Thus, the two terms converge to $()$ with different probabilities. It follows that $M_1 \not\sim M_2$ (see 3.10f). □

$$A, B \ldots := \mathcal{R} \mid \sigma \Rightarrow \tau$$

(a) Types of pRPCF

$$M, N, \ldots := \underline{r} \mid \underline{f}(M_1, \ldots, M_n) \mid x \mid \lambda x^{\sigma} M \mid (M)\, N \mid \mathsf{fix}(M) \mid \mathsf{ifz}(L, M, N) \mid \mathsf{let}(x, M, N) \mid \mathsf{sample}$$

(b) Terms of $\Lambda^p_{\mathcal{R}}$

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{r \in \mathbb{R}}{\Gamma \vdash \underline{r} : \mathcal{R}} \qquad \frac{f \text{ meas. } \mathbb{R}^n \to \mathbb{R} \qquad \Gamma \vdash M_i : \mathcal{R},\ \forall i \leqslant n}{\Gamma \vdash \underline{f}(M_1, \ldots, M_n) : \mathcal{R}}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A M : A \Rightarrow B} \qquad \frac{\Gamma \vdash M : A \Rightarrow B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M)\, N : B} \qquad \frac{\Gamma \vdash M : A \Rightarrow A}{\Gamma \vdash \mathsf{fix}(M) : A}$$

$$\frac{\Gamma \vdash L : \mathcal{R} \qquad \Gamma \vdash M : \mathcal{R} \qquad \Gamma \vdash N : \mathcal{R}}{\Gamma \vdash \mathsf{ifz}(L, M, N) : \mathcal{R}} \qquad \frac{\Gamma \vdash M : \mathcal{R} \qquad \Gamma, x : \mathcal{R} \vdash N : \mathcal{R}}{\Gamma \vdash \mathsf{let}(x, M, N) : \mathcal{R}}$$

$$\frac{}{\Gamma \vdash \mathsf{sample} : \mathcal{R}}$$

A typing context is a sequence $\Gamma = (x_1 : \sigma_1, \ldots, x_n : \sigma_n)$ where the x_i's are pairwise distinct variables.

(c) Typing system of pRPCF

Figure 3.12 – Syntax of pRPCF

3.5 Measurable cones and measurable, stable functions, an adequate model of pRPCF

Our goal is to show the expressiveness of the category of measurable cones and measurable, stable functions, dentoed as $\mathbf{Cstab}_{\mathsf{m}}$.

We introduce a probabilistic programming language pRPCF with reals as ground-types. Notice that this is an ideal language which does not deal with the issues about a realistic implementation of computations over real numbers. We refer the reader interested in such implementation to Vuillemin [1988] or Escardó [1996]. The language pRPCF (see Figure 3.12) follows the same pattern as the language pPCF (see Figure 3.6). Yet there are differences: the ground-type of reals replaces the one of natural numbers; A countable set of basic measurable real functions replace the successor which is the basic natural number construction; the sampling generating the uniform distribution over the unit segment replaces the biased coins that generate true or false with probability p or $1 - p$.

We present the denotational semantics in §3.5.3. This semantics designed by Thomas Ehrhard is a tour de force as it is a cartesian closed category with measurable cones as objects and measurable functions as morphims. Recall that one of the issue to design a semantics handling higher order functions and continuous probabilistic programming is that the categories **Meas** and **Kern** are not cartesian closed.

We present in §3.5.6 a bunch of examples of probabilistic algorithms based on probabilistic distributions and on rejection sampling algorithms. We will prove the correctness of these encodings using the denotational semantics (see §3.5.3), this latter corresponding to the program operational behavior by the Adequacy Theorem 102.

3.5.1 Syntax of pRPCF

We give in Figure 3.12 the syntax of $\Lambda^p_{\mathcal{R}}$ of our probabilistic real PCF, briefly pRPCF, together with the typing rules. The types are presented in Figure 3.12a, where the constant $\mathcal{R}$ is the ground type for the set of real numbers. The terms are presented in Figure 3.12b where the metavariable $\underline{r}$ ranges over the whole set of reals $\mathbb{R}$, while the metavariable $\underline{f}$ ranges over a fixed countable set $\mathcal{C}$ of functional identifiers, basic measurable functions over real numbers. Examples of these functions include addition $+$, comparison $>$, and equality $=$; they are often written in infix notation. When clear from the context, we sometimes write f for $\underline{f}$. To be concise, we consider only the ground type $\mathcal{R}$, the boolean operators (like $>$ or $=$) then evaluate to $\underline{1}$ or $\underline{0}$, representing resp. true and false.

The typing system, presented in Figure 3.12c, extends the usual constructs of PCF with reals and continuous probabilities. The constant sample stands for the uniform distribution over $[0, 1]$, i.e. the Lebesgue measure $\lambda_{[0,1]}$ restricted to the unit interval. The fact that pRPCF has only this distribution as a primitive is not limiting, in fact many other probabilistic measures (like binomial, geometric, gaussian

or exponential distribution) can be defined from $\lambda_{[0,1]}$ and the other constructs of the language, see e.g. Park et al. [2008] and §3.5.6. The `let` construction allows a call-by-value discipline over the ground type $\mathcal{R}$: the execution of $\mathsf{let}(x, M, N)$ samples a value (a real number r) from a probabilistic distribution M and passes it to N by replacing *every* free occurrence of x in N with $\underline{r}$. This primitive[10] is essential for the expressiveness of pRPCF and will be discussed both operationally and semantically.

pRPCF has a limited number of constructs, but it is known that many probabilistic primitives can be introduced as syntactic sugar from the ones in pRPCF, as shown in examples (see §3.5.6).

3.5.2 Operational Semantics

In Figure 3.13, we define the operational semantics of pRPCF as an extension to a *continuous setting* of the operational semantics Markov chain of Danos and Ehrhard [2011] presented in the preceding sections.

The operational semantics is defined starting from the *rewriting rules* of Figure 3.13a, extending the standard call-by-name reduction of PCF (see Plotkin [1977]). The probabilistic primitive sample draws a possible value from $[0, 1]$. A *redex* is a term in one of the forms at left-hand side of the $\xrightarrow{\delta}$ relation, defined in Figure 3.13a. A *normal form* is a term M which is not reducible under $\xrightarrow{\delta}$. Notice that the closed normal forms of ground type $\mathcal{R}$ are the real numerals. The definition of the evaluation context (Figure 3.13b) is the usual one defining the deterministic lazy call-by-name strategy: we do not reduce under an abstraction and there is always at most one redex to reduce. It is standard to check the Subject Reduction property: if $\Gamma \vdash M : A$ and $M \xrightarrow{\delta} N$, then $\Gamma \vdash N : A$.

In order to define observational equivalence, we need to represent the probability of convergence of a term to a normal form. We define, in Figure 3.13d, a stochastic kernel Red (see definition in §3.5.7) such that $\mathrm{Red}(M, U)$ is the probability that a term M reduces into the measurable set of terms U. In this way, we have replaced the *discrete* stochastic matrix used in the Sections 3.3 and 3.4 by its *continuous* counterpart, namely a stochastic kernel.

Before defining this kernel, we equip $\Lambda^p_{\mathcal{R}}$ with a structure of measurable space whose definition is spelled out in Figure 3.13c. This defines a σ-algebra $\Sigma_{\Lambda^p_{\mathcal{R}}}$ of sets of terms equivalent to the one given in Borgström et al. [2016] and Staton et al. [2016] for slightly different languages.

From now on, let us fix $(z_i)_{i\in\mathbb{N}}$, an enumeration of variables of type $\mathcal{R}$ without repetitions. Notice that any term $M \in \Lambda^p_{\mathcal{R}}$ with n different occurrences of real numerals, can be decomposed *univocally* into a term $z_1 : \mathcal{R}, \ldots, z_n : \mathcal{R}, \Gamma \vdash S : A$ without real numerals and a substitution $\sigma = \{\underline{r_1}/z_1, \ldots, \underline{r_n}/z_n\}$, such that: (i) $M = S\sigma$; (ii) each z_i occurs exactly once in S; (iii) z_i occurs before z_{i+1} reading the term from left to right. Because of this latter condition, we can omit the name of the substituted variables, writing simply $S\vec{r}$ with $\vec{r} = (r_1, \ldots, r_n)$. We denote as $\Lambda_n^{\Gamma\vdash A}$ the set of terms in $\Lambda^p_{\mathcal{R}}$ with no occurrence of numerals and respecting conditions (ii) and (iii) above. Let $S, T, \ldots$ range over such real-numeral-free terms.

Given $S \in \Lambda_n^{\Gamma\vdash A}$, we set $\Lambda^p_{\mathcal{R}\,S} = \{M \in \Lambda^p_{\mathcal{R}} \text{ s.t. } \exists \vec{r} \in \mathbb{R}^n, M = S\vec{r}\}$. The bijection $s : \Lambda^p_{\mathcal{R}\,S} \to \mathbb{R}^n$ given by $s(S\vec{r}) = \vec{r}$ endows $\Lambda^p_{\mathcal{R}\,S}$ with a σ-algebra isomorphic to $\Sigma_{\mathbb{R}^n}$: $U \in \Sigma_{\Lambda^p_{\mathcal{R}\,S}}$ iff $s(U) \in \Sigma_{\mathbb{R}^n}$. The fact that $\Lambda_n^{\Gamma\vdash A}$ is countable and that **Kern** has countable coproducts (see §3.5.7), allows us to define the measurable space of pRPCF terms of type $\Gamma \vdash A$ as the coproduct:

$$(\Lambda^p_{\mathcal{R}}, \Sigma_{\Lambda^p_{\mathcal{R}}}) = \coprod_{n\in\mathbb{N}, S\in\Lambda_n^{\Gamma\vdash A}} (\Lambda^p_{\mathcal{R}\,S}, \Sigma_{\Lambda^p_{\mathcal{R}\,S}}) \tag{3.5}$$

We define the stochastic kernel Red in Figure 3.13d. Given a set $U \subseteq \mathbb{R}$, we denote as $\underline{U} \in \Lambda^p_{\mathcal{R}}$ the set of numerals associated with the real numbers in U. Of course U is measurable in $\mathbb{R}$ iff $\underline{U}$ is measurable in $\Lambda^p_{\mathcal{R}}$. The following lemma allows us to define Red and Red^∞

Lemma 89. *Given $\Gamma, x : B \vdash M : A$ the function $\mathrm{Subst}_{x,M}$ mapping $N \in \Lambda^p_{\mathcal{R}}$ to $M\{N/x\} \in \Lambda^p_{\mathcal{R}}$ is measurable.*

The last case of the definition of Red sets the normal forms as accumulation points of Red, so that $\mathrm{Red}(M, U)$ gives the probability that we observe U after *at most one* reduction step applied to M. The definition in the case of $E[\mathsf{sample}]$ specifies that sample is drawing from the uniform distribution over $[0, 1]$. Notice that, if $U \subseteq \mathbb{R}$ is measurable, then the set $\{r \in [0, 1] \text{ s.t. } E[\underline{r}] \in U\}$ is measurable by Lemma 89.

