Curry and Howard Meet Borel

Melissa Antonelli, Ugo Dal Lago, Paolo Pistone

Curry and Howard Meet Borel

Melissa Antonelli
Università di Bologna, Italy
melissa.antonelli2@unibo.it

Ugo Dal Lago
Università di Bologna, Italy
ugo.dallago@unibo.it

Paolo Pistone
Università di Bologna, Italy
paolo.pistone2@unibo.it

ABSTRACT

We show that an intuitionistic version of counting propositional logic corresponds, in the sense of Curry and Howard, to an expressive type system for the probabilistic event λ-calculus, a vehicle calculus in which both call-by-name and call-by-value evaluation of discrete randomized functional programs can be simulated. In this context, proofs (respectively, types) do not guarantee that validity (respectively, termination) holds, but *reveal* the underlying probability. We finally show how to obtain a system precisely capturing the probabilistic behavior of λ-terms, by endowing the type system with an intersection operator.

CCS CONCEPTS

• **Theory of computation** → **Type theory**; **Proof theory**; **Probabilistic computation**.

KEYWORDS

Curry-Howard Correspondence, Probabilistic Computation, Counting Logic, Termination, Intersection Types

ACM Reference Format:
Melissa Antonelli, Ugo Dal Lago, and Paolo Pistone. 2022. Curry and Howard Meet Borel. In *37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (LICS '22), August 2–5, 2022, Haifa, Israel.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3531130.3533361

1 INTRODUCTION

Among the many ways in which mathematical logic influenced programming language theory, the so-called Curry-Howard correspondence is certainly one of the most intriguing and meaningful ones. Traditionally, the correspondence identified by Curry [15] and formalized by Howard [35] (CHC in the following) relates propositional intuitionistic logic and the simply-typed λ-calculus. As is well-known, though, this correspondence holds in other contexts, too. Indeed, in the last fifty years more sophisticated type systems have been put in relation with expressive logical formalisms: from polymorphism [27, 29] to various forms of session typing [13, 55], from control operators [48] to dependent types [41, 53].

Nevertheless, there is a class of programming languages and type systems for which a correspondence in the style of CHC has not yet been found. We are talking about languages with probabilistic effects, for which type-theoretic accounts have recently been put forward in various ways, e.g. type systems based on sized types [17], intersection types [11] or type systems in the style of so-called amortized analysis [59]. In all the aforementioned cases, a type system was built by modifying deterministic, standard ones *without* being guided by logic, and instead incepting inherently quantitative concepts from probability theory into the system. So, one could naturally wonder if there is any logical system behind all this, and what kind of logic could possibly play the role of propositional logic in suggesting meaningful and expressive type systems for a λ-calculus endowed with probabilistic choice effects.

A tempting answer is to start from modal logic, which is known to correspond, in the Curry-Howard sense, to staged computation and algebraic effects [14, 20, 63]. Nevertheless, there is one aspect of randomized computation that modal logic fails to capture,[1] that is the *probability* of certain events, typically termination. In many of the probabilistic type systems mentioned above, for example, a term t receives a type that captures the fact that t has a certain probability q, perhaps strictly smaller than 1, of reducing to a value. This probability is an essential part of what we want to observe about the dynamics of t and, as such, has to be captured by its type, at least if one wants the type system to be expressive. Moreover, several other properties, as reachability and safety, can be reduced to termination (see, e.g., the discussion in [38]).

Recently, the authors have proposed to use *counting quantifiers* [2, 3] as a means to express probabilities within a logical language. These quantifiers, unlike standard ones, determine not only *the existence* of an assignment of values to variables with certain characteristics, but rather count *how many* of those assignments exist. Recent works show that classical propositional logic enriched with counting quantifiers corresponds to Wagner's hierarchy of counting complexity classes [58] (itself intimately linked to probabilistic complexity), and that Peano Arithmetics (PA, for short), enriched with analogous *measure* quantifiers, yields a system capable of speaking of randomized computation in the same sense in which standard PA models deterministic computation [3]. One may now wonder whether all this could scale to something like a proper CHC.

The aim of this work is precisely to give a positive answer to the aforementioned question, at the same time highlighting a few remarkable consequences of our study of probabilistic computation through the lens of logic. More specifically, the contributions of this paper are threefold:

- First of all, we introduce an intuitionistic version of Antonelli et al.'s counting propositional logic [2], together with a Kripke-style semantics based on the Borel σ-algebra of the Cantor space and a sound and complete proof system. Then, we identify a "computational fragment" of this logic for which we design a natural deduction system in which proof normalization simulates well-known evaluation strategies for probabilistic programs. This is in Section 3.

[1] With a few notable exceptions, e.g. [25].

LICS '22, August 2–5, 2022, Haifa, Israel

ACM ISBN 978-1-4503-9351-5/22/08.
https://doi.org/10.1145/3531130.3533361

- We then show that proofs can be decorated with terms of the *probabilistic event λ-calculus*, a calculus for randomized computation introduced by Dal Lago et al. in [18]. This gives rise to a type-system called $\mathbf{C}\lambda^{\{\}}_{\to}$. Remarkably, the correspondence scales to the underlying dynamics, i.e. proof normalization relates to reduction in the probabilistic event λ-calculus. This is in Section 4 and Section 5.
- We complete the picture by giving an intersection type assignment system, derived from $\mathbf{C}\lambda^{\{\}}_{\to}$, and proving that it precisely captures the normalization probability of terms of this λ-calculus. This is in Section 6.

Space limits prevent us from being comprehensive, but full details can be found in a long version of this paper [4].

2 FROM LOGIC TO COUNTING AND PROBABILITY: A ROADMAP

In this section we provide a first overview of our probabilistic CHC, at the same time sketching the route we will follow in the rest of the paper.

2.1 Randomized Programs and Counting Quantifiers

The first question we should ask ourselves concerns the kind of programs we are dealing with. What is a probabilistic functional program? Actually, it is a functional program with the additional ability of sampling from some distributions, or of proceeding by performing some form of discrete probabilistic choice.[2] This has a couple of crucial consequences: program evaluation becomes an essentially stochastic process, and programs may satisfy a given specification *up to* a certain probability. As an example, consider the λ-term,

$$\Xi_{\mathsf{half}} := \lambda x.\lambda y.x \oplus y$$

where $\oplus$ is a binary infix operator for fair probabilistic choice. When applied to two arguments t and u, the evaluation of Ξ_{half} results in either t, with probability one half, or u, again with probability one half.

But now, if we try to take Ξ_{half} as a proof of some logical formula, we see that standard propositional logic is simply not rich enough to capture the behavior above. Indeed, given that Ξ_{half} is a function of two arguments, it is natural to see it as a proof of an implication $A \to B \to C$, namely (following the BHK interpretation) as a function turning a proof of A and a proof of B into a proof of C. What is C, then? Should it be A or should it be B? Actually, it could be both, with some degree of uncertainty, but propositional logic is not able to express all this. At this point, our recent work on *counting quantifiers* in propositional logic [2] comes to the rescue.

Another way to look at discrete probabilistic programs is as programs which are allowed to sample an element ω from the *Cantor space* $2^{\mathbb{N}}$. For example, a probabilistic Turing machine M can be described as a 2-tape machine where the second tape is read-only and sampled from the Cantor space at each run. Similarly, the execution of Ξ_{half} applied to programs t and u can be described as the result of sampling ω and returning either t or u depending on some read value, e.g. $\omega(0)$. Crucially, for each input x and possible output y of a probabilistic program f, the set $S_{x,y}$ of elements in $2^{\mathbb{N}}$, making $f(x)$ produce y, is a *measurable* set (indeed, a Borel set), so it makes sense to say that the probability that $f(x)$ yields y coincides with the *measure* of $S_{x,y}$.

Yet, what has all this to do with logic and counting? The fundamental observation is that also the formulas of classical propositional logic provide ways of denoting Borel sets. Let $x_0, x_1, \ldots$ indicate a countable set of propositional variables (which can be seen as i.i.d. and uniformly distributed random variables $x_i \in \{0,1\}$); each variable x_i can be associated with the *cylinder set* (indeed, a Borel set of measure $\frac{1}{2}$) formed by all $\omega \in 2^{\mathbb{N}}$ such that $\omega(i) = 1$ (i.e., in logical terms, such that $\omega \vDash x_i$). Then, any propositional formula $b, c, \ldots$ formed built using the connectives $\neg, \wedge, \vee$ can be associated with the Borel sets $\{\omega \in 2^{\mathbb{N}} \mid \omega \vDash b\}$ which can be constructed starting from cylinder sets by means of complementation together with finite intersection and union. In this way, for any Boolean formula b, it makes sense to define *new* formulas like $\mathbf{C}^{\frac{1}{2}} b$, which is true when the Borel set associated with b has measure greater than $\frac{1}{2}$ (i.e. when b is satisfied by *at least* $\frac{1}{2}$ of its models). For instance, given two *distinct* indexes $i, j \in \mathbb{N}$, so that the variables x_i, x_j correspond to two independent events, the formula $\mathbf{C}^{\frac{1}{2}}(x_i \vee x_j)$ is true. Indeed, the Borel set formed by the $\omega \in 2^{\mathbb{N}}$ such that either $\omega(i) = 1$ or $\omega(j) = 1$ has measure greater than $\frac{1}{2}$ (equivalently, $x_i \vee x_j$ is satisfied by at least $\frac{1}{2}$ of its models). Instead, $\mathbf{C}^{\frac{1}{2}}(x_i \wedge x_j)$ is false, since, x_i and x_j being independent, the Borel set associated with $x_i \wedge x_j$ has only measure $\frac{1}{4}$.

Counting propositional logic (from now on, CPL) is defined by enriching classical propositional logic with counting quantifiers $\mathbf{C}^q$, where $q \in [0,1] \cap \mathbb{Q}$. Following our sketch, CPL admits a natural semantics in the Borel σ-algebra of the Cantor space, together with a sound and complete sequent calculus [2]. Notice that measuring a Boolean formula actually amounts at *counting* its models, that is, to a purely recursive operation, albeit one which needs not be doable in polynomial time (indeed, CPL is deeply related to Wagner's counting hierarchy [2]).

Going back to the term Ξ_{half}, what seems to be lacking in intuitionistic logic is precisely a way to express that C could be $\mathbf{C}^{\frac{1}{2}}A$, i.e. that it could be A *with probability at least* $\frac{1}{2}$, and, similarly, that it could be by $\mathbf{C}^{\frac{1}{2}}B$. In Section 3 we introduce an intuitionistic version of CPL, called iCPL, which enriches intuitionistic logic with Boolean variables as well as the counting quantifier $\mathbf{C}^q$. Intuitively, if a proof of a formula A can be seen as a deterministic program satisfying the specification A, a proof of $\mathbf{C}^q A$ will correspond to a probabilistic program that satisfies the specification A with probability q. Our main result consists in showing that proofs in iCPL correspond to functional probabilistic programs and, most importantly, that normalization in this logic describes probabilistic evaluation.

2.2 Can CbN and CbV Evaluation Coexist?

When dealing with λ-calculi extended with probabilistic choice, two different evaluation strategies are usually considered: the *call-by-name* (CbN) strategy, which possibly duplicates choices before evaluating them, and the *call-by-value* (CbV) strategy, which instead

[2] Here we are not at all concerned with sampling from continuous distributions, nor with capturing any form of conditioning.

evaluates choices before possibly duplicating their outcomes. Importantly, the probability of termination of a program might differ depending on the chosen strategy. For example, consider the application of the term $\mathbf{2} := \lambda yx.y(yx)$ (i.e. the second Church numeral) to $I \oplus \Omega$, where $I = \lambda x.x$ and Ω is the diverging term $(\lambda x.xx)\lambda x.xx$. Under CbN, the redex $\mathbf{2}(I \oplus \Omega)$ first produces $\lambda x.(I \oplus \Omega)((I \oplus \Omega)x)$, then reduces to any of the terms $\lambda x.u(vx)$, with u, v chosen from $\{I, \Omega\}$, each with probability $\frac{1}{4}$. Since $\lambda x.u(vx)$ converges only when $u = v = I$, the probability of convergence is thus $\frac{1}{4}$. Under CbV, in $\mathbf{2}(I \oplus \Omega)$ one first has to evaluate $I \oplus \Omega$, and then passes the result to $\mathbf{2}$, hence returning either the converging term $I(Ix)$ or the diverging term $\Omega(\Omega x)$, each with probability $\frac{1}{2}$.