10. Notice that this primitive corresponds to the `sample` construction in Park et al. [2008].

$$(\lambda x^{\sigma}\, M)\, N \xrightarrow{\delta} M\,[N/x] \qquad \underline{f}(\underline{r_1}, \ldots, \underline{r_n}) \xrightarrow{\delta} \underline{f(r_1, \ldots, r_n)}$$

$$\mathsf{ifz}(\underline{r}, M, N) \xrightarrow{\delta} \begin{cases} M & \text{if } r = 0, \\ N & \text{otherwise.} \end{cases} \qquad \mathsf{let}(x, \underline{r}, M) \xrightarrow{\delta} M\{\underline{r}/x\}$$

$$\mathsf{fix}(M) \xrightarrow{\delta} M(\mathsf{fix}(M)) \qquad \mathsf{sample} \xrightarrow{\delta} \underline{r} \quad \text{for any } r \in [0,1].$$

(a) Reduction rules of a pRPCF redex.

$$E[\cdot] := (E[\cdot])\, M \mid \mathsf{ifz}(E[\cdot], M, N) \mid \mathsf{let}(x, E[\cdot], N) \mid \underline{f}(\underline{r}_1, \ldots, \underline{r}_{i-1}, E[\cdot], M_{i+1}, \ldots, M_n)$$

$$E[M] \xrightarrow{\delta} E[N], \text{ whenever } M \xrightarrow{\delta} N$$

(b) Grammar of the evaluation contexts and context closure of the reduction.

$$U \subseteq \Lambda^p_{\mathcal{R}} \text{ is measurable if and only if } \forall n, \forall S \in \Lambda_n^{\Gamma \vdash A}, \{\vec{r} \text{ s.t. } S\vec{r} \in U\} \in \Sigma_{\mathbb{R}^n}$$

(c) Measurable space structure on $(\Lambda^p_{\mathcal{R}}, \Sigma_{\Lambda^p_{\mathcal{R}}})$

If $M \in \Lambda^p_{\mathcal{R}}$ and $U \subseteq \Lambda^p_{\mathcal{R}}$ measurable,

$$\mathrm{Red}(M,U) = \begin{cases} \delta_{E[N]}(U) & \text{if } M = E[R], \\ & R \xrightarrow{\delta} N \text{ and } R \neq \mathsf{sample}. \\ \lambda\{r \in [0,1] \text{ s.t. } E[\underline{r}] \in U\} & \text{if } M = E[\mathsf{sample}], \\ \delta_M(U) & \text{if } M \text{ normal form.} \end{cases}$$

$$\mathrm{Red}^{n+1}(M,U) = (\mathrm{Red} \circ \mathrm{Red}^n)(M,U) = \int_{\Lambda^p_{\mathcal{R}}} \mathrm{Red}(t,U)\, \mathrm{Red}^n(M, dt)$$

$$\mathrm{Red}^{\infty}(M,U) = \bigvee_n \left(\mathrm{Red}^n(M,U)\right).$$

(d) Stochastic kernels of reduction.

$M \sim M'$, if $\mathrm{Red}^{\infty}_{(C)M,U} = \mathrm{Red}^{\infty}_{(C)M',U}$,
for all closed term C of type $A \Rightarrow \mathcal{R}$, for all measurable set $U \subseteq \Lambda^p_{\mathcal{R}}$ of numerals.

(e) Observational equivalence of closed terms.

Figure 3.13 – Operational semantics of pRPCF.

Proposition 90. *For any sequent* $\Gamma \vdash A$*, the map* Red *is a stochastic kernel from* $\Lambda^p_{\mathcal{R}}$ *to* $\Lambda^p_{\mathcal{R}}$*.*

Proof. The fact that $\mathrm{Red}(M, \_)$ is a measure is an immediate consequence of the definition of Red and the fact that any evaluation context $E[\,]$ defines a measurable map (Lemma 89).

Given a measurable set $U \subseteq \Lambda^p_{\mathcal{R}}$, we must prove that $\mathrm{Red}(\_, U)$ is a measurable function from $\Lambda^p_{\mathcal{R}}$ to $[0,1]$. Since $\Lambda^p_{\mathcal{R}}$ can be written as the coproduct described in Equation (3.5), it is sufficient to prove that for any n and $S \in \Lambda_n^{\Gamma \vdash A}$, $\mathrm{Red}_S(\_, U) : \Lambda^p_{\mathcal{R}S} \to [0,1]$ is a measurable function. We reason by case study on the shape of S using the definition of a redex and the fact that there is always at most one redex to reduce. □

We can then iterate Red using the composition of stochastic kernels (see Equation (3.7) of §3.5.7). In Figure 3.13d, we define Red^{n+1} which gives the probability that we observe U after at most $n+1$ reduction steps from M. Because the normal forms are accumulation points, one can prove by induction on n that:

Lemma 91. *Let* $\Gamma \vdash M : A$ *and let* U *be a measurable set of normal forms in* $\Lambda^p_{\mathcal{R}}$*. The sequence* $(\mathrm{Red}^n(M,U))_n$ *is monotone non-decreasing.*

We can then define, for $M \in \Lambda^p_{\mathcal{R}}$ and U a measurable set of normal forms, its limit Red^∞ (see Figure 3.13d). In particular, if M is a closed term of ground type $\mathcal{R}$, the only normal forms that M can reach are numerals, in this case $\mathrm{Red}^\infty(M, \_)$ corresponds to the probabilistic distribution over $\mathbb{R}$ which is computed by M according to the operational semantics of pRPCF.

The *observational equivalence* relation $\sim$ between closed terms is given in Figure 3.13e.

3.5.3 Denotational Semantics: the category $\mathbf{Cstab}_{\mathsf{m}}$

In §3.1, we have presented **Pcoh** as a model for probabilistic language restricted to *countable* data types (like booleans and natural numbers, excluding the real numbers). Any distribution over a countable set is *discrete*, i.e. it can be described as a linear combination of its possible outcomes and there is no need of a notion of measurable space. In previous sections, we have shown that the category $\mathbf{PCoh}_!$ of *probabilistic coherence spaces* and analytic functions gives fully abstract denotational models of functional languages extended with a random *natural number* generator. The main goal of this section is to present a generalization of these models in order to account for *continuous* data types also.

The major difficulty for such a generalization is that a probabilistic coherence space is defined with respect to a kind of canonical basis (called *web*) that corresponds to the possible samples of a distribution, at the level of ground types. For continuous data types, these webs should be replaced by measurable spaces, and then one is stuck on the already mentioned (see Page 63) impossibility of associating a measurable space with a functional type – both **Meas** and **Kern** being not cartesian closed.

Complete cones

Our solution, for an axiomatic presentation not referring to a web, is to replace probabilistic coherence spaces with cones defined in Andô [1962] and already used by Selinger [2004]. A cone is similar to a normed vector space, but with non-negative real scalars.

A *complete cone* P is an $\mathbb{R}^+$-semimodule together with a norm $\|\_\|_P$ satisfying some natural axioms and such that the unit ball $\mathcal{B}P$, defined by the norm, is complete with respect to the cone order $\leqslant_P$.

Definition 92. A *cone* P is an $\mathbb{R}^+$-semimodule given together with an $\mathbb{R}^+$-valued function $\|\_\|_P$ such that the following conditions hold for all $x, x', y, y' \in P$ and $\alpha \in \mathbb{R}^+$

$$x + y = x + y' \Rightarrow y = y' \qquad \|\alpha x\|_P = \alpha \|x\|_P \qquad \|x + x'\|_P \leqslant \|x\|_P + \|x'\|_P$$
$$\|x\|_P = 0 \Rightarrow x = 0 \qquad \|x\|_P \leqslant \|x + x'\|_P.$$

For $\alpha \in \mathbb{R}^+$ the set $\mathcal{B}P(\alpha) = \{x \in P \mid \|x\|_P \leqslant \alpha\}$ is *the ball of* P *of radius* α. The *unit ball* is $\mathcal{B}P = \mathcal{B}P(1)$. A subset S of P is *bounded* if $S \subseteq \mathcal{B}P(\alpha)$ for some $\alpha \in \mathbb{R}^+$.

Observe that $\|0\|_P = 0$ by the second condition (homogeneity of the norm) and that if $x + x' = 0$ then $x' = x = 0$ by the last condition (monotonicity of the norm).

Let us define *the cone order relation* of P. Let x and $x' \in P$. We write $x \leqslant_P x'$ if there is a $y \in P$ such that $x' = x + y$. This y is then unique, and we denote $x' - x = y$. The relation $\leqslant_P$ is easily seen to be an order relation on P. (The usual laws of calculus using subtraction hold under the restriction that all usages of subtraction must be well-defined, for instance, if $x, y, z \in P$ satisfy $z \leqslant_P y \leqslant_P x$ then we have $x - z = (x - y) + (y - z)$.)

A cone P is *complete* if any non-decreasing sequence $(x_n)_{n \in \mathbb{N}}$ of elements of $\mathcal{B}P$ has a least upper bound $\sup_{n \in \mathbb{N}} x_n \in \mathcal{B}P$.