If we now try to think of the Church numeral $\mathbf{2}$ as a proof of some counting quantified formula, we see that, depending on the reduction strategy considered, it must prove a *different* formula. Indeed, given that $I \oplus \Omega$ proves $\mathbf{C}^{\frac{1}{2}}(A \to A)$, in the CbN case, $\mathbf{2}$ proves $\mathbf{C}^{\frac{1}{2}}(A \to A) \to A \to \mathbf{C}^{\frac{1}{4}}A$, since only in one out of four cases it yields a proof of A, while in the CbV case, $\mathbf{2}$ proves the formula $\mathbf{C}^{\frac{1}{2}}(A \to A) \to A \to \mathbf{C}^{\frac{1}{2}}A$, as it yields a proof of A in one case out of two.

In the literature on probabilistic λ-calculi, the apparent incompatibility of CbN and CbV evaluation is usually resolved by restricting to calculi with one or the other strategy. Nevertheless, the observation above suggests that, if functional programs are typed using counting quantifiers, it should become possible to make the two evaluation strategies coexist, by assigning them different types.

Actually, a few recent approaches [16, 18, 21, 23] already suggest ways to make CbN and CbV evaluation live together. In particular, in the *probabilistic event λ-calculus* [18] the choice operator $\oplus$ is decomposed into two different operators, yielding a confluent calculus: a *choice operator* $t \oplus_a u$, depending on some probabilistic event $a \in \{0, 1\}$, and a *probabilistic event generator* $va.t$, which actually "flips the coin". In this language, the CbN and CbV applications of $\mathbf{2}$ to $I \oplus \Omega$ are encoded by two *distinct* terms $\mathbf{2}(va.I \oplus_a \Omega)$ and $va.\mathbf{2}(I \oplus_a \Omega)$, crucially distinguishing between generating a probabilistic choice *before* or *after* a duplication takes place.

This calculus constitutes then an ideal candidate for our CHC. Indeed, as we shall see, the logical rules for the counting quantifier $\mathbf{C}^q$ naturally give rise to typing rules for the event generator va. In Section 4 we introduce a variant $\Lambda^{\{\}}_{\mathsf{PE}}$ of the calculus from [18], with the underlying probability space $\{0, 1\}$ replaced by the Cantor space, and in Section 5 we introduce a type system $\mathbf{C}\lambda^{\{\}}_{\to}$ for simple types with counting quantifiers, showing that natural deduction derivations translate into typing derivations in $\mathbf{C}\lambda^{\{\}}_{\to}$, with normalization precisely corresponding to reduction in $\Lambda^{\{\}}_{\mathsf{PE}}$.

2.3 Capturing Probability of Normalization via Types

As observed in the Introduction, a quantitative property we would like to observe using types is the probability of termination. Nevertheless, given that the reduction of $\Lambda^{\{\}}_{\mathsf{PE}}$ is purely deterministic, what notions of probabilistic termination should we actually observe?

Rather than evaluating programs by implementing probabilistic choices, reduction in the probabilistic event λ-calculus progressively generates the *full tree* of outcomes of (sequences of) probabilistic choices, giving rise to a *distribution* of values. Therefore, given a term t, rather than asking whether *some* or *all* reductions of t terminate, it makes sense to ask *what is the probability* for a normal form to be found by generating all probabilistic outcomes of t.

In Section 6 we will first show that when the type $\mathbf{C}^q\sigma$ is assigned to a program t, the value $q \in \mathbb{Q} \cap (0, 1]$ provides a lower bound for the actual probability of finding a (head) normal form in the development of t. Then we show that, in analogy with what happens in the deterministic case, by extending the type system with an *intersection* operator, one can attain an upper bound and, thus, fully characterize the distribution of values associated with a term.

2.4 Preliminaries on the Cantor Space

Throughout the paper, we exploit some basic facts about the Cantor space, its Borel σ-algebra, and their connections with Boolean logic, that we briefly recall here.

We consider a countably infinite set $\mathcal{A}$ of *names*, noted $a, b, c, \ldots$. For any finite subset $X \subseteq \mathcal{A}$, we let $\mathbf{B}_X$ (resp. $\mathbf{B}_{\mathcal{A}}$) indicate the *Borel σ-algebra* on the X-th product of the Cantor space $(2^{\mathbb{N}})^X$ (resp. on the $\mathcal{A}$-th product $(2^{\mathbb{N}})^{\mathcal{A}}$), that is, the smallest σ-algebra containing all open sets under the product topology. There exists a unique measure μ on $\mathbf{B}_{\mathcal{A}}$ such that $\mu(C_{a,i}) = \frac{1}{2}$ for all *cylinders* $C_{a,i} = \{\omega \mid \omega(a)(i) = 1\}$. The measure μ restricts to a measure μ_X on $\mathbf{B}_X$ by letting $\mu_X(S) = \mu(S \times (2^{\mathbb{N}})^{\mathcal{A}-X})$.

Boolean formulas with names in $\mathcal{A}$ are defined by:

$$\mathcal{b} ::= \top \mid \bot \mid x^i_a \mid \neg\mathcal{b} \mid \mathcal{b} \wedge \mathcal{b} \mid \mathcal{b} \vee \mathcal{b}$$

where $a \in \mathcal{A}$ and $i \in \mathbb{N}$. We let $\mathsf{FN}(\mathcal{b}) \subseteq \mathcal{A}$ be the set of names occurring in $\mathcal{b}$. For all Boolean formulas $\mathcal{b}$ and $X \supseteq \mathsf{FN}(\mathcal{b})$, we let $[\![\mathcal{b}]\!]_X$ indicate the Borel set $\{\omega \in (2^{\mathbb{N}})^X \mid \omega \vDash \mathcal{b}\}$ (notice that $[\![x^i_a]\!]_{\{a\}} = C_{a,i}$). As the value $\mu_X([\![\mathcal{b}]\!]_X) \in [0, 1] \cap \mathbb{Q}$ does not depend on the choice of $X \supseteq \mathsf{FN}(\mathcal{b})$, we will note it simply as $\mu(\mathcal{b})$.

3 INTUITIONISTIC COUNTING PROPOSITIONAL LOGIC

In this section we introduce a constructive version of CPL, that we call iCPL, which extends standard intuitionistic logic with Boolean variables and counting quantifiers. The logic iCPL combines constructive reasoning (corresponding, under the standard CHC, to functional programming) with semantic reasoning on Boolean formulas and their models (corresponding, as we have seen, to discrete probabilistic reasoning). Formulas of iCPL are somehow *hybrid*, as comprising both a countable set $\mathcal{P} = \{p, q, \ldots\}$ of intuitionistic propositional variables, and named Boolean propositional variables x^i_a.

For example, consider the formula below:

$$A := p \to q \to (x^0_a \wedge p) \vee (\neg x^0_a \wedge q).$$

Intuitively, proving A amounts to showing that, whenever p and q hold, given that either x^0_a or $\neg x^0_a$ do, in the first case p holds, and in the second q does. Suppose we test A against some element ω from the Cantor space. A way of proving A, in the "environment" ω, could then be as follows: given assumptions p and q, conclude p and x^0_a if $\omega(0) = 1$, i.e. if ω satisfies x^0_a, and conclude q and $\neg x^0_a$ if $\omega(0) = 0$, i.e. if ω does not satisfy $\neg x^i_a$. In other words, a proof of A under ω could be something like $\lambda xy.x \oplus_a y$. Now, assuming

that ω is uniformly sampled, what are the chances that our strategy will actually yield a proof of A? Well, there are two possible cases, and in both cases we get a proof of one of the disjuncts $x_a^0 \wedge p$ and $\neg x_a^0 \wedge q$, and thus a proof of A. Hence, we obtain a proof of A with probability 1 or, otherwise said, a proof of $\mathbf{C}_a^1 A$ (this time independent from any "environment"). As a term, this proof looks then precisely as the closed term $va.\lambda xy.x \oplus_a y$.

Consider now another formula:

$$B := p \to q \to x_a^0 \wedge p.$$

Given an environment ω, if $\omega(0) = 1$, a proof of B can conclude both p and x_a^0 from the assumptions p and q; instead, if $\omega(0) = 0$, there is nothing to be proved, since the conclusion $x_a^0 \wedge p$ can in no way be true under ω. Depending on the environment ω, we can thus either construct a correct proof of B or acknowledge that B cannot be true. If ω is sampled uniformly, we obtain thus a proof of B only in one half of the cases, i.e. we get a proof of $\mathbf{C}_a^{\frac{1}{2}} B$, and this proof looks precisely as the term $va.\lambda xy.x \oplus_a ?$, where ? can be any term.

3.1 The Semantics and Proof-Theory of iCPL.

The formulas of the logic just sketched are defined by:

$$A ::= \top \mid \bot \mid x_a^i \mid p \mid A \wedge A \mid A \vee A \mid A \to A \mid \mathbf{C}_a^q A,$$

where $p \in \mathcal{P}$ and $q \in (0,1] \cap \mathbb{Q}$.[3] A natural semantics for iCPL-formulas is given in terms of Kripke-like structures.

Definition 3.1. An iCPL-*structure* is a triple $\mathcal{M} = (W, \preceq, \mathbf{i})$ where W is a countable set, $\preceq$ is a preorder on W, and $\mathbf{i} : \mathcal{P} \to W^{\uparrow}$, where $W^{\uparrow}$ is the set of upper subsets of W.

The interpretation of formulas in iCPL-structures combines a set W of worlds (for the interpretation of intuitionistic propositional variables) with the choice of an element of the Cantor space (for the interpretation of Boolean variables): for any iCPL-structure $\mathcal{M} = (W, \preceq, \mathbf{i})$ and finite set X, we define the relation $w, \omega \Vdash^X_{\mathcal{M}} A$ (where $w \in W$, $\omega \in (2^{\mathbb{N}})^X$ and $\mathsf{FN}(A) \subseteq X$) by induction as follows:

- $w, \omega \nVdash^X_{\mathcal{M}} \bot$ and $w, \omega \Vdash^X_{\mathcal{M}} \top$;
- $w, \omega \Vdash^X_{\mathcal{M}} x_a^i$ iff $\omega(a)(i) = 1$;
- $w, \omega \Vdash^X_{\mathcal{M}} p$ iff $w \in \mathbf{i}(p)$;
- $w, \omega \Vdash^X_{\mathcal{M}} A \wedge B$ iff $w, \omega \Vdash^X_{\mathcal{M}} A$ and $w, \omega \Vdash^X_{\mathcal{M}} B$;
- $w, \omega \Vdash^X_{\mathcal{M}} A \vee B$ iff $w, \omega \Vdash^X_{\mathcal{M}} A$ or $w, \omega \Vdash^X_{\mathcal{M}} B$;
- $w, \omega \Vdash^X_{\mathcal{M}} A \to B$ iff for all $w' \succeq w$, $w', \omega \Vdash^X_{\mathcal{M}} A$ implies $w', \omega \Vdash^X_{\mathcal{M}} B$;
- $w, \omega \Vdash^X_{\mathcal{M}} \mathbf{C}_a^q A$ iff

$$\mu\left(\left\{\omega' \in 2^{\mathbb{N}} \mid w, \omega +_a \omega' \Vdash^{X \cup \{a\}}_{\mathcal{M}} A\right\}\right) \geq q$$

where $\omega +_a \omega' \in (2^{\mathbb{N}})^{X \cup \{a\}}$ is defined by $(\omega +_a \omega')(b)(n) = \omega(b)(n)$, for $b \in X$, and $(\omega +_a \omega')(a)(n) = \omega'(n)$.

Using properties of the Borel σ-algebra $\mathbf{B}_X$, it can be shown that for any $w \in W$ and formula A, the set $\{\omega \in (2^{\mathbb{N}})^X \mid w, \omega \Vdash^X_{\mathcal{M}} A\}$ is a Borel set, and thus measurable. We write $\Gamma \Vdash^X_{\mathcal{M}} A$ when for all $w \in W$ and $\omega \in (2^{\mathbb{N}})^X$, whenever $w, \omega \Vdash^X_{\mathcal{M}} \Gamma$ holds, also $w, \omega \Vdash^X_{\mathcal{M}} A$ holds. We write $\Gamma \vDash A$ when for any iCPL-structure $\mathcal{M}$ and $X \supseteq \mathsf{FN}(A)$, $\Gamma \Vdash^X_{\mathcal{M}} A$ holds.

A sound and complete proof system for iCPL can be defined (full details can be found in [4]). Indeed, starting from usual natural deduction for intuitionistic logic, one obtains a calculus $\mathbf{ND}_{\mathsf{iCPL}}$ for iCPL by adding the excluded middle $x_a^i \vee \neg x_a^i$ as an axiom for Boolean variables, together with suitable rules and axioms for counting quantifiers.

Theorem 3.2. $\Gamma \vDash A$ *iff* $\Gamma \vdash_{\mathbf{ND}_{\mathsf{iCPL}}} A$.

For example, the counting quantifier $\mathbf{C}_a^q$ admits an introduction rule as below

$$\frac{\Gamma, d \vdash A \qquad \mu(d) \geq q}{\Gamma \vdash \mathbf{C}_a^q A} \tag{CI}$$

with the proviso that a does not occur in Γ, and is the only name occurring in d. Intuitively, this rule says that if we can prove A under assumptions Γ and d, and if a randomly chosen valuation of a has chance q of being a model of d, then we can build a proof of $\mathbf{C}_a^q A$ from Γ. Observe that the rule (CI) has a semantic premise $\mu(d) \geq q$: we ask to an oracle to count the models of d for us (this is similar to what happens in sequent calculi for CPL, see [2]).

3.2 The Computational Fragment of iCPL.

From the perspective of the CHC, however, iCPL is not what we are looking for, due to the presence of classical Boolean formulas. In order to relate proofs and programs, one should first choose among the several existing constructive interpretations of classical logic. Yet, in our previous examples Boolean formulas were not treated as formulas to be proved but, rather, as *semantic constraints* that programs may or may not satisfy (for example, when saying that a program, depending on some event ω, yields a proof of A *when* ω *satisfies* b, or that a program has q chances of yielding a proof of A *when* b *has measure at least* q).

Would it be possible, then, to somehow *separate* purely constructive reasoning from Boolean semantic reasoning within formulas and proofs of iCPL? The following lemma suggests that this is indeed possible.

Lemma 3.3 (Decomposition Lemma). *For any formula A of* iCPL *there exist Boolean formulas b_v and purely intuitionistic formulas A_v (i.e. formulas of* iCPL *containing no Boolean variables), where v varies over all possible valuations of the Boolean variables in A, such that:*

$$\vDash A \leftrightarrow \bigvee_v b_v \wedge A_v$$

Proof sketch. The idea is to let b_v be the formula characterizing v, i.e. the conjunction of all variables true in v and of all negations of variables false in v, and let A_v be obtained from A by replacing each Boolean variable by either $\top$ or $\bot$, depending on its value under v. □

By Lemma 3.3, any sequent $\Gamma \vdash A$ of iCPL can be associated with a family of intuitionistic sequents of the form $\Gamma_v, b_v \vdash A_v$, where v ranges over all valuations of the Boolean variables of Γ and A, and Γ_v, b_v, A_v are as in Lemma 3.3. We will note such special sequents as $\Gamma_v \vdash b_v \rightarrowtail A_v$, in order to highlight the special role

[3] With respect to CPL, we do not consider quantified formulas of the form $\mathbf{C}_a^0 A$, as these are equivalent to $\top$.

played by the Boolean formula $\mathscr{b}_v$, which intuitively describes the "environment" in which the sequent is considered.

The sequents $\Gamma \vdash \mathscr{b} \rightarrowtail A$ have a natural computational interpretation: they express program specifications of the form "Π yields a proof of A from Γ whenever its sampled function satisfies $\mathscr{b}$". By the way, Lemma 3.3, ensures that, *modulo* Boolean reasoning, logical arguments in iCPL can be reduced to (families of) arguments of this kind.

Let then iCPL_0 be the fragment formed by the purely intuitionistic formulas of iCPL, which are defined by:

$$A ::= p \mid A \to A \mid \mathbf{C}^q A$$

where $q \in (0,1] \cap \mathbb{Q}$. For simplicity, and since this is enough for the CHC, we take here implication as the only connective, yet all other propositional connectives could be added. As formulas do not contain Boolean variables, counting quantifiers in iCPL_0 are not named. We use $\mathbf{C}^{q_1 * \cdots * q_n} A$ (or even $\mathbf{C}^{\vec{q}}$, for simplicity) as an abbreviation for $\mathbf{C}^{q_1} \dots \mathbf{C}^{q_n} A$.

The natural deduction system $\mathbf{ND}_{\mathsf{iCPL}_0}$ for iCPL_0 is defined by the rules illustrated in Fig. 1. In the rule (CI) it is assumed that $\mathsf{FN}(\mathscr{b}) \cap \mathsf{FN}(\mathscr{d}) = \emptyset$ and that $\mathscr{d}$ contains at most one name. A few rules involve *semantic premises* of the forms $\mathscr{b} \vDash \mathscr{c}$ and $\mu(\mathscr{b}) \geq q$. Beyond standard intuitionistic rules, $\mathbf{ND}_{\mathsf{iCPL}_0}$ comprises structural rules to manipulate Boolean formulas, and introduction and elimination rules for the counting quantifier. The rule ($\bot$) yields *dummy* proofs of any formula, i.e. proofs which are correct for *no* possible event; the rule (m) combines two proofs Π_1, Π_2 of the same formula into one single proof Π', with the choice depending on the value of some Boolean variable x^i_a (Π' is thus something like $\Pi_1 \oplus_a \Pi_2$). The introduction rule for $\mathbf{C}^q$ is similar to rule (CI). It is explained as follows: if Π, in the "environment" $\omega +_a \omega' \in (2^{\mathbb{N}})^{X \cup \{a\}} \simeq (2^{\mathbb{N}})^X \times 2^{\mathbb{N}}$, yields a proof of A whenever $\omega +_a \omega'$ satisfies the two *independent* constraints $\mathscr{b}$ and $\mathscr{d}$ (i.e. $\omega \vDash \mathscr{b}$ and $\omega' \vDash \mathscr{d}$), then by randomly choosing $\omega' \in 2^{\mathbb{N}}$, we have at least $q \geq \mu(\mathscr{d})$ chances of getting a proof of A (Π' is thus something like $va.\Pi$). Finally, the elimination rule (CE) essentially turns a proof of $A \to B$ into a proof of $\mathbf{C}^q A \to \mathbf{C}^{qs} B$. As we will see, this rule captures CbV function application.

Example 3.4. Fig. 2a illustrates a proof $\Pi_{\frac{1}{2}\mathsf{id}}$ of $\mathbf{C}^{\frac{1}{2}}(A \to A)$ obtained by first "mixing" a correct proof of $A \to A$ with a dummy one, and then introducing the counting quantifier. Fig. 2b illustrates a derivation of $\mathbf{C}^q(A \to A) \to A \to \mathbf{C}^{q \star q} A$.

As shown in detail in [4], natural deduction proofs for iCPL can be related with *families* of natural deduction proofs in iCPL_0, along the lines of Lemma 3.3.

3.3 Normalization in iCPL_0

From the CHC perspective, natural deduction proofs correspond to programs, and normalization corresponds to execution. Let us look at normalization in iCPL_0, then.

The two main normalization steps are ($\to$I/$\to$E) and (CI/CE). The cuts ($\to$I/$\to$E) are eliminated, as usual, by means of the admissible substitution rule

$$\frac{\Gamma \vdash \mathscr{b} \rightarrowtail A \qquad \Gamma, A \vdash \mathscr{b} \rightarrowtail B}{\Gamma \vdash \mathscr{b} \rightarrowtail B} \text{ (subst)}$$

Identity Rule

$$\frac{}{\Gamma, A \vdash \mathscr{b} \rightarrowtail A} \text{ (id)}$$

Structural Rules

$$\frac{\mathscr{b} \vDash \bot}{\Gamma \vdash \mathscr{b} \rightarrowtail A} (\bot)$$

$$\frac{\Gamma \vdash \mathscr{c} \rightarrowtail A \qquad \Gamma \vdash \mathscr{d} \rightarrowtail A \qquad \mathscr{b} \vDash H(\mathscr{c}, \mathscr{d}, x^i_a)}{\Gamma \vdash \mathscr{b} \rightarrowtail A} \text{ (m)}$$

$$(H(\mathscr{c}, \mathscr{d}, x^i_a) = (\mathscr{c} \wedge x^i_a) \vee (\mathscr{d} \wedge \neg x^i_a))$$

Logical Rules

$$\frac{\Gamma, A \vdash \mathscr{b} \rightarrowtail B}{\Gamma \vdash \mathscr{b} \rightarrowtail (A \to B)} (\to\text{I})$$

$$\frac{\Gamma \vdash \mathscr{b} \rightarrowtail (A \to B) \qquad \Gamma \vdash \mathscr{b} \rightarrowtail A}{\Gamma \vdash \mathscr{b} \rightarrowtail B} (\to\text{E})$$

Counting Rules

$$\frac{\Gamma \vdash \mathscr{b} \wedge \mathscr{d} \rightarrowtail A \qquad \mu(\mathscr{d}) \geq q}{\Gamma \vdash \mathscr{b} \rightarrowtail \mathbf{C}^q A} \text{ (CI)}$$

$$\frac{\Gamma \vdash \mathscr{b} \rightarrowtail \mathbf{C}^q A \qquad \Gamma, A \vdash \mathscr{b} \rightarrowtail B}{\Gamma \vdash \mathscr{b} \rightarrowtail \mathbf{C}^{qs} B} \text{ (CE)}$$

Figure 1: Rules of $\mathbf{ND}_{\mathsf{iCPL}_0}$.

The normalization step (CI/CE), illustrated in Fig. 3, applies the rule (subst) to the premiss of (CI) and the minor premiss of (CE), and permutes the rule (CI) downwards. The other normalization steps (fully illustrated in [4]) permute (m) with other rules.