Any probabilistic coherence space can be seen as a cone as well as the set $\mathsf{Meas}(X)$ of all bounded measures over a measurable space X:

Example 93. Let $\mathcal{X} = (|\mathcal{X}|, \mathsf{P}\mathcal{X})$ be a probablistic coherence space (see §3.1.1). Remember that this means that $|\mathcal{X}|$ is a countable set (called web) and $\mathsf{P}\mathcal{X} \subseteq (\mathbb{R}^+)^{|\mathcal{X}|}$ satisfies $\mathsf{P}\mathcal{X} = \mathsf{P}\mathcal{X}^{\perp\perp}$ (where, given $\mathcal{F} \subseteq (\mathbb{R}^+)^{|\mathcal{X}|}$, the set $\mathcal{F}^\perp \subseteq (\mathbb{R}^+)^{|\mathcal{X}|}$ is $\mathcal{F}^\perp = \{u' \in (\mathbb{R}^+)^{|\mathcal{X}|} \mid \forall u \in \mathcal{F},\ \sum_{a\in|\mathcal{X}|} u_a u'_a \leqslant 1\}$)[11]. We define a cone $\widehat{\mathcal{X}}$ by setting $\widehat{\mathcal{X}} = \{u \in (\mathbb{R}^+)^{|\mathcal{X}|} \mid \exists \varepsilon > 0\ \varepsilon u \in \mathsf{P}\mathcal{X}\}$, defining algebraic operations in the usual componentwise way and setting $\|u\|_{\widehat{\mathcal{X}}} = \inf\{\alpha > 0 \mid \frac{1}{\alpha}u \in \mathsf{P}\mathcal{X}\} = \sup\{\sum_{a\in|\mathcal{X}|} u_a u'_a \mid u' \in \mathsf{P}\mathcal{X}^\perp\}$.

Example 94. Let X be a measurable space. The set of all $\mathbb{R}^+$-valued measures on X is the cone $\mathsf{Meas}(X)$. The algebraic operations are defined in the usual "pointwise" way (e.g. $(\mu+\nu)(U) = \mu(U) + \nu(U)$) and the norm given by $\|\mu\|_{\mathsf{Meas}(X)} = \mu(X)$. Observe that such a cone has not to be of the shape $\widehat{\mathcal{X}}$. For instance, the cone $\mathsf{Meas}(\mathbb{R})$ associated with the Lebesgue σ-algebra on $\mathbb{R}$ (made of Borel sets) will be our interpretation of the ground type $\mathcal{R}$.

A type A of pRPCF will be associated with a complete cone $[\![A]\!]$ and a closed program of type A will be denoted as an element in the unit ball $\mathcal{B}[\![A]\!]$. The order completeness of $\mathcal{B}[\![A]\!]$ is crucial for defining the interpretation of the recursive programs, as usual.

Stable functions

A program taking inputs of type A and giving outputs of type B will be denoted as a map from $\mathcal{B}[\![A]\!]$ to $\mathcal{B}[\![B]\!]$. The difficulty was to find the right properties enjoyed by such functions in order to get a cartesian closed category, namely that the set of these functions generates a complete cone compatible with the cartesian structure (which will be the denotation of the type $A \to B$). A first attempt is to use the usual notion of Scott continuity.

Definition 95. Let P and Q be cones. A *bounded map* from P to Q is a function $f : \mathcal{B}P \to Q$ such that $f(\mathcal{B}P) \subseteq \mathcal{B}Q(\alpha)$ for some $\alpha \in \mathbb{R}^+$. The greatest lower bound of these α's is called *the norm of* f and is denoted as $\|f\|$. Let f be a bounded map from P to Q. Then, $\|f\| = \sup_{x\in\mathcal{B}P} \|f(x)\|_Q$ and $f(\mathcal{B}P) \subseteq \mathcal{B}Q(\|f\|)$.

A function $f : P \to Q$ is *linear* if it commutes with sums and scalar multiplication. A *Scott-continuous function* from a complete cone P to a complete cone Q is a bounded map from P to Q which is non-decreasing and commutes with the least upper bounds of non-decreasing sequences.

It turns out that Scott-continuity is a condition too weak for ensuring cartesian closeness. A precise analysis of this point led us to the conclusion that these functions have also to be *absolutely monotonic*. Given a function $f : \mathcal{B}P \to Q$ which is non-decreasing, we introduce a notation for the difference near $x \in \mathcal{B}P$: for all $u \in P$ such that $x + u \in \mathcal{B}P$,

$$\Delta f(x; u) = f(x + u) - f(x) \in \mathcal{B}Q$$

Definition 96. Let P be a cone and let $u \in \mathcal{B}P$. We define a new cone P_u (*the local cone of* P *at* u) as follows. We set $P_u = \{x \in P \mid \exists \varepsilon > 0\ \varepsilon x + u \in \mathcal{B}P\}$ and

$$\|x\|_{P_u} = \inf\{1/\varepsilon \mid \varepsilon > 0 \text{ and } \varepsilon x + u \in \mathcal{B}P\}$$

An *n-non-decreasing* function from P to Q is a function $f : \mathcal{B}P \to Q$ such that if $n = 0$, then f is non-decreasing and if $n > 0$, then f is non-decreasing and, for all $u \in \mathcal{B}P$, the function $\Delta f(\_; u)$ is $(n-1)$-non-decreasing from P_u to Q.

One says that f is *absolutely monotonic* if it is n-non-decreasing for all $n \in \mathbb{N}$.

This latter condition is usually expressed by saying that all derivatives are everywhere non-negative. However we define it as the non-negativity of iterated differences. Such non-differential definitions of absolute monotonicity have already been considered various times in classical analysis, see for instance [McMillan, 1954].

We call *stable functions* the Scott continuous and absolutely monotonic functions, allowing for a cpo-enriched cartesian closed structure over the category of cones:

Theorem 97. *Complete cones and stable functions consitute a category denoted* **Cstab**. *More explicitly,* $\mathbf{Cstab}(P, Q)$ *is the set of all functions* $f : \mathcal{B}P \to Q$ *which are absolutely monotonic from* P *to* Q*, Scott-continuous and satisfy* $f(\mathcal{B}P) \subseteq \mathcal{B}Q$.

We borrow the term of "stable function" from Berry's analysis of sequential computation [Berry, 1978]. In fact, our definition is deeply related with a notion of "probabilistic" sequentiality, as it rejects the "parallel or" (but not the "Gustave function").

11. There are actually two additional conditions which are not essential here.

Measurability

The notion of stability is however not enough to interpret all primitives of probabilistic functional programming. One should be able to integrate at least first-order functions in order to sample programs denoting probabilistic distributions, e.g. the denotation of the `let` construct.

The cone $\mathsf{Meas}(\mathbb{R})$ of $\mathbb{R}^+$-valued measures on $\mathbb{R}$ (Example 94) is the natural candidate to model the ground type $\mathcal{R}$. In particular, a real numeral will be interpreted as the Dirac measure δ_r. Consider now a closed term $\mathsf{let}(x, M, N)$ of type $\mathcal{R}$, so that $\vdash M : \mathcal{R}$ and $x : \mathcal{R} \vdash N : \mathcal{R}$. The term M will be associated with a measure μ in $\mathcal{B}(\mathsf{Meas}(\mathbb{R}))$, while N will be a stable function f from the whole $\mathcal{B}(\mathsf{Meas}(\mathbb{R}))$ to $\mathcal{B}(\mathsf{Meas}(\mathbb{R}))$. However, according to the operational semantics (Figure 3.13), N is supposed to get a real number r for x, and not a generic measure. This means that one has to compose f with a map $\delta : \mathbb{R} \to \mathcal{B}(\mathsf{Meas}(\mathbb{R}^+))$ defined by $\delta(r) = \delta_r$, so that $f \circ \delta : \mathbb{R} \to \mathcal{B}(\mathsf{Meas}(\mathbb{R}))$. Now, the natural way to pass μ to $f \circ \delta$ is then by the integral $\int_{\mathbb{R}}(f \circ \delta)(r)\mu(dr)$. However, this would be meaningful only in case $f \circ \delta$ is measurable, and this is not the case for all stable functions f.

The problem is that there are stable functions which are not measurable, so not Lebesgue integrable. We therefore equip the cones with a notion of measurability tests, inducing a notion of measurable paths in a cone.

Definition 98. We consider cones P equipped with a collection $(\mathsf{M}^n(P))_{n\in\mathbb{N}}$ where $\mathsf{M}^n(P) \subseteq (\bot\ P)^{\mathbb{R}^n}$ satisfies the following properties:

- $0 \in \mathsf{M}^n(P)$,
- if $\ell \in \mathsf{M}^n(P)$ and $h : \mathbb{R}^p \to \mathbb{R}^n$ is measurable then $\ell \circ h \in \mathsf{M}^p(P)$,
- and for any $\ell \in \mathsf{M}^n(P)$ any $x \in P$, the function $\mathbb{R}^n \to \mathbb{R}^+$ which maps $\vec{r}$ to $\ell(\vec{r})(x)$ is in $\mathcal{M}^n$, i.e. is a measurable map $\mathbb{R}^n \to \mathbb{R}^+$.

A cone P equipped with a family $(\mathsf{M}^n(P))_{n\in\mathbb{N}}$ satisfying the above conditions will be called a *measurable cone*. The elements of the sets $\mathsf{M}^n(P)$ will be called the *measurablility tests* of P.

Measurability tests have parameters in $\mathbb{R}^n$ and are not simply Scott-continuous linear forms for making it possible to prove that the evaluation function of the cartesian closed structure is well behaved.

Definition 99. Let P be a cone and let $n \in \mathbb{N}$. A *measurable path* of arity n in P is a function $\gamma : \mathbb{R}^n \to P$ such that: $\gamma(\mathbb{R}^n)$ is bounded in P and, for all $k \in \mathbb{N}$ and all $\ell \in \mathsf{M}^k(P)$, the function $\ell * \gamma : \mathbb{R}^{k+n} \to \mathbb{R}^+$ defined by $(\ell * \gamma)(\vec{r}, \vec{s}) = \ell(\vec{r})(\gamma(\vec{s}))$ is in $\mathcal{M}^{k+n}$, i.e. is a measurable map $\mathbb{R}^{k+n} \to \mathbb{R}^+$.

We use $\mathsf{Path}^n(P)$ for the set of measurable paths of P and $\mathsf{Path}^n_1(P)$ for the set of measurable paths which take their values in $\mathcal{B}P$.

In the case the cone is associated with a standard measurable space X, i.e. it is of the form $\mathsf{Meas}(X)$, then the measurability tests are the measurable sets of X. However, at higher-order types the definition is less immediate. The crucial point is that the measurable paths in $\mathsf{Meas}(X)$ are Lebesgue integrable, as expected. We then call *measurable* a stable map preserving measurable paths:

Definition 100. Let P and Q be measurable complete cones. A stable function from P to Q (remember that then f is actually a function $\mathcal{B}P \to Q$) is *measurable* if, for all $n \in \mathbb{N}$ and all $\gamma \in \mathsf{Path}^n_1(P)$, one has $f \circ \gamma \in \mathsf{Path}^n(Q)$.