For example, if we cut the proof Π from Fig. 2b with $\Pi_{\frac{1}{2}\mathsf{id}}$ from Fig. 2a (by letting $q = \frac{1}{2}$), we obtain, after normalization, a normal proof of $A \to \mathbf{C}^{\frac{1}{2} * \frac{1}{2}} A$. One can see that normalization duplicates Π, that is, it duplicates the choice between the correct and the dummy proof of $A \to A$, yielding a proof which is correct only in one case over four. To make the study of normalization as simple as possible, we did not consider a "multiplication rule" to pass from $\mathbf{C}^q \mathbf{C}^s A$ to $\mathbf{C}^{qs} A$ (i.e., in our example, from $\mathbf{C}^{\frac{1}{2} * \frac{1}{2}} A$ to $\mathbf{C}^{\frac{1}{4}} A$), as this would introduce other normalization steps.[4]

In Section 5 it will be shown (Prop. 5.4) that, once proofs in $\mathbf{ND}_{\mathsf{iCPL}_0}$ are seen as probabilistic programs, the two normalization steps ($\to$I/$\to$E) and (CI/CE) simulate CbN and CbV evaluation, respectively, and that all other permuting rules correspond to the *permuting reductions* for the probabilistic λ-calculus introduced in the next section. Moreover, as a by-product of the CHC developed in the following sections, we will obtain a strong normalization theorem for iCPL_0 (Corollary 5.5).

4 THE PROBABILISTIC EVENT LAMBDA CALCULUS

In this section we introduce the computational side of the CHC, that is, a variant of the probabilistic event λ-calculus Λ_{PE} from [18], with choices depending on events from the Cantor space. We will then

[4] In [4], an alternative "CbN" proof-system $\mathbf{ND}_{\mathsf{iCPL}_0}$ also comprising this rule is studied.

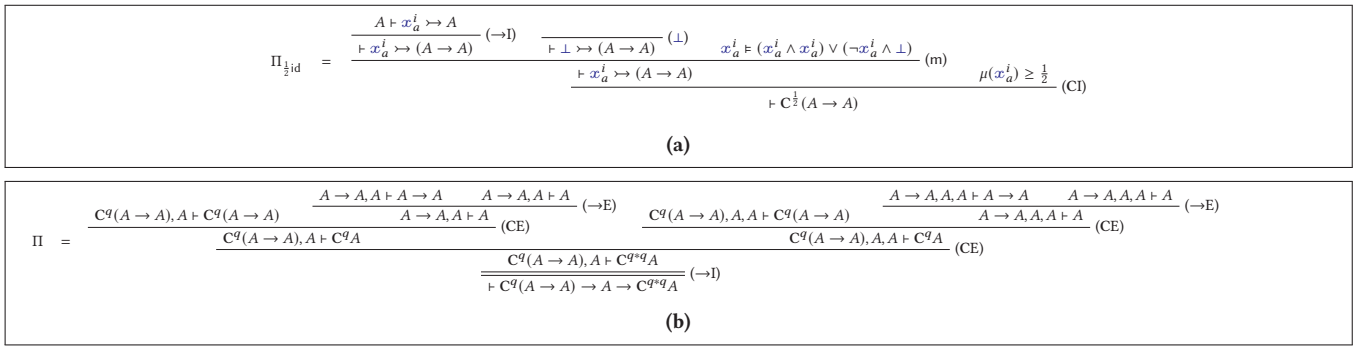

Figure 2: Examples of derivations in $\mathrm{ND}_{\mathsf{iCPL}_0}$.

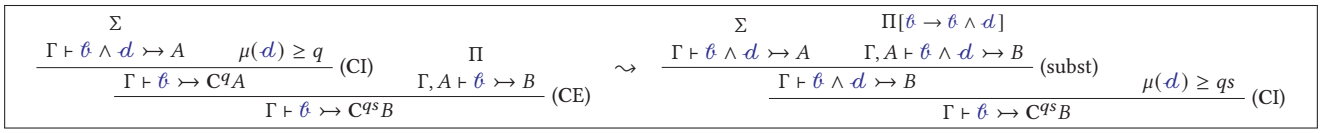

Figure 3: Normalization step (CI)/(CE) of $\mathrm{ND}_{\mathsf{iCPL}_0}$.

discuss how terms of Λ_{PE} yield distributions of values, and define two notions of probabilistic normalization for such distributions. We finally introduce a further variant $\Lambda_{\mathsf{PE}}^{\{\}}$ of Λ_{PE}, which provides a smoother representation of CbV functions.

4.1 A λ-Calculus Sampling from the Cantor Space

The terms of Λ_{PE} are defined by the grammar below:

$$t ::= x \mid \lambda x.t \mid tt \mid t \oplus_a^i t \mid va.t$$

with $a \in \mathcal{A}$, and $i \in \mathbb{N}$. The intuition is that va samples some function ω from the Cantor space, and $t \oplus_a^i u$ yields either t or u depending on the value $\omega(a)(i) \in \{0,1\}$. In the following we let $t \oplus^i u$ be an abbreviation for $va.t \oplus_a^i u$ (supposing a does not occur free in either t or u). For any term t, finite set X and $\omega \in (2^{\mathbb{N}})^X$, the "application of ω to t through X", noted $\pi_X^\omega(t)$, is defined by:

$$\begin{aligned}
\pi_X^\omega(x) &= x \\
\pi_X^\omega(\lambda x.t) &= \lambda x.\pi_X^\omega(t) \\
\pi_X^\omega(tu) &= \pi_X^\omega(t)\pi_X^\omega(u) \\
\pi_X^\omega(t \oplus_a^i u) &= \begin{cases} \pi_X^\omega(t) & \text{if } a \in X \text{ and } \omega(a)(i) = 1 \\ \pi_X^\omega(u) & \text{if } a \in X \text{ and } \omega(a)(i) = 0 \\ \pi_X^\omega(t) \oplus_a^i \pi_X^\omega(u) & \text{if } a \notin X \end{cases} \\
\pi_X^\omega(vb.t) &= vb.\pi_X^\omega(t)
\end{aligned}$$

In usual randomized λ-calculi, program execution is defined so to be inherently probabilistic: for example a term $t \oplus u$ can reduce to either t or u, with probability $\frac{1}{2}$. In this way, chains of reduction can be described as *stochastic Markovian sequences* [50], leading to formalize the idea of *normalization with probability* $r \in [0,1]$ (see [10]).

By contrast, reduction in Λ_{PE} is fully deterministic: beyond the usual (and un-resticted) β-rule $(\lambda x.t)u \to_\beta t[u/x]$, it comprises a

$$\begin{array}{rll}
t \oplus_a^i t \to_{\mathsf{p}} t & & \text{(i)} \\
(t \oplus_a^i u) \oplus_a^i v \to_{\mathsf{p}} t \oplus_a^i v & & (\mathsf{c}_1) \\
t \oplus_a^i (u \oplus_a^i v) \to_{\mathsf{p}} t \oplus_a^i v & & (\mathsf{c}_2) \\
\lambda x.(t \oplus_a^i u) \to_{\mathsf{p}} (\lambda x.t) \oplus_a^i (\lambda x.u) & & (\oplus\lambda) \\
(t \oplus_a^i u)v \to_{\mathsf{p}} (tu) \oplus_a^i (uv) & & (\oplus\mathsf{f}) \\
t(u \oplus_a^i v) \to_{\mathsf{p}} (tu) \oplus_a^i (tv) & & (\oplus\mathsf{a}) \\
(t \oplus_a^i u) \oplus_b^j v \to_{\mathsf{p}} (t \oplus_b^j v) \oplus_a^i (u \oplus_b^j v) & ((a,i) < (b,j)) & (\oplus\oplus_1) \\
t \oplus_b^j (u \oplus_a^i v) \to_{\mathsf{p}} (t \oplus_b^j u) \oplus_a^i (t \oplus_b^j v) & ((a,i) < (b,j)) & (\oplus\oplus_2) \\
vb.(t \oplus_a^i u) \to_{\mathsf{p}} (vb.t) \oplus_a^i (vb.u) & (a \neq b) & (\oplus\nu) \\
va.t \to_{\mathsf{p}} t & (a \notin \mathsf{FN}(t)) & (\neg\nu) \\
\lambda x.va.t \to_{\mathsf{p}} va.\lambda x.t & & (\nu\lambda) \\
(va.t)u \to_{\mathsf{p}} va.(tu) & & (\nu\mathsf{f})
\end{array}$$

Figure 4: Permutative reductions.

permutative reduction $t \to_{\mathsf{p}} u$ defined by the rules in Fig. 4 (where $(a,i) < (b,j)$ if either vb occurs in the scope of va, or $a = b$ and $i < j$). Intuitively, permutative reductions implement probabilistic choices by computing the full tree of possible choices. For example, given terms t_1, t_2, u_1, u_2, one can see that the term $va.(t_1 \oplus_a^0 t_2)(u_1 \oplus_a^1 u_2)$ reduces to $va.(t_1u_1 \oplus_a^1 t_1u_2) \oplus_a^0 (t_2u_1 \oplus_a^1 t_2u_2)$, hence displaying all possible alternatives.

The fundamental properties of Λ_{PE} are the following:

Theorem 4.1 ([18]). *$\to_{\mathsf{p}}$ is confluent and strongly normalizing. Full reduction $\to := \to_\beta \cup \to_{\mathsf{p}}$ is confluent.*

The existence and unicity of normal forms for $\to_{\mathsf{p}}$ (that we call *permutative normal forms*, PNFs for short) naturally raises the question of what these normal forms represent.

Let $\mathcal{T}$ indicate the set of PNFs containing no free name occurrence. A PNF $t \in \mathcal{T}$ can be of two forms: either t starts with a generator, i.e. $t = va.t'$, and t' is a tree of a-labeled choices $\oplus_a^i$ whose leaves form a finite set of $\mathcal{T}$ (the *support* of t', $\mathrm{supp}(t)$); otherwise, t is of the form $\lambda x_1.\dots.\lambda x_n.t' t_1 \dots t_p$, where t' is either a variable or a λ. We call these last terms *pseudo-values*, and we let $\mathcal{V} \subseteq \mathcal{T}$ indicate the set formed by them.

Using this decomposition, any $t \in \mathcal{T}$ can be associated in a unique way with a *(sub-)distribution* of pseudo-values $\mathcal{D}_t : \mathcal{V} \to [0,1]$ by letting $\mathcal{D}_t(v) = \delta_t$ (where $\delta_t(t) = 1$ and $\delta_t(v) = 0$ if $t \neq v$) when $t \in \mathcal{V}$, and

$$\mathcal{D}_t(v) = \sum_{u \in \mathrm{supp}(t')} \mathcal{D}_u(v) \cdot \mu\Big(\big\{\omega \in 2^{\mathbb{N}} \mid \pi^{\omega}_{\{a\}}(t') = u\big\}\Big)$$

if $t = va.t'$. Intuitively, $\mathcal{D}_t(v)$ measures the probability of finding v by iteratively sampling events from the Cantor space and applying them to t any time a generator v is found.

4.2 Probabilistic (Head) Normalization

Given a term $t \in \mathcal{T}$, the questions "is t in normal form?" and "does t reduce to a normal form?" have univocal yes/no answers, because $\to$ is deterministic. However, if we think of t rather as $\mathcal{D}_t$, the relevant questions become "with what probability *is* t in normal form?" and "with what probability does t *reduce* to normal form?". To answer this kind of questions we introduce functions $\mathrm{HNV}_{\to}(t)$, $\mathrm{NF}_{\to}(t)$ measuring the probability that t reduces to a normal form. Let us consider head-normal forms, first. A *head-reduction* $t \to_{\mathsf{h}} u$ is either a $\to_{\mathsf{p}}$-reduction or a $\to_\beta$-reduction of the form $\mathsf{R}[\lambda\vec{x}.(\lambda y.t)uu_1 \dots u_n] \to_\beta \mathsf{R}[\lambda\vec{x}.t[u/x]u_1 \dots u_n]$, where R is a *randomized context*, defined by the grammar

$$\mathsf{R}[\,] ::= [\,] \mid \mathsf{R}[\,] \oplus_a^i u \mid t \oplus_a^i \mathsf{R}[\,] \mid va.\mathsf{R}[\,].$$

A *head normal value* (in short, HNV) is a $\to_{\mathsf{h}}$-normal term which is also a pseudo-value, i.e. is of the form $\lambda\vec{x}.yu_1 \dots u_n$. We let HNV indicate the set of such terms.