We use $\mathbf{Cstab}_\mathsf{m}$ for the subcategory of $\mathbf{Cstab}$ whose morphisms are measurable.

This gives a cartesian category, denoted $\mathbf{Cstab}_\mathsf{m}$, with an object of morphisms from P to Q in $\mathbf{Cstab}_\mathsf{m}$ given by the pair $(P \Rightarrow_\mathsf{m} Q, \mathsf{Ev})$ (see Figure 3.14).

Theorem 101. *The category* $\mathbf{Cstab}_\mathsf{m}$ *is cartesian closed.*

Let us turn back to the initial problem of integrating maps appearing in the let constructions: the map $\delta : \mathbb{R} \to \mathsf{Meas}(\mathbb{R})$ such that $\delta(r) = \delta_r$ belongs to $\mathsf{Path}^1_1(\mathsf{Meas}(\mathbb{R}))$. Therefore, given $f \in \mathbf{Cstab}_\mathsf{m}(\mathsf{Meas}(\mathbb{R}), \mathsf{Meas}(\mathbb{R}))$, the function $f \circ \delta$ is a measurable path from $\mathbb{R}$ to $\mathsf{Meas}(\mathbb{R})$ which can be integrated.

This result can be generalized, to $f \in \mathbf{Cstab}_\mathsf{m}(\mathsf{Meas}(\mathbb{R}), P)$, where P is isomorphic to a cone of the form $Q \Rightarrow_\mathsf{m} \mathsf{Meas}(\mathbb{R})$ for some Q), so that we can define the semantics of the `let` construct for terms N with non empty contexts appearing in pRPCF (see Figure 3.15).

$$\prod_{i\in I} P_i = \{(x_i)_{i\in I} \text{ s.t. } \forall i \in I, x_i \in P_i\}, \qquad \|(x_i)_{i\in I}\|_{\prod_{i\in I} P_i} = \sup_{i\in I} \|x_i\|_{P_i}$$
$$\mathsf{M}^n(\prod_{i\in I} P_i) = \{\bigoplus_{i\in I} l_i \mid \forall i \in I\ l_i \in \mathsf{M}^n(P_i)\}, \qquad \text{with } \bigoplus_{i\in I} l_i(\vec{r})(x_i)_{n\in I} = \sum_{i\in I} l_i(\vec{r})(x_i)$$

(a) finite cartesian product (I finite set).

$$P \Rightarrow_{\mathsf{m}} Q = \{f : \mathcal{B}P \to Q \mid \exists \varepsilon > 0, \varepsilon f \in \mathbf{Cstab}_{\mathsf{m}}(P, Q)\},$$
$$\|f\|_{P\Rightarrow_{\mathsf{m}} Q} = \sup_{x\in\mathcal{B}P} \|f(x)\|_Q$$
$$\mathsf{M}^n(P \Rightarrow_{\mathsf{m}} Q) = \{\gamma \triangleright m \mid \gamma \in \mathsf{Path}^n(P), m \in \mathsf{M}^n(Q)\},$$
$$\text{with } \gamma \triangleright m(\vec{r})(f) = m(\vec{r})(f(\gamma(\vec{r})))$$

(b) object of morphisms

Figure 3.14 – The CCC structure of $\mathbf{Cstab}_{\mathsf{m}}$. The projections, pairing and the evaluation are defined as standard.

$$[\![x]\!](\vec{g}, a) = a \qquad [\![\underline{r}]\!]\vec{g} = \delta_r \qquad [\![\mathsf{sample}]\!]\vec{g} = \lambda_{[0,1]}$$
$$[\![\lambda x^A.M]\!]\vec{g} = a \mapsto [\![M]\!](\vec{g}, a) \qquad [\![MN]\!]\vec{g} = [\![M]\!]\vec{g}([\![N]\!]\vec{g})$$
$$[\![\mathsf{Y}M]\!]\vec{g} = \sup_n (([\![M]\!]\vec{g})^n 0)$$
$$[\![\underline{f}(M_1, \dots, M_n)]\!]\vec{g} = U \mapsto ([\![M_1]\!]\vec{g} \otimes \cdots \otimes [\![M_n]\!]\vec{g})(f^{-1}(U))$$
$$[\![\mathsf{if}\mathsf{z}(L, M, N)]\!]\vec{g} = ([\![L]\!]\vec{g}\{0\})\,[\![M]\!]\vec{g} + ([\![L]\!]\vec{g}(\mathbb{R}\setminus\{0\}))\,[\![N]\!]\vec{g}$$
$$[\![\mathsf{let}(x, M, N)]\!]\vec{g} = U \mapsto \int_{\mathbb{R}} [\![N]\!](\vec{g}, \delta_r)(U)\,[\![M]\!]\vec{g}(dr)$$

Figure 3.15 – Interpretation of pRPCF in $\mathbf{Cstab}_{\mathsf{m}}$. The terms are supposed typed as in Figure 3.12c, and $\vec{g} \in [\![\Gamma]\!]$, $a \in [\![A]\!]$.

3.5.4 Interpretation of pRPCF in $\mathbf{Cstab}_{\mathsf{m}}$

Let us underline that, although the definition of $\mathbf{Cstab}_{\mathsf{m}}$ and the proof of its cartesian closeness are not trivial, the denotation of the programs (Figure 3.15) is completely standard, extending the usual interpretation of PCF programs as Scott-continuous functions (see Plotkin [1977]).

The interpretation of pRPCF in $\mathbf{Cstab}_{\mathsf{m}}$ extends the standard model of PCF in a cpo-enriched category. The ground type $\mathcal{R}$ is denoted as the cone $\mathsf{Meas}(\mathbb{R})$ of bounded measures over $\mathbb{R}$ (see Example 94), the arrow $A \to B$ by the object of morphisms $[\![A]\!] \Rightarrow_{\mathsf{m}} [\![B]\!]$ and a sequence $A_1, \dots, A_n$ by the cartesian product $\prod_{i=1}^n [\![A_i]\!]$ (see Figure 3.14). The denotation of a judgement $\Gamma \vdash M : A$ is a morphism $[\![M]\!]^{\Gamma\vdash A} \in \mathbf{Cstab}_{\mathsf{m}}([\![\Gamma]\!], [\![A]\!])$, given in Figure 3.15 by structural induction on M. We omit the type exponent when clear from the context. Notice that if $\Gamma \vdash M : \mathcal{R}$, then for $\vec{g} \in [\![\Gamma]\!]$, $[\![M]\!]\vec{g}$ is a measure on $\mathbb{R}$.

The fact that the definitions of Figure 3.15 lead to morphisms in the category $\mathbf{Cstab}_{\mathsf{m}}$ results easily from the cartesian closeness of this category and from the algebraic and order theoretic properties of its objects.

3.5.5 Soundness and Adequacy

The soundness property states that the interpretation is invariant under reduction. In a non-deterministic case, this means that the semantics of a term is the sum of the semantics of all its possible one-step reducts (see Laird et al. [2013]). Here, the reduction is a stochastic kernel, so this sum becomes an integral,

$$\forall M \in \Lambda^p_{\mathcal{R}}, \quad [\![M]\!]^{\Gamma\vdash A} = \int_{\Lambda^p_{\mathcal{R}}} [\![t]\!]^{\Gamma\vdash A} \mathrm{Red}(M, dt). \tag{3.6}$$

Let us turn to adequacy: Let M be a closed term of ground type of pRPCF. Both the operational and the denotational semantics associate with M a distribution over $\mathbb{R}$ — the adequacy property states that these two distributions are actually the same:

Theorem 102 (Adequacy)**.** *Let $\vdash M : \mathcal{R}$. Then, for every measurable set $U \subseteq \mathbb{R}$,*

$$[\![M]\!]^{\vdash \mathcal{R}}(U) = \mathrm{Red}^{\infty}(M, \underline{U}),$$

where $\underline{U}$ is the set of numerals corresponding to the real numbers in U.

The proof is standard: the soundness property gives as a corollary that the "operational" distribution is bounded by the "denotational" one. The converse is obtained by using a suitable logical relation.

A major byproduct of this result is then to make it possible to reason about higher-order programs as functions between cones, which is quite convenient when working with programs acting on measures.

3.5.6 Examples

Numerals are associated with Dirac measures and a functional constant $\underline{f}$ yields the pushforward measure of the product of the measures denoting the arguments of $\underline{f}$. For example,

$$[\![\underline{+}(\underline{3}, \underline{2})]\!]^{\vdash \mathcal{R}} = U \mapsto \delta_3 \otimes \delta_2(\{(r_1, r_2) \text{ s.t. } r_1 + r_2 \in U\}) = \delta_5.$$

In order to make clear the difference between call-by-value and call-by-name reductions in a probabilistic setting, let us consider the following two terms:

$$M = (\lambda x.(x = x))\mathsf{sample}, \qquad N = \mathsf{let}(x, \mathsf{sample}, x = x).$$

Both are closed terms of type $\mathcal{R}$, "applying" the uniform distribution to $x \mapsto x = x$, the diagonal function. However, M implements a call-by-name application, whose reduction duplicates the probabilistic primitive before sampling the distribution, while the evaluation of N first samples a real number $\underline{r}$ and then duplicates it:

$$\begin{aligned} M &\to \mathsf{sample} = \mathsf{sample} \to \underline{r} = \underline{s} && \text{for any } r \text{ and } s, \\ N &\to \mathsf{let}(x, \underline{r}, x = x) \to \underline{r} = \underline{r} && \text{for any } r. \end{aligned}$$

The distribution associated with M by Red^{∞} is the Dirac δ_0, because $\mathrm{Red}^{\infty}(M, \underline{U}) = \mathrm{Red}^{3}(M, \underline{U}) = \lambda(\{(r, r) \text{ s.t. } r \in [0,1]\}) \times \delta_1(U) + \lambda(\{(r, s) \text{ s.t. } r \neq s, r, s \in [0,1]\}) \times \delta_0(U) = \delta_0(U)$. The last equality is because the diagonal set $\{(r, r) \text{ s.t. } r \in [0,1]\}$ has Lebesgue measure zero. This expresses that M evaluates to $\underline{0}$ (i.e. "false") with probability 1, although there are an uncountable number of reduction paths reaching $\underline{1}$. On the contrary, the distribution associated with N is δ_1: $\mathrm{Red}^{\infty}(N, \underline{U}) = \mathrm{Red}^{3}(M, \underline{U}) = \lambda([0,1]) \times \delta_1(U) = \delta_1(U)$, meaning that N always evaluates to $\underline{1}$ (i.e. "true").