Definition 4.2. For any $t \in \mathcal{T}$, $\mathrm{HNV}(t) := \sum_{v \in \mathrm{HNV}} \mathcal{D}_t(v)$ and $\mathrm{HNV}_{\to}(t) := \sup\{\mathrm{HNV}(u) \mid t \to_{\mathsf{h}}^* u\}$. When $\mathrm{HNV}_{\to}(t) \geq q$, we say that t *yields a HNV with probability at least* q.

For example, if $t = va.(\lambda x\lambda y.(y \oplus_a^i I)x)u$, where $u = I \oplus^j \Omega$, then $\mathrm{HNV}_{\to}(t) = \frac{3}{4}$. Indeed, we have

$$t \to_{\mathsf{h}}^* va.\lambda y.(y(vb.I \oplus_b^j \Omega)) \oplus_a^i (vb'.I \oplus_{b'}^j \Omega)$$

and three over the four possible choices (corresponding to choosing between either left or right for both va and vb') yield a HNV. Observe that the choice about vb does not matter, since $\lambda y.yu$ is already a HNV.

Let us now consider normal forms. The first idea might be to define a similar function $\mathrm{NF}(t) = \sum_{v \text{ normal form}} \mathcal{D}_t(v)$. Nevertheless, according to this definition, a term like $t = \lambda x.x(va.I \oplus_a^0 \Omega)$ would have probability 0 of yielding a normal form. Instead, our guiding intuition here is that t should yield a normal form with probability $\frac{1}{2}$, i.e. depending on a choice for a. This leads to the following definition:

Definition 4.3. For any $t \in \mathcal{T}$, $\mathrm{NF}(t)$ is defined by:

- if $t = \lambda\vec{x}.yu_1 \dots u_n \in \mathrm{HNV}$, then $\mathrm{NF}(t) := \prod_{i=1}^n \mathrm{NF}(u_i)$;
- otherwise $\mathrm{NF}(t) := \sum_{u \in \mathrm{HNV}} \mathrm{NF}(u) \cdot \mathcal{D}_t(u)$.

We let $\mathrm{NF}_{\to}(t) = \sup\{\mathrm{NF}(u) \mid t \to^* u\}$ and, if $\mathrm{NF}_{\to}(t) \geq q$, we say that t *yields a normal form with probability at least* q.

For example, for the term t considered above, $\mathrm{NF}_{\to}(t) = \frac{4}{8} = \frac{1}{2}$: four over the eight possible choices for va, vb and vb' yield a normal form (i.e. either choose left for va and vb and choose anything for vb', or choose right for va, left for vb', and choose anything for vb).

4.3 Extending Λ_{PE} with CbV Functions

Λ_{PE} makes it possible to encode a CbV redex like $va.\mathbf{2}(I \oplus_a^0 \Omega)$, as we have seen. However, in view of the functional interpretation of iCPL_0, it would be convenient to also be able to represent the CbV *functions* mapping $va.(u \oplus_a v)$ onto $va.\mathbf{2}(u \oplus_a^0 v)$. A simple way to do this is by enriching the language of Λ_{PE} with a "CbV application" operator $\{t\}u$, with suitable permutative rules. Let $\Lambda_{\mathsf{PE}}^{\{\}}$ indicate the extension of the syntax of Λ_{PE} with the operator $\{\}$. β-reduction for $\Lambda_{\mathsf{PE}}^{\{\}}$ is defined as for Λ_{PE}; permutative reduction $\to_{\mathsf{p}\{\}}$ is defined by all rules in Fig. 4 except for $(\neg v)$, together with three *new* permutations:

$$\{t\}va.u \to_{\mathsf{p}\{\}} va.tu \qquad (\{\}v)$$

$$\{t \oplus_a^i u\}v \to_{\mathsf{p}\{\}} \{t\}v \oplus_a^i \{t\}v \qquad (\{\}\oplus_1)$$

$$\{t\}(u \oplus_a^i v) \to_{\mathsf{p}\{\}} \{t\}u \oplus_a^i \{t\}v \qquad (\{\}\oplus_2)$$

For instance, a CbV Church numeral can be encoded in $\Lambda_{\mathsf{PE}}^{\{\}}$ as $\mathbf{2}^{\mathrm{CbV}} := \lambda f.\{\mathbf{2}\}f$, since one has $\mathbf{2}^{\mathrm{CbV}}(va.u \oplus_a^i v) \to_\beta \{\mathbf{2}\}va.u \oplus_a^i v \to_{\mathsf{p}\{\}} va.\mathbf{2}(u \oplus_a^i v)$.

Several important properties of Λ_{PE} scale to $\Lambda_{\mathsf{PE}}^{\{\}}$:

Proposition 4.4. *$\to_{\mathsf{p}\{\}}$ is confluent and strongly normalizing. Full reduction $\to_{\{\}} := \to_\beta \cup \to_{\mathsf{p}\{\}}$ is confluent.*

Moreover, also the definitions of $\mathcal{D}_t$ and $\mathrm{HNV}_{\to}(t)$ scale to $\Lambda_{\mathsf{PE}}^{\{\}}$ in a natural way (see [4]).

5 THE CORRESPONDENCE, STATICALLY AND DYNAMICALLY

In this section we present the core of our CHC. First, we introduce two type systems $\mathrm{C}\lambda_{\to}$ and $\mathrm{C}\lambda_{\to}^{\{\}}$ for Λ_{PE} and $\Lambda_{\mathsf{PE}}^{\{\}}$, respectively, which extend the simply typed λ-calculus with counting quantifiers. Then, we show that each proof Π in iCPL_0 can be associated with a typing derivation in $\mathrm{C}\lambda_{\to}^{\{\}}$ of some probabilistic term t^Π, in such a way that normalization of Π corresponds to reduction of t^Π. Observe that translating the rule (CE) requires the CbV application operator $\{\}$. However, it is possible to define an alternative "CbN" proof-system for iCPL_0 which translates into $\mathrm{C}\lambda_{\to}$, thus not requiring the operator $\{\}$ (see [4]).

5.1 Two Type Systems with Counting Quantifiers

Both type systems $\mathrm{C}\lambda_{\to}$ and $\mathrm{C}\lambda_{\to}^{\{\}}$ extend the simply typed λ-calculus with counting quantifiers, but in a slightly different way: in $\mathrm{C}\lambda_{\to}$ types are of the form $\mathrm{C}^q\mathbb{s}$, i.e. prefixed by *exactly one* counting

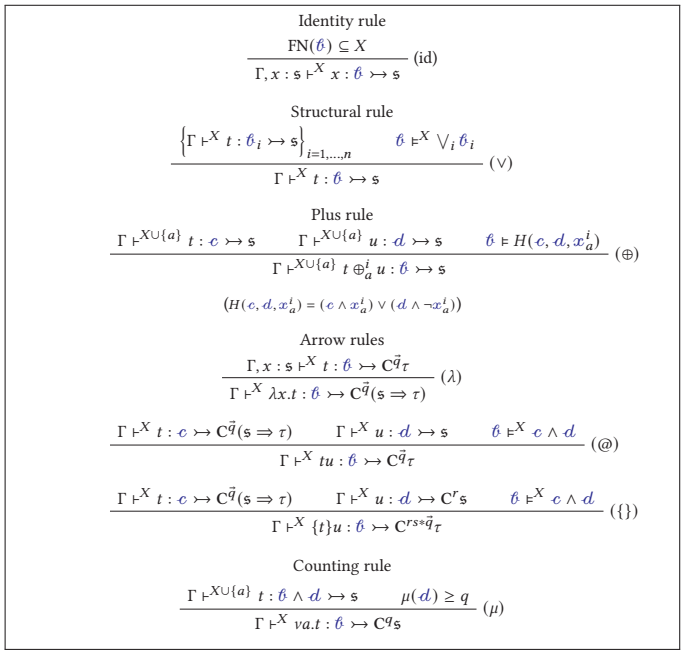

Figure 5: Typing rules of $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$.

quantifier, while in $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$ types are of the form $\mathrm{C}^{\vec{q}}\mathcal{s}$, i.e. prefixed by a (possibly empty) list of counting quantifiers.

More precisely, the types (noted $\mathcal{s}, \mathcal{t}$) of $\mathrm{C}\lambda_{\rightarrow}$ and $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$ are generated by the grammars below:

$$\mathcal{s} ::= \mathrm{C}^{q}\sigma \qquad \sigma ::= o \mid \mathcal{s} \Rightarrow \sigma \qquad (\mathrm{C}\lambda_{\rightarrow})$$

$$\mathcal{s} ::= \mathrm{C}^{\vec{q}}\sigma \qquad \sigma ::= o \mid \mathcal{s} \Rightarrow \sigma \qquad (\mathrm{C}\lambda_{\rightarrow}^{\{\}})$$

where, in both cases, the qs are chosen in $(0, 1] \cap \mathbb{Q}$.

Judgements in both systems are of the form $\Gamma \vdash^X t : \mathcal{b} \rightarrowtail \mathcal{s}$, where Γ is a set of type declarations $x_i : \mathcal{s}_i$ of pairwise distinct variables, t is a term of Λ_{PE} (resp. of $\Lambda_{\mathsf{PE}}^{\{\}}$), $\mathcal{b}$ a Boolean formula, and X a finite set of names with $\mathrm{FN}(t), \mathrm{FN}(\mathcal{b}) \subseteq X$. The intuitive reading of $\Gamma \vdash^X t : \mathcal{b} \rightarrowtail \mathcal{s}$ is that, whenever $\omega \in (2^{\mathbb{N}})^X$ satisfies $\mathcal{b}$, then $\pi_X^{\omega}(t)$ correctly maps programs of type Γ into programs of type $\mathcal{s}$.

The *typing rules* of $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$, which are essentially derived from those of iCPL_0, are illustrated in Fig. 5, where in the rule (μ) it is assumed that $\mathrm{FN}(\mathcal{b}) \subseteq X$, $\mathrm{FN}(d) \subseteq \{a\}$ and $a \notin X$. The rule ($\vee$) allows one to merge n typing derivations for the same term; in particular, with $n = 0$, one has that $\Gamma \vdash^X t : \bot \rightarrowtail \mathcal{s}$ holds for *any* term t. The rules ($\oplus$), (μ) and ({}) are reminiscent, respectively, of the rules (m), (CI) and (CE) of iCPL_0.

The typing rules of $\mathrm{C}\lambda_{\rightarrow}$ coincide with those of $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$ (with $\mathrm{C}^{\vec{q}}$ replaced everywhere by C^{q}), except for the rule ({}), which is obviously absent, and for the rule (μ), which is adapted as follows:

$$\frac{\Gamma \vdash^{X\cup\{a\}} t : \mathcal{b} \wedge d \rightarrowtail \mathrm{C}^{q}\sigma \qquad \mu(d) \geq s}{\Gamma \vdash^X va.t : \mathcal{b} \rightarrowtail \mathrm{C}^{qs}\sigma} \ (\mu')$$

with the similar proviso that $\mathrm{FN}(\mathcal{b}) \subseteq X$, $\mathrm{FN}(d) \subseteq \{a\}$ and $a \notin X$.

Both systems enjoy the subject reduction property:

Proposition 5.1 (Subject Reduction). *If $\Gamma \vdash^X t : \mathcal{b} \rightarrowtail \mathcal{s}$ in $\mathrm{C}\lambda_{\rightarrow}$ (resp. in $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$) and $t \twoheadrightarrow u$ (resp. $t \twoheadrightarrow_{\{\}} u$), then $\Gamma \vdash^X u : \mathcal{b} \rightarrowtail \mathcal{s}$.*

The choice of considering arrow types of the form $\mathrm{C}^{\vec{q}}(\mathcal{s} \Rightarrow \sigma)$, i.e. of never having a counting quantifier *to the right* of $\Rightarrow$ (as in e.g. $\mathcal{s} \Rightarrow \mathrm{C}^{q}\sigma$), was made to let the rule (λ) be permutable over (μ), as required by the permuting rule ($\nu\lambda$).