The two terms implementing the diagonal have different semantics: for any measurable U of $\mathbb{R}$, for any $r, s \in U$, $r = s$ has value 0 or 1. Besides, the diagonal $\{(r, s) \text{ s.t. } r = s \in \{1\}\}$ in $[0,1]^2$ has measure 0, and its complementary $\{(r, s) \text{ s.t. } r = s \in \{0\}\}$ has measure 1. Thus,

$$[\![(\lambda x.(x = x))\mathsf{sample}]\!]^{\vdash \mathcal{R}}(U) = (\lambda_{[0,1]} \otimes \lambda_{[0,1]})\{(r, s) \text{ s.t. } r = s \in U\} = \delta_0(U).$$

and, $[\![\mathsf{let}(x, \mathsf{sample}, x = x)]\!]^{\vdash \mathcal{R}}(U) = \delta_1(U)$, as it is equal by definition to:

$$\int_{\mathbb{R}} (\delta_r \otimes \delta_r)\{(x, y) \text{ s.t. } x = y \in U\} \lambda_{[0,1]}(dr) = \int_{\mathbb{R}} \delta_1(U) \lambda_{[0,1]}(dr) = \delta_1(U).$$

Example 103 (Extended branching)**.** Let U be a measurable set of real numbers whose characteristic function χ_U is in $\mathcal{C}$, let $L \in \Lambda^p_{\mathcal{R}}$ and $M, N \in \Lambda^p_{\mathcal{R}}$ for $A = B_1 \Rightarrow \dots B_n \Rightarrow \mathcal{R}$. Then the term $\Gamma \vdash \mathsf{if}(L \in U, M, N) : A$, branching between M and N according to the outcome of L being in U, is a syntactic sugar[12] for

$$\mathsf{if}(L \in U, M, N) = \lambda x^{B_1} \dots \lambda x^{B_n}.\mathsf{ifz}(\underline{\chi_U}(L), Nx_1 \dots x_n, Mx_1 \dots x_n).$$

The construct `ifz` sums up the denotation of the two branches according to the probability that the first term evaluates to $\underline{0}$ or not. Given a measurable set $U \subseteq \mathbb{R}$, a closed term L of ground type and two closed terms M, N of a type A,

$$\begin{aligned} [\![\mathsf{if}(L \in U, M, N)]\!]^{\vdash A} &= ([\![\underline{\chi_U}(L)]\!]^{\vdash \mathcal{R}}(\mathbb{R} \setminus \{0\}))\, [\![M]\!]^{\vdash A} + ([\![\underline{\chi_U}(L)]\!]^{\vdash \mathcal{R}}(\{0\}))\, [\![N]\!]^{\vdash A} \\ &= ([\![L]\!]^{\vdash \mathcal{R}}(U))\, [\![M]\!] + ([\![L]\!]^{\vdash \mathcal{R}}(\mathbb{R} \setminus U))\, [\![N]\!]. \end{aligned}$$

12. The swap between M and N is due to fact that `ifz` is the test to zero.

Similarly, the `let` constructor can be extended to any output type $A = B_1 \Rightarrow \dots B_n \Rightarrow \mathcal{R}$. Given $M \in \Lambda^p_{\mathcal{R}}$ and $N \in \Lambda^p_{\mathcal{R}}$, we denote by $\mathsf{let}(x, M, N)$ the term $\lambda x^{B_1} \dots \lambda x^{B_n}.\mathsf{let}(x, M, N x_1 \dots x_n)$ which is in $\Lambda^p_{\mathcal{R}}$. However we do not know in general how to extend the type of the bound variable x to higher types in this model. The issue is clear at the denotational level, where the `let` construction is expressed with an integral (see Figure 3.15). With each ground type, we associate a positive cone $\mathsf{Meas}(X)$ which is generated by a measurable space X. At higher types, the associated cones do not have to be generated by measurable spaces.

Notice that, because of this restriction on the type of the bound variable x, our `let` constructor does not allow to embed into our language the full call-by-value PCF.

Example 104 (Probability distributions)**.**

The Bernoulli distribution takes the value 1 with some probability p and the value 0 with probability $1-p$. It can be expressed as the term of type $\mathcal{R} \to \mathcal{R}$, taking the parameter p as argument and testing whether sample draws a value within $[0, p]$,

$$\texttt{bernoulli} = \lambda p.\mathsf{let}(x, \mathsf{sample}, \underline{x \leq p}) \quad \text{and} \quad [\![\texttt{bernoulli}\ \underline{p}]\!]^{\vdash \mathcal{R}} = p\delta_1 + (1-p)\delta_0$$

Indeed, if U is measurable, then $[\![\texttt{bernoulli}\ \underline{p}]\!]^{\vdash \mathcal{R}}(U) = \int_{\mathbb{R}} \delta_r \otimes \delta_p(\{(x,y) \text{ s.t. } x \leq y \in U\})\lambda_{[0,1]}(dr)$, this latter being equal to $\lambda_{[0,1]}([0,p])\delta_1(U) + \lambda_{[0,1]}((p,1])\delta_0(U) = p\delta_1(U) + (1-p)\delta_0(U)$.

The exponential distribution at rate 1 is specified by its density e^{-x}. The exponential distribution exp computes the probability that an exponential random variable belongs to U. It can be implemented as the term exp of type $\mathcal{R}$ by the inversion sampling method:

$$\exp = \mathsf{let}(x, \mathsf{sample}, \underline{-\log(x)}) \quad \text{and} \quad [\![\exp]\!]^{\vdash \mathcal{R}}(U) = \int_{\mathbb{R}^+} \chi_U(s)\mathrm{e}^{-s}\lambda(ds)$$

Indeed, $[\![\mathsf{let}(x, \mathsf{sample}, \underline{-\log(x)})]\!]^{\vdash \mathcal{R}}(U) = \int_{\mathbb{R}} \delta_r(\{x \text{ s.t. } -\log x \in U\})\lambda_{[0,1]}(dr)$, which, by substitution of $r = \mathrm{e}^{-s}$, is equal to

$$\int_{\mathbb{R}} \chi_U(-\log r)\lambda_{[0,1]}(dr) = \int_{\mathbb{R}^+} \chi_U(s)\mathrm{e}^{-s}\lambda(ds).$$

The standard normal distribution (gaussian with mean 0 and variance 1) is defined by its density $\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}$. We use the well-known method of Box and Muller [1958] to encode it:

$$\texttt{normal} = \mathsf{let}(x, \mathsf{sample}, \mathsf{let}(y, \mathsf{sample}, \underline{(-2\log(x))^{\frac{1}{2}}}\ \underline{\cos(2\pi y)})).$$

We compute the semantics of `normal` and check that we get a normal distribution:

$$\begin{aligned}
[\![\texttt{normal}]\!]^{\vdash \mathcal{R}}(U) &= [\![\mathsf{let}(x, \mathsf{sample}, \mathsf{let}(y, \mathsf{sample}, \underline{(-2\log(x))^{\frac{1}{2}}}\ \underline{\cos(2\pi y)}))]\!]^{\vdash \mathcal{R}}(U) \\
&= \int_{\mathbb{R}^2} \chi_U(\sqrt{-2\log u}\cos(2\pi v))\lambda_{[0,1]}(du)\lambda_{[0,1]}(dv) \\
&= \tfrac{1}{2\pi}\int_{\mathbb{R}^2} \chi_U(x)\mathrm{e}^{-\frac{(x^2+y^2)}{2}}\lambda(dx)\lambda(dy).
\end{aligned}$$

By polar substitution with $x = \sqrt{-2\log u}\cos(2\pi v)$, $y = \sqrt{-2\log u}\cos(2\pi v)$, we get:

$$[\![\texttt{normal}]\!]^{\vdash \mathcal{R}}(U) = \tfrac{1}{\sqrt{2\pi}}\int_U \mathrm{e}^{-\frac{x^2}{2}}\lambda(dx)$$

The Gaussian distribution. We can encode the Gaussian distribution as a function of the expected value x and standard deviation σ by

$$\texttt{gauss} = \lambda x \lambda \sigma.\ \mathsf{let}(y, \texttt{normal}, (\sigma y)\underline{+x}).$$

We compute $[\![\texttt{gauss}\ \underline{r}\ \underline{\sigma}]\!]^{\vdash \mathcal{R}}(U) = \frac{1}{\sqrt{2\pi}}\int_{\mathbb{R}} \chi_U(\sigma\, y + r)\mathrm{e}^{-\frac{y^2}{2}}\lambda(dy) = \frac{1}{\sigma\sqrt{2\pi}}\int_U \mathrm{e}^{-(\frac{z-r}{\sigma})^2}\lambda(dz)$.

The next example uses rejection sampling, which we already used in the discrete case to encode a uniform law in pPCF (see Example 3.3.2).

Example 105 (Conditioning)**.** Let U be a measurable set of real numbers such that $\chi_U \in \mathcal{C}$. We define a term $\mathtt{observe}(U)$ of type $\mathcal{R} \Rightarrow \mathcal{R}$, taking a term M and returning the renormalization of the distribution of M on the only samples that satisfy U:

$$\mathtt{observe}(U) = \lambda m.\mathsf{fix}(\lambda y.\mathsf{let}(x, m, \mathsf{if}(x \in U, x, y))).$$

This corresponds to the usual way of implementing conditioning by rejection sampling: the evaluation of $(\mathtt{observe}(U))M$ samples a real r from M; if $r \in U$ holds, then the program returns $\underline{r}$, otherwise it iterates the procedure. Notice that $\mathtt{observe}(U)M$ makes a crucial use of sampling. The program $\lambda m.\mathsf{fix}(\lambda y.\mathsf{if}(m \in U, m, y))$ has a different behavior, because the two occurrences of m correspond in this case to two independent random variables.