Example 5.2. Fig. 6a and Fig. 6b illustrate typing derivations in $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$ for a version of the CbN Church numeral $2^{\mathrm{CbN}} = \lambda y.\lambda x.\{y\}(yx)$, with type $\mathrm{C}^{q*q}(\mathrm{C}^{q}(o \Rightarrow o) \Rightarrow (o \Rightarrow o))$, and for the CbV numeral 2^{CbV} with type $\mathrm{C}^{q}(\mathrm{C}^{q}(o \Rightarrow o) \Rightarrow (o \Rightarrow o))$.

Both $\mathrm{C}\lambda_{\rightarrow}$ and $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$ can type *non-normalizable* terms. For example, one can type all terms of the form $I \oplus^i \Omega$ with $\mathrm{C}^{\frac{1}{2}}(o \Rightarrow o)$ in $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$ and with $\mathrm{C}^{\frac{1}{2}}(\mathrm{C}^{1}o \Rightarrow o)$ in $\mathrm{C}\lambda_{\rightarrow}$. Actually, the failure of normalization for typable programs can be ascribed to the rule ($\vee$), as shown by the result below (where we let $\Gamma \vdash_{\neg\vee} t : \mathcal{b} \rightarrowtail \mathcal{s}$ indicate that $\Gamma \vdash t : \mathcal{b} \rightarrowtail \mathcal{s}$ is deduced without using the rule ($\vee$)).

Theorem 5.3 (Deterministic Normalization). *In both $\mathrm{C}\lambda_{\rightarrow}$ and $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$, if $\Gamma \vdash_{\neg\vee} t : \mathcal{b} \rightarrowtail \mathcal{s}$, then t is strongly normalizable.*

As observed in the previous section, restricting to terms of Λ_{PE} having a normal form excludes the most interesting part of the calculus, which is made of terms for which normalization is inherently probabilistic. Similarly, restricting to type derivations without ($\vee$) trivializes the most interesting features of $\mathrm{C}\lambda_{\rightarrow}$ and $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$, that is, their ability to estimate probabilities of termination. We will explore the expressiveness of these systems in this sense in the next section.

5.2 Translating iCPL_0 into $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$.

We now show how derivations in iCPL_0 translate into typing derivations in $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$.[5]

For any formula A of iCPL_0, let us define a corresponding type $\mathcal{s}_A$ by letting $\mathcal{s}_p := o$, $\mathcal{s}_{A\to\mathrm{C}^{\vec{q}}B} := \mathrm{C}^{\vec{q}}(\mathcal{s}_A \Rightarrow \mathcal{s}_B)$ and $\mathcal{s}_{\mathrm{C}^{q}A} := \mathrm{C}^{q}\mathcal{s}_A$. Fig. 7 shows how a derivation Π of $\Gamma \vdash \mathcal{b} \rightarrowtail A$ in iCPL_0 translates into a typing derivation D^{Π} of $\mathcal{s}_\Gamma \vdash t^{\Pi} : \mathcal{b} \rightarrowtail \mathcal{s}_A$ in $\mathrm{C}\lambda_{\rightarrow}^{\{\}}$, with $\mathrm{FN}(t^{\Pi}) \subseteq \mathrm{FN}(\mathcal{b})$, by induction on Π. Notice that we exploit a special constant $\mathbf{c}$ to translate the rule ($\bot$). Moreover, the rule ($\to$E) translates as CbN application tu, while the rule (CE) translates as CbV application $\{t\}u$.

As required by the CHC, normalization steps of iCPL_0 are simulated by $\twoheadrightarrow_{\{\}}$-reductions:

Proposition 5.4 (Stability Under Normalization). *If $\Pi \rightsquigarrow \Pi'$, then $t^{\Pi} \twoheadrightarrow_{\{\}}^{*} t^{\Pi'}$.*

Proof sketch. The normalization step ($\to$I/$\to$E) translates into β-reduction, while the normalization step (CI/CE) in Fig. 3 translates into the following chain of reductions:

$$\{\lambda x^{\mathcal{s}_A}.t^{\Pi}\}va.t^{\Sigma} \twoheadrightarrow_{\mathrm{p}\{\}} va.(\lambda x^{\mathcal{s}_A}.t^{\Pi})t^{\Sigma} \twoheadrightarrow_{\beta} va.t^{\Pi}[t^{\Sigma}/x]$$

All other normalization steps translate into $\twoheadrightarrow_{\mathrm{p}\{\}}$-reductions. □

[5] A similar translation of the "CbN" proof-system for iCPL_0 into $\mathrm{C}\lambda_{\rightarrow}$ is presented in [4].

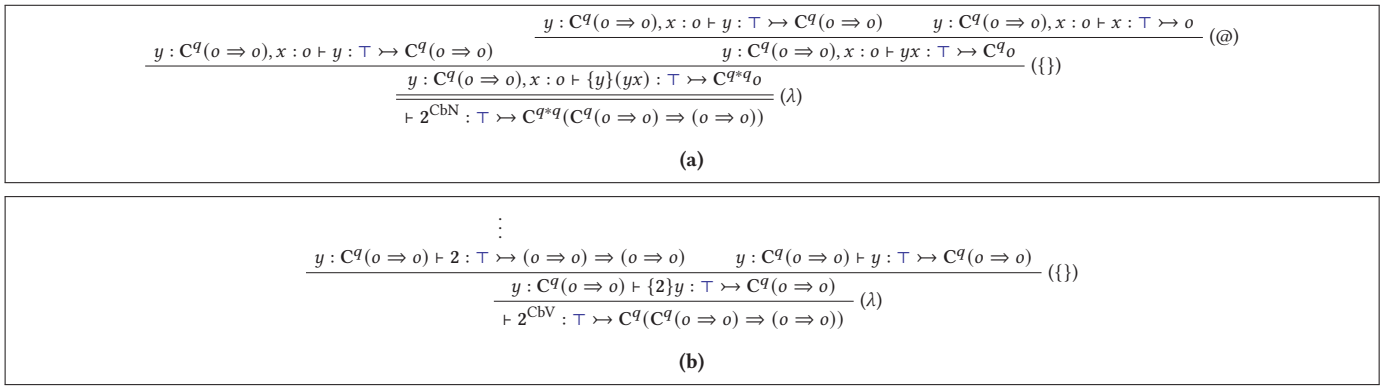

Figure 6: Typing of CbN and CbV Church numerals in $C\lambda^{\{\}}_{\to}$.

Notice that the normalization step (→I/→E) translates into CbN reduction (i.e. plain β-reduction): the "choice" $va.t^{\Sigma}$ is directly substituted, and thus possibly duplicated; instead the (CI/CE) step translates into CbV reduction (i.e. (vf) followed by β-reduction): the generator va is first permuted down and only t^{Σ} is substituted.

By observing that the only use of ($\vee$) coming from the translation introduces a constant $\mathbf{c}$, from Theorem 5.3 we deduce, as promised, the following:

Corollary 5.5. iCPL$_0$ *is strongly normalizing.*

6 FROM TYPE SOUNDNESS TO TYPE COMPLETENESS: INTERSECTION TYPES

In this section we first show that derivations in $C\lambda_{\to}$ and $C\lambda^{\{\}}_{\to}$ provide sound approximations of HNV(t) and NF(t). In order to achieve completeness, we then introduce an extension $C\lambda_{\to,\cap}$ of $C\lambda_{\to}$ with intersection types and show that this system fully captures both deterministic and probabilistic notions of termination for Λ_{PE}.

6.1 From Types to Probability

We already observed, through examples, that if a term t has a type like, e.g. $C^{\frac{1}{2}}(o \Rightarrow o)$, then t has one chance over two of yielding a "correct" program for $o \Rightarrow o$. The result below makes this intuition precise, by showing that the probabilities derived in $C\lambda_{\to}$ and $C\lambda^{\{\}}_{\to}$ are lower bounds for the function $\mathrm{HNV}_{\to}(t)$, that is, for the actual probability of finding a head normalizable term in the distribution $\mathscr{D}_t$.

Theorem 6.1. *If* $\vdash_{C\lambda_{\to}} t : \top \rightarrowtail C^{q}\sigma$, *then* $\mathrm{HNV}_{\to}(t) \geq q$. *If* $\vdash_{C\lambda^{\{\}}_{\to}} t : \top \rightarrowtail C^{q_1 * \cdots * q_n}\sigma$, *then* $\mathrm{HNV}_{\to}(t) \geq \prod_{i=1}^{n} q_i$.

What about reduction to normal form, i.e. the function $\mathrm{NF}_{\to}(t)$? A result like Theorem 6.1 cannot hold in this case. Indeed, consider the term $t = \lambda y.y(I \oplus^{i} \Omega)$. While $\mathrm{NF}(t) = \frac{1}{2}$, $C\lambda^{\{\}}_{\to}$ types t with $\mathfrak{s} = C^{1}(C^{1}(C^{\frac{1}{2}}\sigma \Rightarrow \sigma) \Rightarrow \sigma)$, with $\sigma = o \Rightarrow o$. The problem in this example is that the type $\mathfrak{s}$ contains the "unbalanced" assumption $C^{1}(C^{\frac{1}{2}}\sigma \Rightarrow \sigma)$ (corresponding, in logical terms, to the formula $C^{\frac{1}{2}}A \to C^{1}A$), i.e. it exploits the assumption of the existence of a function turning a $\frac{1}{2}$-correct input into a 1-correct output. Notice that such a function f can only be one that erases its input, and these are the only functions such that tf can reduce to a normal form.

Nevertheless, soundness with respect to NF(t) can be proved, for $C\lambda_{\to}$, by restricting to those types not containing such "unbalanced" assumptions, i.e. to types whose programs cannot increase probabilities.

Definition 6.2. For any type $\mathfrak{s}$ of $C\lambda_{\to}$ of the form $C^{q}\sigma$, let $\lceil \mathfrak{s} \rceil = q$. A type $C^{q}(\mathfrak{s}_1 \Rightarrow \ldots \Rightarrow \mathfrak{s}_n \Rightarrow o)$ of $C\lambda_{\to}$ is *balanced* if all $\mathfrak{s}_i$ are balanced and $q \leq \prod_{i=1}^{n} \lceil \mathfrak{s}_i \rceil$.

Theorem 6.3. *If* $\vdash t : \top \rightarrowtail \mathfrak{s}$ *is derivable in* $C\lambda_{\to}$, *where* $\mathfrak{s}$ *is balanced, then* $\mathrm{NF}_{\to}(t) \geq \lceil \mathfrak{s} \rceil$.

Theorems 6.1 and 6.3 are proved by adapting the standard technique of *reducibility predicates* to the quantitative notion of probabilistic normal form.[6].

6.2 From Probability to Intersection Types

To achieve a type-theoretic characterization of $\mathrm{HNV}_{\to}(t)$ and $\mathrm{NF}_{\to}(t)$, we introduce an extension of $C\lambda_{\to}$ with intersection types. Like those of $C\lambda_{\to}$, the types of $C\lambda_{\to,\cap}$ are of the form $\mathfrak{s} = C^{q}\sigma$, but σ is now defined as:

$$\sigma ::= o \mid \mathsf{n} \mid \mathsf{hn} \mid \mathfrak{M} \Rightarrow \sigma \qquad \mathfrak{M} ::= [\mathfrak{s}, \ldots, \mathfrak{s}]$$

where $[a_1, \ldots, a_n]$ indicates a finite set. While $\mathfrak{M}$ intuitively stands for a finite intersection of types, the new ground types n and hn correspond to the types of normalizable and head-normalizable programs.