Let M be a closed term of type $\mathcal{R}$. We compute the semantics of $\vdash (\mathtt{observe}(U))M : \mathcal{R}$ by using soundness. Since $\mathtt{observe}(U)M \to^* \mathsf{let}(x, M, \mathsf{if}(x \in U, x, (\mathtt{observe}(U))M))$, we get by soundness that for all $V \subseteq \mathbb{R}$ measurable,

$$\begin{aligned}
[\![(\mathtt{observe}(U))M]\!](V) &= \int_{\mathbb{R}} [\![\mathsf{if}(x \in U, x, (\mathtt{observe}(U))M)]\!]^{x:\mathcal{R}\vdash\mathcal{R}}(\delta_r)(V)\,[\![M]\!](dr) \\
&= \int_{\mathbb{R}} (\delta_r(U)\,\delta_r(V) + (\delta_r(\mathbb{R}\setminus U))\,([\![(\mathtt{observe}(U))M]\!](V)))\,[\![M]\!](dr) \\
&= \int_{\mathbb{R}} (\chi_U(r)\,\chi_V(r))\,[\![M]\!](dr) + ([\![(\mathtt{observe}(U))M]\!](V)) \int_{\mathbb{R}} \chi_{\mathbb{R}\setminus U}(r)\,[\![M]\!](dr).
\end{aligned}$$

The last equality holds since $[\![(\mathtt{observe}(U))M]\!]$ does not depend on r. If M represents a probability distribution, so that $[\![M]\!](U) = 1 - [\![M]\!](\mathbb{R}\setminus U)$ and if moreover $[\![M]\!](U) \neq 0$, this equation gives the conditional probability:

$$[\![(\mathtt{observe}(U))M]\!](V) = \frac{\int_{\mathbb{R}}(\chi_U(r)\,\chi_V(r))\,[\![M]\!](dr)}{1 - \int_{\mathbb{R}} \chi_{\mathbb{R}\setminus U}(r)\,[\![M]\!](dr)} = \frac{[\![M]\!](V \cap U)}{[\![M]\!](U)}$$

If $[\![M]\!](U) = 0$, then the denotation of the fixpoint is $[\![(\mathtt{observe}(U))M]\!] = 0$. Indeed, we check that $([\![\lambda y\,\mathsf{let}(x, M, \mathsf{if}(x \in U, x, y))]\!])^n 0 = 0$. By adequacy, the program then loops with probability 1 when $[\![M]\!](U) = 0$.

Now, consider the term $O = \lambda m.\mathsf{Y}(\lambda y.\mathsf{if}(m \in U, m, y))$ presented in Example 105 as a wrong implementation of $\mathtt{observe}(U)$. Since $(O)M \to^* \mathsf{if}(M \in U, M, (O)M)$, assuming that $[\![M]\!]$ is a probability distribution and V a measurable set, one gets with a similar reasoning that, in case $[\![M]\!]U \neq 0$, then $[\![(O)M]\!](V) = ([\![M]\!](V)\,[\![M]\!](U))/[\![M]\!](U) = [\![M]\!](V)$. As before, if $[\![M]\!](U) = 0$, then $[\![(O)M]\!] = 0$.

The next example necessitates to perform independent copies of a real random variable.

Example 106 (Monte Carlo Simulation)**.** We use the Monte Carlo method to compute the n-th estimate and approximate the expected value of a measurable function.

By definition, the expected value of a Lebesgue integrable function f with respect to a distribution μ is $\int_{\mathbb{R}} f(x)\mu(dx)$. The Monte Carlo method relies on the law of large numbers: if $\mathbf{x_1}, \ldots, \mathbf{x_n}$ are independent and identically distributed random variables of equal probability distribution μ, then the n-th estimate $\frac{1}{n}(f(\mathbf{x_1}) + \cdots + f(\mathbf{x_n}))$ converges almost surely to $\int_{\mathbb{R}} f(x)\mu(dx)$, as n tends to ∞.

Let $n \in \mathbb{N}$. We encode the n-th estimate combinator of type $(\mathcal{R} \Rightarrow \mathcal{R}) \Rightarrow \mathcal{R} \Rightarrow \mathcal{R}$ by:

$$\mathtt{expectation}_n = \lambda f.\lambda m.\tfrac{1}{n}(\overbrace{f(m)\underline{+}\ldots\underline{+}f(m)}^{n \text{ times}})$$

Notice that it is crucial here that the variable m has n occurrences representing n independent random variables, this being in contrast with Example 105.

Let us compute its semantics. Let $\underline{f} \in \mathcal{C}$, M be a term of type $\mathcal{R}$ and $U \subseteq \mathbb{R}$ be measurable. Then,

$$[\![\mathtt{expectation}_n \underline{f}\, M]\!]^{\vdash\mathcal{R}}(U) = [\![M]\!]^{\vdash\mathcal{R}} \otimes \cdots \otimes [\![M]\!]^{\vdash\mathcal{R}}(\{(x_1, \cdots, x_n) \text{ s.t. } \tfrac{1}{n}(f(x_1) + \cdots + f(x_n)) \in U\}).$$

This is exactly the law of $\frac{1}{n}(f(\mathbf{x}_1)+\cdots+f(\mathbf{x}_n))$ where the $\mathbf{x}_i$'s are independent and identically distributed random variables of law $[\![M]\!]^{\vdash\mathcal{R}}$. Thus, we have correctly encoded the n-th estimate of the expected value of f with respect to μ.

When the preceding Example 106 cannot be applied, because, for instance, it is hard to compute independent and identically distributed random variables following law μ. Then, we can replace them by a Markov-Chain $(\mathbf{x}_n)$ with invariant measure μ. The purpose of the Metropolis-Hasting is to compute this Markov-Chain.

Example 107 (Metropolis-Hasting)**.** We present an encoding of the Metropolis-Hasting algorithm which is used in statistical inference and physics. This algorithm has been studied in Borgström et al. [2016] from a syntactical viewpoint and in Scibior et al. [2018] in the semantics of Quasi-Borel Spaces (see Heunen et al. [2017]).

Let μ be a distribution with density π with respect to the Lebesgue measure (i.e. for all $U \subseteq \mathbb{R}$ measurable, $\mu(U) = \int_U \pi(x)\lambda(dx)$). Assume that we know π up to a normalization factor (which can be hard to compute has it needs to evaluate an integral). The Metropolis-Hasting algorithms generates a Markov-Chain which is designed to be easy to compute and to converge quickly to a random variable that follows the distribution law μ. We first choose a suitable conditional density function $g(\cdot, x)$ such that $g(x, y) = g(y, x)$ (for instance, a gaussian distribution with mean x). Let x_0 be such that $\pi(x_0) > 0$. Then, the steps of the algorithms are:

1. Initial state is $x := x_0$
2. Sample next state y according to probability density $g(\cdot, x)$
3. Compute the acceptance probability $\alpha(x, y) = \frac{\pi(y)}{\pi(x)}$ and sample uniformly u in the interval $[0, 1]$. Then, if $u \leq \alpha(x, y)$, then accept it and assign $x := y$, otherwise, if $u > \alpha(x, y)$, then reject it and leave x unchanged.

Intuitively, if y is better than x, then we keep it, otherwise, we also keep it but with probability $\alpha(x, y)$.

We assume to be given a term $G : \mathcal{R} \to \mathcal{R}$ encoding the conditional density function g (for instance, $G = \lambda x.\, \mathtt{gauss}\, x \underline{\sigma}$ if we choose the gaussian distribution with standard deviation σ). Then, we encode the Metropolis-Hasting algorithm as $\mathtt{MH}$ such that if n is an integer, then $\mathtt{MH}\underline{n}x_0$ is the n-th iteration of the algorithm, with x_0 as starting point:

$$\mathtt{MH} = \mathsf{fix}\ \lambda h \lambda n \lambda x_0\, \mathsf{ifz}(n, x_0, \mathsf{let}(x, h(n-1)x_0, \mathsf{let}(y, G\, x, \mathsf{let}(z, \mathtt{bernoulli}(\underline{\alpha}(x, y)), \mathsf{ifz}(z, x, y)))))$$

We want to prove that this program encodes correctly the Metropolis-Hasting algorithm. Let $\mathbf{x}_n$ be the result of the evaluation of $\mathtt{MH}\underline{n}x_0$, following the operational semantics described in Figure 3.13. Let $U \subseteq \mathbb{R}$ be measurable. We need to prove that $\mathbf{x}_n$ is a Markov Chain (see §3.5.7) whose invariant measure is π and whose initial state is x_0 such that $\pi(x_0) > 0$. The measure of $\mathbf{x}_n$, defined as the probability that $\mathbf{x}_n \in U$, is given by $\mathrm{Red}(\mathtt{MH}(\underline{n})x_0, U)$ (see the Adequacy Lemma 102).

Now, we want to compute $\mathrm{Red}(\mathtt{MH}(\underline{n})x_0, \cdot)$. We first apply the Adequacy Theorem 102: for all $n \in \mathbb{N}$, $\mathrm{Red}(\mathtt{MH}(\underline{n})x_0, U) = [\![\mathtt{MH}(\underline{n})x_0]\!]^{\vdash\mathcal{R}}(U)$. Then, we apply the Soundness Theorem 3.6. Indeed, since

$$\begin{aligned}
\mathtt{MH}\underline{0}x_0 &\xrightarrow{\delta} x_0 \\
\mathtt{MH}(\underline{n+1})x_0 &\xrightarrow{\delta} \mathsf{let}(x, \mathtt{MH}nx_0, \mathsf{let}(y, G\, x, \mathsf{let}(z, \mathtt{bernoulli}(\underline{\alpha}(x, y)), \mathsf{ifz}(z, x, y))))
\end{aligned}$$

and thanks to the interpretation of terms defined in Figure 3.15, we deduce:

$$\begin{aligned}
\mathrm{Red}(\mathtt{MH}(\underline{0})x_0, U) &= \delta_{x_0}(U) \\
\mathrm{Red}(\mathtt{MH}(\underline{n+1})x_0, U) &= [\![\mathtt{MH}(\underline{n+1})x_0]\!]^{\vdash\mathcal{R}}(U) \\
&= [\![\mathsf{let}(x, \mathtt{MH}\underline{n}x_0, \mathsf{let}(y, G\, x, \mathsf{let}(z, \mathtt{bernoulli}(\underline{\alpha}(x, y)), \mathsf{ifz}(z, x, y))))]\!]^{\vdash\mathcal{R}}(U) \\
&= \int_{\mathbb{R}} [\![\mathsf{let}(y, G\, x, \mathsf{let}(z, \mathtt{bernoulli}(\underline{\alpha}(x, y)), \mathsf{ifz}(z, x, y)))]\!]^{x:\mathcal{R}\vdash\mathcal{R}}(\delta_r)(U)\ [\![\mathtt{MH}\underline{n}x_0]\!]^{\vdash\mathcal{R}}(dr) \\
&= \int_{\mathbb{R}} P_{\mathtt{MH}}(r, U)\, \mathrm{Red}(\mathtt{MH}\underline{n}x_0, dr),
\end{aligned}$$

This last equality shows that $\mathbf{x}_n$ is a Markov-Chain whose law is defined with respect to the kernel $P_{\mathtt{MH}}(r, U)$. Recall that Gx is a term of type $\mathcal{R}$ whose semantics has density function $g(\cdot, x)$, meaning that $[\![Gx]\!]^{\vdash\mathcal{R}}(U) = \int_U g(y, x)\lambda(dy)$. Thus, we now compute

$$\begin{aligned}
P_{\mathtt{MH}}(r, U) &= [\![\mathsf{let}(y, G\, x, \mathsf{let}(z, \mathtt{bernoulli}(\underline{\alpha}(x, y)), \mathsf{ifz}(z, x, y)))]\!]^{x:\mathcal{R}\vdash\mathcal{R}}(\delta_r)(U) \\
&= \int_{\mathbb{R}} [(1 - \alpha(r, t))\delta_r(U) + \alpha(r, t)\delta_t(U)]g(t, r)\lambda(dt) \\
&= \delta_r(U)\left(1 - \int_{\mathbb{R}} \alpha(r, t)g(t, r)\lambda(dt)\right) + \int_U \alpha(r, t)g(t, r)\lambda(dt)
\end{aligned}$$

This is the classical form of the Metropolis-Hasting kernel associated to the algorithm described above.