We introduce a preorder $\sigma \preceq \tau$ over types by $\alpha \preceq \alpha$, for $\alpha = o, \mathsf{n}, \mathsf{hn}$, $C^{q}\sigma \preceq C^{s}\tau$ if $q \leq s$ and $\sigma \preceq \tau$, and $(\mathfrak{M} \Rightarrow \sigma) \preceq (\mathfrak{N} \Rightarrow \tau)$ if $\sigma \preceq \tau$ and $\mathfrak{N} \preceq^{*} \mathfrak{M}$, where $[\mathfrak{s}_1, \ldots, \mathfrak{s}_n] \preceq^{*} [\mathfrak{t}_1, \ldots, \mathfrak{t}_m]$ holds is there exists an injective function $f : \{1, \ldots, m\} \to \{1, \ldots, n\}$ such that $\mathfrak{s}_{f(i)} \preceq \mathfrak{t}_i$.

A type judgment of $C\lambda_{\to,\cap}$ is of the form $\Gamma \vdash^{X} t : \mathscr{b} \rightarrowtail \mathfrak{s}$, where Γ is made of declarations of the form $x_i : \mathfrak{M}_i$. The typing rules of $C\lambda_{\to,\cap}$ are illustrated in Fig. 8. We omit the ($\vee$)- and ($\oplus$)-rules,

[6]For further details, see [4]

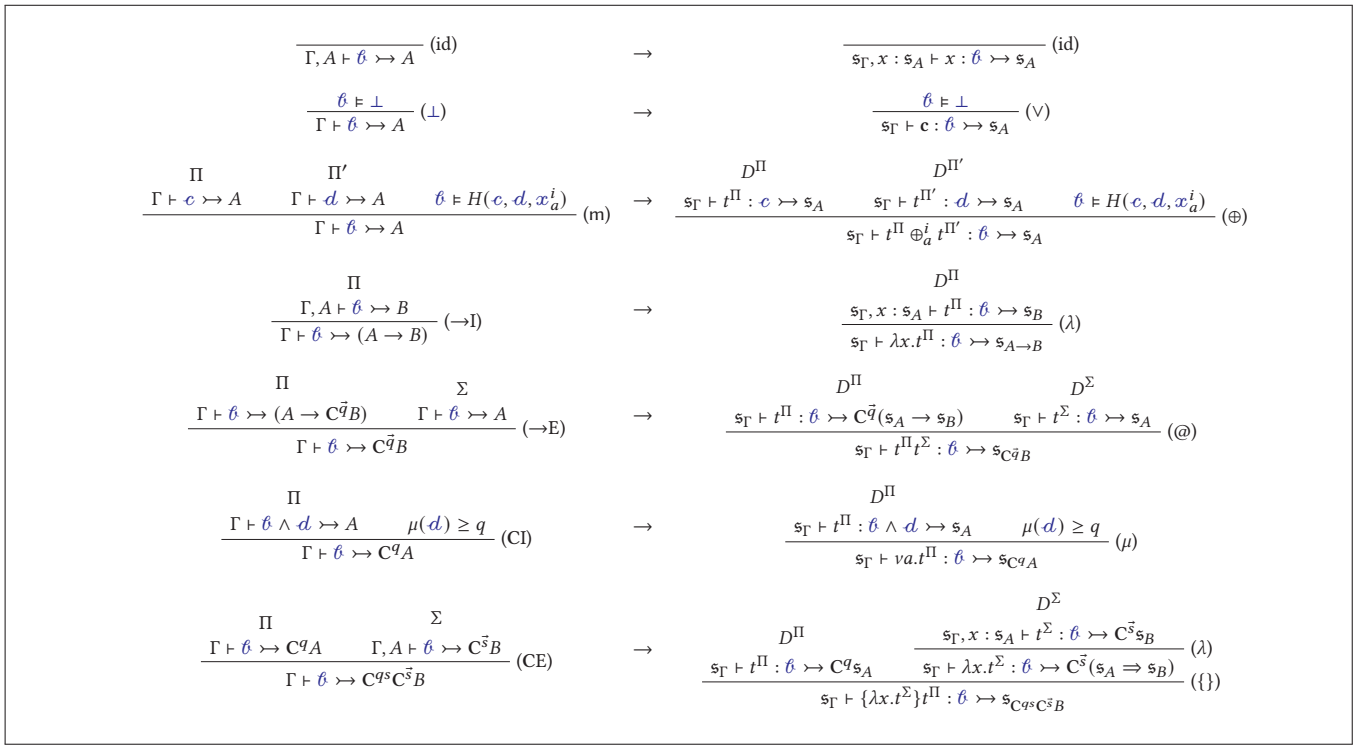

Figure 7: Translation $\Pi \mapsto D^{\Pi}$ from $\mathbf{ND}_{\mathsf{iCPL}_0}$ to $\mathbf{C}\lambda_{\rightarrow}^{\{\}}$.

which are as in $\mathbf{C}\lambda_{\rightarrow}$. In the rule (μ_Σ) it is assumed that a does not occur in $\mathcal{B}$, is the only name in the $\mathcal{A}_i$, and that for $i \neq j$, $\mathcal{A}_i \wedge \mathcal{A}_j \vDash \bot$.

The two rules (hn) and (n) are justified by Proposition 6.5 and Theorem 6.6 below. As rule (n) must warrant a bound on normal forms, following Theorem 6.3, σ has to be *safe*, i.e. balanced[7] and $\{[], \mathsf{hn}\}$-free. The rule $(@_\cap)$ is a standard extension of rule (@) of $\mathbf{C}\lambda_{\rightarrow}$ to finite intersections.

The counting rule (μ_Σ) requires some discussion. The rule admits $n+1$ major premisses expressing typings for t which depend on pairwise disjoint events (the Boolean formulas $\mathcal{A}_i$). This is needed to cope with situations as follows: let

$$t[a,b] = \left((I \oplus_b^1 \Omega) \oplus_b^0 \Omega\right) \oplus_a^0 \left(\Omega \oplus_b^0 I\right)$$

$t[a,b]$ can be given type $\sigma = \mathbf{C}^1(\mathbf{C}^1 o \Rightarrow o)$ under either of the two *disjoint* Boolean constraints $\mathcal{A}_1 := x_a^0 \wedge (x_b^0 \wedge x_b^1)$ and $\mathcal{A}_2 := \neg x_a^0 \wedge \neg x_b^0$. Notice that the term $\nu a.\nu b.t[a,b]$ has probability $\mu(x_a^0)\mu(x_b^0 \wedge x_b^1) + \mu(\neg x_a^0)\mu(\neg x_b^0) = \frac{1}{2}\cdot\frac{1}{4} + \frac{1}{2}\cdot\frac{1}{2} = \frac{3}{8}$ of yielding a head normal value. Yet, in $\mathbf{C}\lambda_{\rightarrow}$, the best we can achieve is $\vdash \nu a.\nu b.t[a,b] : \top \rightarrowtail \mathbf{C}^{\frac{1}{4}}\sigma$, that is, a probability estimation of $\frac{1}{4} < \frac{3}{8}$. Indeed, the rule (μ') forces us to approximate $\mu(x_b^0 \wedge x_b^1)$ and $\mu(\neg x_b^1)$ to a *common* lower bound, i.e. $\frac{1}{4}$, in order to apply a $(\vee)$-rule as illustrated in Fig. 9a. Instead, using (μ_Σ) we can reach the *actual* probability $\frac{3}{8}$, as illustrated in Fig. 9b.

[7] Definition 6.2 extends to the types of $\mathbf{C}\lambda_{\rightarrow,\cap}$ by letting $\lceil \mathsf{hn} \rceil = \lceil [] \rceil = 0$, $\lceil \mathsf{n} \rceil = 1$ and $\lceil [\mathfrak{s}_1, \ldots, \mathfrak{s}_{n+1}] \rceil = \max\{\lceil \mathfrak{s}_i \rceil\}$.

Thanks to the rule (μ_Σ), the generalized counting rule (μ^*) below becomes admissible in $\mathbf{C}\lambda_{\rightarrow,\cap}$:

$$\frac{\Gamma \vdash^{\{a_1,\ldots,a_n\}} t : \mathcal{B} \rightarrowtail \mathbf{C}^q\sigma}{\Gamma \vdash^{\emptyset} \nu a_1.\ldots.\nu a_n.t : \top \rightarrowtail \mathbf{C}^{q\cdot\mu(\mathcal{B})}\sigma} \ (\mu^*)$$

This rule plays an essential role in the completeness results below, together with the standard result that both *subject reduction* and *subject expansion* hold for intersection types.

Proposition 6.4 (Subject Reduction/Expansion). *If* $\Gamma \vdash^X t : \mathcal{B} \rightarrowtail \mathfrak{s}$ *and either* $t \twoheadrightarrow u$ *or* $u \twoheadrightarrow t$, *then* $\Gamma \vdash^X t : \mathcal{B} \rightarrowtail \mathfrak{s}$.

We will now discuss how typings in $\mathbf{C}\lambda_{\rightarrow,\cap}$ capture both deterministic and probabilistic properties of terms. First we have the following facts, which show that the types hn and n capture deterministic termination.

Proposition 6.5 (Deterministic Completeness). *For any closed term* t,

(i.) t *is head-normalizable iff* $\vdash_{\neg\vee} t : \top \rightarrowtail \mathbf{C}^1\mathsf{hn}$;

(ii.) t *is normalizable iff* $\vdash_{\neg\vee} t : \top \rightarrowtail \mathbf{C}^1\mathsf{n}$.

(iii.) t *is strongly normalizable iff* $\vdash_{\neg\vee} t : \top \rightarrowtail \mathbf{C}^1\mathsf{n}$ *and all types in the derivation are safe.*

Proof sketch. Using standard intersection types arguments, it is shown that $\vdash_{\neg\vee} t : \top \rightarrowtail \mathbf{C}^1\mathsf{hn}$ holds for any head-normal t. The first half of (i.) is deduced then using Proposition 6.4. The second half follows from a normalization argument similar to that of Theorem 5.3. Cases (ii.) and (iii.) are similar. □

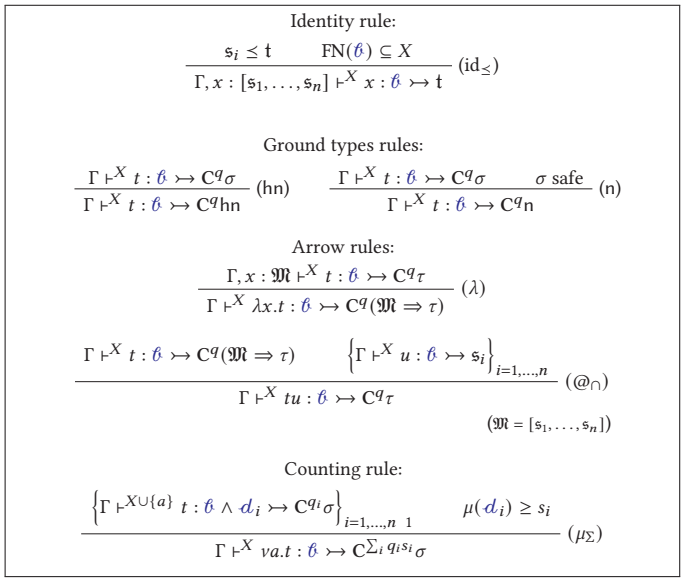

Figure 8: Typing rules of $\mathrm{C}\lambda_{\to,\cap}$.