Let us show that π is the density of the invariant measure of the Markov-Chain $\mathbf{x}_n$. For this, we need to prove that $\int_{\mathbb{R}} P_{\mathtt{MH}}(r,U)\pi(r)\lambda(dr) = \int_U \pi(r)\lambda(dr)$. This results from the Fubini theorem and from $\alpha(r,t)\pi(r) = \alpha(t,r)\pi(t)$ and $g(r,t) = g(t,r)$.

To sum up, we have proved that our encoding of the n-th step of the Metropolis-Hasting algorithm evaluates to a random variable $\mathbf{x}_n$. Besides, $(\mathbf{x}_n)$ is a Markov-Chain whose invariant measure has density π. Thus, it converges to a random variable $\mathbf{x}$ whose law has also density π. This proves that our encoding is correct.

3.5.7 Some measure theory

This paragraph gathers the probability theory that is needed to understand the above development.

A *σ-algebra* Σ_X on a set X is a family of subsets of X that is non-empty, closed under complements and countable unions and such that $\emptyset \in \Sigma_X$. A *measurable space* is a pair (X, Σ_X) of a set X equipped with a σ-algebra Σ_X. A *measurable set* of (X, Σ_X) is an element of Σ_X. From now on, we will denote a measurable space (X, Σ_X) simply by its underlying set X, whenever the σ-algebra is clear or irrelevant. We consider $\mathbb{R}$ and $\mathbb{R}^+$ as measurable spaces equipped with the Lebesgue σ-algebra, generated by the open intervals. A *bounded measure* on a measurable space X is a map $\mu : \Sigma_X \to \mathbb{R}^+$ satisfying $\mu(\biguplus_{i\in I} S_i) = \sum_{i\in I} \mu(S_i)$ for any countable family $\{S_i\}_{i\in I}$ of disjoint sets in Σ_X. We call μ a *probability* (resp. *subprobability*) *measure*, whenever $\mu(X) = 1$ (resp. $\mu(X) \leq 1$). When μ is a measure on $\mathbb{R}^n$, we often call it a *distribution*.

A *measurable function* $f : (X, \Sigma_X) \to (Y, \Sigma_Y)$ is a function $f : X \to Y$ such that $f^{-1}(U) \in \Sigma_X$ for every $U \in \Sigma_Y$. The *pushforward measure* $f_*\mu$ from a measure μ on X along a measurable map f is defined as $(f_*\mu)(U) = \mu(f^{-1}(U))$, for every $U \in \Sigma_Y$.

These notions have been introduced in order to define the *Lebesgue integral* $\int_X f(x)\mu(dx)$ of a generic measurable function $f : X \to \mathbb{R}$ with respect to a measure μ over X. This paper uses only basic facts about the Lebesgue integral which we do not detail here.

Measures are special cases of kernels. A *bounded kernel* K from X to Y is a function $K : X \times \Sigma_Y \to \mathbb{R}^+$ such that: (i) for every $x \in X$, $K(x,\cdot)$ is a bounded measure over Y; (ii) for every $U \in \Sigma_Y$, $K(\cdot, U)$ is a measurable map from X to $\mathbb{R}^+$. A *stochastic kernel* K is a kernel such that $K(x,\cdot)$ is a sub-probability measure for every $x \in X$. Notice that a bounded measure (resp. sub-probability measure) μ over X can be seen as a particular bounded kernel (resp. stochastic kernel) from the singleton measurable space $(\{\star\}, \{\emptyset, \{\star\}\})$ to X.

A random variable $\mathbf{x} : (X, \Sigma, \mu) \to \mathbb{R}$ is a real valued Borel measurable function.

A stochastic process is an indexed family of random variables $\mathbf{x}_n : X \to \mathbb{R}$. Let $\mu(U|x_1, \ldots, x_n)$ denotes the conditional probability that x_{n+1} is in A given that $\mathbf{x}_i$ is x_i. Then a Markov chain is a stochastic process such that $\mu(U|x_1, \ldots, x_n) = \mu(U|x_n)$, meaning that the probability of $\mathbf{x}_{n+1}$ only depends on the value of $\mathbf{x}_n$.

Categorical Approach. We use two categories having measurable spaces as objects, denoted respectively **Meas** and **Kern**.

The category **Meas** has measurable functions as morphisms. This category is cartesian (but not cartesian closed), the *cartesian product* $(X, \Sigma_X) \times (Y, \Sigma_Y)$ of (X, Σ_X) and (Y, Σ_Y) is $(X \times Y, \Sigma_X \otimes \Sigma_Y)$, where $X \times Y$ is the set-theoretic product and $\Sigma_X \otimes \Sigma_Y$ is the σ-algebra generated by the rectangles $U \times V$, where $U \in \Sigma_X$ and $V \in \Sigma_Y$. It is easy to check that the usual projections are measurable maps, as well as that the set-theoretic pairing $\langle f, g\rangle$ of two functions $f : Z \to X$, $g : Z \to Y$ is a measurable map from Z to $X \times Y$, whenever f, g are measurable.

The category **Kern** has stochastic kernels as morphisms[13]. Given a stochastic kernel H from X to Y and K from Y to Z, the *kernel composition* $K \circ H$ is a stochastic kernel from X to Z defined as, for every $x \in X$ and $U \in \Sigma_Z$:

$$(K \circ H)(x, U) = \int_Y K(y, U)H(x, dy). \tag{3.7}$$

Notice that the above integral is well-defined because $H(x,\cdot)$ is a stochastic measure from condition (i) on kernels and $K(\cdot, U)$ is a measurable function from condition (ii). A simple application of Fubini's theorem gives the associativity of the kernel composition. The *identity kernel* is the function mapping (x, U) to 1 if $x \in U$ and to 0 otherwise.

Unlike **Meas**, we consider a tensor product $\otimes$ in **Kern** which is a symmetric monoidal product but not the cartesian product[14]. The action of $\otimes$ over the objects X, X' is defined as the cartesian product

13. One can well define the category of bounded kernels also, but this is not used here.
14. Indeed, **Kern** has cartesian products, but we will not use them.

in **Meas**, so that we still denote it as $X \times X'$. The tensor of a kernel K from X to Y and K' from X' to Y' is the kernel $K \otimes K'$ given as follows, for $(x, x') \in X \times X'$ and $U \in \Sigma_Y$, $U' \in \Sigma_{Y'}$:

$$K \otimes K'((x, x'), U \times U') = K(x, U)K'(x', U') \tag{3.8}$$

Notice that **Kern** is not closed with respect to $\otimes$. Recall that a measure can be seen as a kernel from the singleton measurable space, so that Equation (3.8) defines also a *tensor product* between measures over resp. Y and Y'.

The category **Kern** has also *countable coproducts*. Given a countable family $(X_i, \Sigma_i)_{i \in I}$ of measurable spaces, the coproduct $\coprod_{i \in I}(X_i, \Sigma_i)$ has as underlining set the disjoint union $\cup_{i \in I} X_i \times \{i\}$ of the X_i's, and as the σ-algebra the one generated by $\cup_{i \in I} U_i \times \{i\}$ disjoint union of $U_i \in \Sigma_i$. The injections ι_j from X_j to $\coprod_{i \in I} X_i$ are defined as $\iota_i(x, \cup_{j \in I} U_j \times \{j\}) = \chi_{U_i}(x)$. Given a family K_i from X_i to Y, the copairing $[K_i]_{i \in I}$ from $\coprod_{i \in I} X_i$ to Y is defined by $[K_i]_{i \in I}((x, j), U) = K_j(x, U)$.

Actually, the categories **Meas** and **Kern** can be related in a very similar way as the relation between the categories **Set** (of sets and functions) and **Rel** (of sets and relations). In fact, **Kern** corresponds to the Kleisli category of the so-called Giry's monad over **Meas** [Giry, 1982], exactly as the category **Rel** of relations is the Kleisli category of the powerset monad over **Set** (see [Panangaden, 1999]). Since this paper does not use this construction, we do not detail it.

3.6 Conclusion

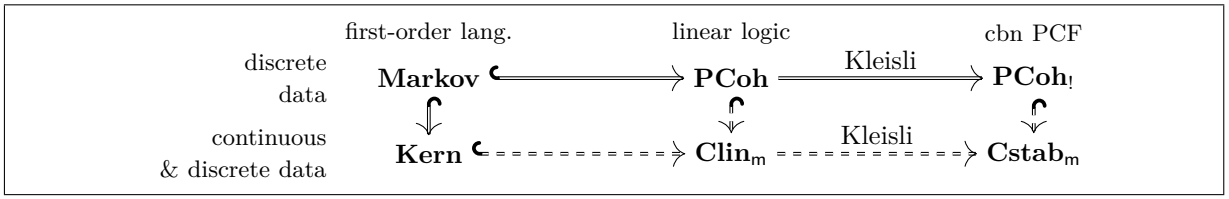


Figure 3.16 – Relationship between probabilistic coherence categories and the category of measurable cones and stable, measurable morphisms. Dashed arrows are still conjectures.

Let us sketch the relations, sketched in Figure 3.16, between the two categories we have presented in this chapter: the category $\mathbf{PCoh}_!$ of probabilistic coherence spaces and analytic functions which has been the starting point of our approach and the category $\mathbf{Cstab}_\infty$ of measurable cones and stable, measurable functions presented in the last section. The two categories give models of the functional primitives (PCF-like languages), but $\mathbf{PCoh}_!$ is restricted to discrete data types, while $\mathbf{Cstab}_\infty$ extends the model to continuous types. We guess this extension to be conservative, hence the arrow is hooked but just dashed.

There are many open questions in this area that are listed below:

- *How to decompose the category* $\mathbf{Cstab}_\infty$ *into a model of Linear Logic, that is a category* $\mathbf{Clin}_\infty$ *of measurable cones and continuous, linear, measurable functions; together with an exponential comonad ?*
- *Does the stable, measurable functions admits a decomposition as analytic functions, that we could exploit to apply our Full-Abstraction proof technique ?*

  A step forward has been taken in this direction in Crubillé [2018] where morphisms of Probabilistic Coherent Spaces are proved to be exactly stable measurable morphisms.
- *Differential programming relies on the optimization of statstical algorithms through gradient descent. How to combine differential extensions of Linear Logic and stable functions that are infinitely differentiable (inside the unit ball) in order to give a semantical account of this topical subject ?*

3.7 Bibliography

An explicit formula for the free exponential modality of linear logic. *Mathematical Structures in Computer Science*, page 1–34, 2017. 64, 68

Roberto Amadio and Pierre-Louis Curien. *Domains and lambda-calculi*, volume 46 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1998. 83

Tsuyoshi Andô. On fundamental properties of a banach space with a cone. *Pacific Journal of Mathematics*, 12(4):1163–1169, 1962. 97

Robert J. Aumann. Borel structures for function spaces. *Illinois J. Math.*, 5(4):614–630, 12 1961. 63

Gérard Berry. Stable models of typed lambda-calculi. In *Automata, Languages and Programming, Fifth Colloquium, Udine, Italy, July 17-21, 1978, Proceedings*, volume 62 of *Lecture Notes in Computer Science*, pages 72–89. Springer, 1978. 62, 98

Gavin Bierman. What is a categorical model of intuitionistic linear logic? In Mariangiola Dezani-Ciancaglini and Gordon D. Plotkin, editors, *Proceedings of the second Typed Lambda-Calculi and Applications conference*, volume 902 of *Lecture Notes in Computer Science*, pages 73–93. Springer-Verlag, 1995. 66

Johannes Borgström, Ugo Dal Lago, Andrew D. Gordon, and Marcin Szymczak. A lambda-calculus foundation for universal probabilistic programming. In Jacques Garrigue, Gabriele Keller, and Eijiro Sumii, editors, *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, ICFP 2016, Nara, Japan, September 18-22, 2016*, pages 33–46. ACM, 2016. 95, 104

G. E. P. Box and Mervin E. Muller. A note on the generation of random normal deviates. *Ann. Math. Statist.*, 29(2):610–611, 06 1958. doi: 10.1214/aoms/1177706645. 102

Antonio Bucciarelli, Alberto Carraro, Thomas Ehrhard, and Giulio Manzonetto. Full Abstraction for Resource Calculus with Tests. In Marc Bezem, editor, *CSL*, volume 12 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011. 84

Simon Castellan, Pierre Clairambault, Hugo Paquet, and Glynn Winskel. The concurrent game semantics of probabilistic PCF. In *LICS*, pages 215–224. ACM, 2018. 63

Raphaëlle Crubillé. Probabilistic stable functions on discrete cones are power series. In *LICS*, pages 275–284. ACM, 2018. 106

Raphaelle Crubillé, Thomas Ehrhard, Michele Pagani, and Christine Tasson. The Free Exponential Modality of Probabilistic Coherence Spaces. In J. Esperanza and A. Murawski, editors, *Proceedings of the 20th International Conference on Foundations of Software Science and Computation Structures, FOSSACS 2017*. ARCoSS, 2017. 64, 68

Vincent Danos and Thomas Ehrhard. Probabilistic coherence spaces as a model of higher-order probabilistic computation. *Information and Computation*, 152(1):111–137, 2011. 61, 65, 66, 68, 70, 74, 78, 79, 85, 95

Thomas Ehrhard. Call-by-push-value from a linear logic point of view. In *ESOP*, volume 9632 of *Lecture Notes in Computer Science*, pages 202–228. Springer, 2016. 64, 71, 73, 90

Thomas Ehrhard. An introduction to differential linear logic: proof-nets, models and antiderivatives. *Mathematical Structures in Computer Science*, 28(7):995–1060, 2018. 72

Thomas Ehrhard and Christine Tasson. Probabilistic call by push value. *Logical Methods in Computer Science*, 2018. Accepted for publication, under minor revisions. 61, 64, 65, 85, 91, 92

Thomas Ehrhard, Michele Pagani, and Christine Tasson. The computational meaning of probabilistic coherence spaces. In *LICS*, pages 87–96. IEEE Computer Society, 2011. 63, 74, 76

Thomas Ehrhard, Michele Pagani, and Christine Tasson. Probabilistic coherence spaces are fully abstract for probabilistic PCF. In *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. 61, 63

Thomas Ehrhard, Michele Pagani, and Christine Tasson. Full abstraction for probabilistic PCF. *Journal of the ACM*, 65(4):23:1–23:44, 2018a. 61, 64, 65, 78

Thomas Ehrhard, Michele Pagani, and Christine Tasson. Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming. In *POPL*, pages 59:1–59:28. ACM, 2018b. 62, 64

Martin Escardó. Pcf extended with real numbers. *Theoretical Computer Science*, 162(1):79 – 115, 1996. 94

Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987. 69, 92

Jean-Yves Girard. Normal functors, power series and the λ-calculus. *Annals of Pure and Applied Logic*, 37:129–177, 1988. 61

Jean-Yves Girard. Between logic and quantic: a tract. In Thomas Ehrhard, Jean-Yves Girard, Paul Ruet, and Philip Scott, editors, *Linear Logic in Computer Science*, volume 316 of *London Mathematical Society Lecture Notes Series*, pages 346–381. Cambridge University Press, 2004. 61

Michèle Giry. *A categorical approach to probability theory*, pages 68–85. Springer Berlin Heidelberg, Berlin, Heidelberg, 1982. 63, 106

Noah D. Goodman and Joshua B. Tenenbaum. Probabilistic models of cognition. http://probmods.org, 2014. 63

Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Joshua B. Tenenbaum. Church: a language for generative models. In David A. McAllester and Petri Myllymäki, editors, *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008*, pages 220–229, 2008. 63

Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. A convenient category for higher-order probability theory. In *LICS*, pages 1–12. IEEE Computer Society, 2017. 63, 104

Martin Hyland. A syntactic characterization of the equality in some models for the lambda calculus. *J. London Math. Soc*, 12:361–370, 1976. 76

Claire Jones and Gordon Plotkin. A probabilistic powerdomains of evaluation. In *Proceedings of the 4th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 1989. 62

Achim Jung and Regina Tix. The troublesome probabilistic powerdomain. *Electr. Notes Theor. Comput. Sci.*, 13:70–91, 1998. 63

Klaus Keimel and Gordon D. Plotkin. Mixed powerdomains for probability and nondeterminism. *Logical Methods in Computer Science*, 13(1), 2017. 63

Dexter Kozen. Semantics of probabilistic programs. In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*, pages 101–114. IEEE Computer Society, 1979. 62, 63

Jim Laird, Giulio Manzonetto, Guy McCusker, and Michele Pagani. Weighted relational models of typed lambda-calculi. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*. IEEE Computer Society, June 2013. 100

Brockway McMillan. Absolutely Monotone Functions. *Annals of Mathematics*, 60(3):467–501, 1954. 98

Paul-André Melliès. Categorical semantics of linear logic. *Panoramas et Synthèses*, 27, 2009. 66, 72

Eugenio Moggi. Computational lambda-calculus and monads. In *LICS*, pages 14–23. IEEE Computer Society, 1989. 62

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. 63, 106

Sungwoo Park, Frank Pfenning, and Sebastian Thrun. A probabilistic language based on sampling functions. *ACM Trans. Program. Lang. Syst.*, 31(1):4:1–4:46, December 2008. 63, 95

Andrew M. Pitts. Computational adequacy via 'mixed' inductive definitions. In Stephen Brookes, Michael Main, Austin Melton, Michael Mislove, and David Schmidt, editors, *Mathematical Foundations of Programming Semantics*, pages 72–82, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. ISBN 978-3-540-48419-6. 61, 77, 91

Gordon Plotkin. A powerdomain construction. *SIAM Journal of Computing*, 5(3):452–487, 1976. 62

Gordon Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–256, 1977. 76, 83, 95, 100

Mathys Rennela. Convexity and order in probabilistic call-by-name FPC. *CoRR*, abs/1607.04332, 2016. URL http://arxiv.org/abs/1607.04332. 63

John C. Reynolds. On the relation between direct and continuation semantics. *ICALP, Lecture Notes in Comput. Sci.*, 14:141–156, 1974. 76

Adam Scibior, Ohad Kammar, Matthijs Vákár, Sam Staton, Hongseok Yang, Yufei Cai, Klaus Ostermann, Sean K. Moss, Chris Heunen, and Zoubin Ghahramani. Denotational validation of higher-order bayesian inference. *PACMPL*, 2(POPL):60:1–60:29, 2018. 104

Peter Selinger. Towards a semantics for higher-order quantum computation. In *Proceedings of the 2nd International Workshop on Quantum Programming Languages, Turku, Finland*, number 33 in TUCS General Publication. Turku Centre for Computer Science, 2004. 97

Sam Staton, Hongseok Yang, Frank D. Wood, Chris Heunen, and Ohad Kammar. Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints. In *LICS*, pages 525–534. ACM, 2016. 63, 95

Alfred Tarski. A lattice theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5:285–309, 1955. 70, 77

Jean Vuillemin. Exact real computer arithmetic with continued fractions. In *Proceedings of the 1988 ACM Conference on LISP and Functional Programming*, LFP '88, pages 14–27, New York, NY, USA, 1988. ACM. ISBN 0-89791-273-X. 94

Frank D. Wood, Jan-Willem van de Meent, and Vikash Mansinghka. A new approach to probabilistic programming inference. In *AISTATS*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 1024–1032. JMLR.org, 2014. 63

List of Figures