The probabilistic normalization theorems 6.1 and 6.3 (which extend smoothly to $\mathrm{C}\lambda_{\to,\cap}$) ensure that if t has type $\mathrm{C}^q\mathsf{hn}$ (resp. $\mathrm{C}^q\mathsf{n}$), then $\mathrm{HNV}_{\to_\flat}(t) \geq q$ (resp. $\mathrm{NF}_{\to_\flat}(t) \geq q$). Conversely, $\mathrm{HNV}_{\to_\flat}(t)$ and $\mathrm{NF}_{\to_\flat}(t)$ can be bounded by means of derivations in $\mathrm{C}\lambda_{\to,\cap}$, in the following sense:

Theorem 6.6 (Probabilistic Completeness). *For any closed term t,*

$$\mathrm{HNV}_{\to_\flat}(t) = \sup\{q \mid \vdash t : \top \rightarrowtail \mathrm{C}^q\mathsf{hn}\}$$
$$\mathrm{NF}_{\to_\flat}(t) = \sup\{q \mid \vdash t : \top \rightarrowtail \mathrm{C}^q\mathsf{n}\}$$

Proof sketch. Suppose w.l.o.g. that $t = va_1.\ldots.va_k.t'$. For any $u \in \mathrm{HNV}$ such that $\mathscr{D}_t(u) > 0$, we can deduce $\vdash u : \top \rightarrowtail \mathsf{hn}$. The sequence of probabilistic choices leading to u is finite, and thus captured by a Boolean formula $\mathscr{b}_{t \mapsto u}$. Using subject reduction/expansion we thus deduce $\vdash t' : \mathscr{b}_{t \mapsto u} \rightarrowtail \mathsf{hn}$. Hence, for any finite number of head normal forms $u_1, \ldots, u_n$ such that $\mathscr{D}_t(u_i) > 0$, we deduce $\vdash t' : \mathscr{b}_{t \mapsto u_i} \rightarrowtail \mathsf{hn}$. Using $(\vee)$ and the generalized counting rule (μ^*) we deduce then $\vdash t : \top \rightarrowtail \mathrm{C}^s\mathsf{hn}$, where $s = \sum_{i=1}^n \mu(\mathscr{b}_{t \mapsto u_i}) = \mu(\bigvee_{i=1}^n \mathscr{b}_{t \mapsto u_i})$. The argument for $\mathrm{NF}(t)$ is similar. □

7 RELATED WORK

To the best of our knowledge, our results provide the first clear correspondence between a logical proof system and a probabilistic extension of the λ-calculus. Nevertheless, this does not mean that our logic and calculi come from nowhere.

Different kinds of measure-theoretic quantifiers have been investigated in the literature, with the intuitive meaning that "A is true *for almost all* x", see [44, 54] and more recently [42, 43], or "A is true for the *majority of* x", as for example in [47, 61, 62]. Our use of the term "counting quantifier" comes from [2], where an extension of classical propositional logic with such quantifiers is studied and related to Wagner's counting operator on classes of languages [56–58]. To our knowledge, the present work is the first to apply some form of measure quantifier to typed probabilistic functional programs.

Despite the extensive literature on logical systems enabling - in various ways and for different purposes - some form of probabilistic reasoning, there is not much about logics tied to computational aspects, as iCPL is. Most of the recent logical formalisms have been developed in the realm of modal logic, as for example [5–7, 24, 31, 32, 45, 46]. Another class of probabilistic modal logics have been designed to model Markov chains and similar structures [33, 39, 40]. With the sole exception of *Riesz modal logic* [25], we are not aware of sequent calculi for probability logics.

Intuitionistic modal logic has been related in the Curry-Howard sense to monadic extensions of the λ-calculus [1, 8, 14, 20, 63]. Nevertheless, in these correspondences modal operators are related to *qualitative* properties of programs (typically, tracing algebraic effects), as opposed to the *quantitative* properties expressed by counting quantifiers. However, our Kripke structures for iCPL can be related to standard Kripke structures for intuitionistic modal logic [49, 52]. These are based on a set W with *two* pre-order relations, $\leq$ and R, enjoying a suitable "diamond" property $R; \leq \subseteq \leq; R$. We obtain a similar structure by considering worlds to be pairs w, ω made of a world and an outcome from the Cantor space, with $(w, \omega) \leq (w', \omega)$, whenever $w \leq w'$, and $(w, \omega) R (w, \omega + \omega')$. The clause for $\mathrm{C}^q A$ can then be seen as a quantitative variant of the corresponding clause for $\Diamond A$. Actually, this is not very surprising, given the similarity between the introduction and elimination rules for C^q and those for $\Diamond$, see e.g. [1, 8].

On the other hand, quantitative semantics arising from linear logic have been largely used in the study of probabilistic λ-calculi, e.g. [16, 21, 23]. Notably, *probabilistic coherence spaces* [21, 22, 28] have been shown to provide a fully abstract model of probabilistic PCF. While we are not aware of correspondences relating probabilistic programs with proofs in linear logic, it seems that the proof-theory of counting quantifiers could somehow be related to that of *bounded exponentials* [19, 30] and, more generally, to the theory of *graded* monads and comonads [12, 26, 36, 37].

The calculus Λ_{PE} is strongly inspired by [18], where a simple type system is also introduced. However, since typings in this system ensure strong normalization, they do not provide information about probability of termination for non-normalizable terms. Several other type systems for probabilistic λ-calculi, rather focused on capturing genuinely probabilistic properties of normalization, have been recently introduced. Among these, we can certainly mention systems based on *type distributions* [17], where a single derivation assigns several types to a term, each with some probability, and systems based on *oracle intersection types* [11], where type derivations capture single evaluations as determined by an oracle. Our type systems sit in between these two approaches: like the first (and unlike the second), typing derivations can capture a finite number of different evaluations, although without using distributions of types; like the second, typings reflect the dependency of evaluation on oracles, although the latter are manipulated in an aggregate way by means of Boolean constraints. Finally, in [60] dependent type theory is enriched with a probabilistic choice operator, yielding a calculus

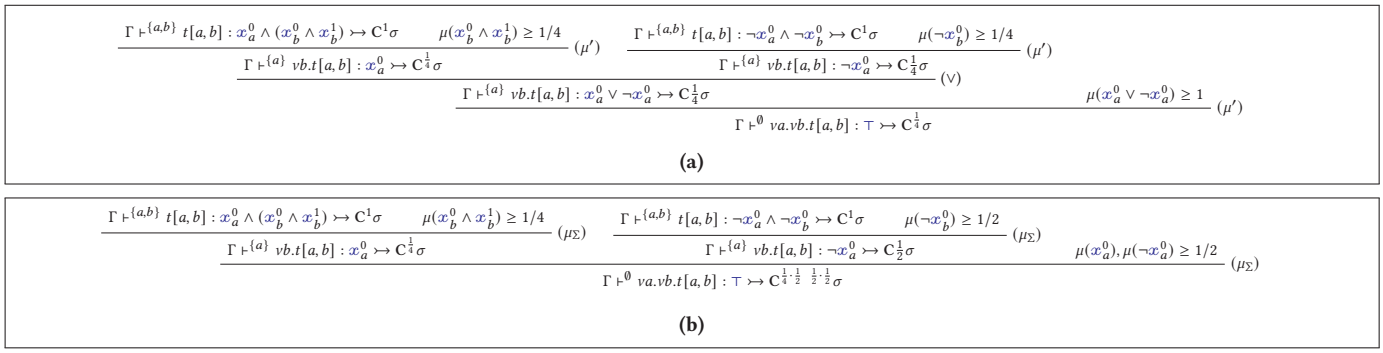

Figure 9: Comparing probabilities derived with the rules (μ') and (μ_Σ).

with both term and type distributions. Interestingly, a fragment of this system enjoys a sort of CHC with so-called *Markov Logic Networks* [51], a class of probabilistic graphical models specified by means of first-order logic formulas.

8 CONCLUSIONS

The main contribution of this work consists in defining a Curry-Howard correspondence between a logic with counting quantifiers and a type system providing lower bounds to the probability of termination. Moreover, in analogy with what happens in the deterministic case, extending the type system with an intersection operator leads to a full characterization of probability of termination. Even though intersection types do not have a clear logical counterpart, the existence of this extension convinces us that the correspondence here introduced is meaningful. The possibility of defining a Curry-Howard correspondence relating algebraic effects, on the program side, with a modal operator, on the logical side, is certainly not surprising. Instead, it seems to us that the main (and unexpected!) contribution of this work is showing that the peculiar features of probabilistic effects can be managed in an elegant way using ideas coming from logic and proof theory.

Among the many avenues of research that this work opens, the study of type inference must certainly be mentioned, as well as the extension of the correspondence to polymorphic types or to control operators. Particularly intriguing, then, is the possibility of studying systems of intersection types supporting program synthesis, again in analogy with what is already known in the framework of deterministic computation [9, 34].

ACKNOWLEDGMENTS

This work is supported by the ERC CoG "DIAPASON" under Grant No. 818616, and by the ANR PRC project "PPS", ANR-19-CE48-0014.

REFERENCES

[1] N. Alechina, M. Mendler, V. de Paiva, and E. Ritter. 2001. Categorical and Kripke Semantics for Constructive S4 Modal Logic. In *Proc. of CSL 2021*. Springer, Berlin, Heidelberg, 292–307.
[2] M. Antonelli, U. Dal Lago, and P. Pistone. 2021. On Counting Propositional Logic and Wagner's Hierarchy. In *Proc. of ICTCS 2021*, Vol. 3072. CEUR Workshop Proceedings, Aachen, 107–121.
[3] M. Antonelli, U. Dal Lago, and P. Pistone. 2021. On Measure Quantifiers in First-Order Arithmetic. In *Proc. of CiE 2021*. Springer, Switzerland, 12–24.
[4] M. Antonelli, U. Dal Lago, and P. Pistone. 2022. Curry and Howard Meet Borel (Long Version). (March 2022). https://arxiv.org/abs/2203.11265.
[5] F. Bacchus. 1990. Lp, a Logic for Representing and Reasoning with Statistical Knowledge. *Computational Intelligence* 6, 4 (1990), 209–231.
[6] F. Bacchus. 1990. On Probability Distributions over Possible Worlds. In *Uncertainty in Artificial Intelligence*. Machine Intelligence and Pattern Recognition, Vol. 9. North-Holland, Amsterdam, 217–226.
[7] F. Bacchus. 1990. *Representing and Reasoning with Probabilistic Knowledge*. MIT Press, Cambridge MA.
[8] N. P. Benton, G. M. Bierman, and V. C. V De Paiva. 1998. Computational types from a logical perspective. *Journal of Functional Programming* 8, 2 (1998), 177–193.
[9] J. Bessai, A. Dudenhefner, B. Düdder, M. Martens, and J. Rehof. 2016. Combinatory Process Synthesis. In *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques*, T. Margaria and B. Steffen (Eds.). Springer International Publishing, Cham, 266–281.
[10] O. Bournez and C. Kirchner. 2002. Probabilistic Rewrite Strategies. Applications to ELAN. In *Proc. of RTA 2002*. Springer, Berlin, Heidelberg, 252–266.
[11] F. Breuvart and U. Dal Lago. 2018. On Intersection Types and Probabilistic Lambda Calculi. In *Proc. of PPDP 2018*. Association for Computing Machinery, New York, NY, USA, 1–13.
[12] A. Brunel, M. Gaboardi, D. Mazza, and S. Zdancewic. 2014. A Core Quantitative Coeffect Calculus. In *Proc. of ESOP 2014*. Springer, Berlin, Heidelberg, 351–370.
[13] L. Caires, F. Pfenning, and B. Toninho. 2016. Linear logic propositions as session types. *Mathematical Structures in Computer Science* 26, 3 (2016), 367–423.
[14] P.-L. Curien, M. Fiore, and G. Munch-Maccagnoni. 2016. A Theory of Effects and Resources: Adjunction Models and Polarised Calculi. In *Proc. of POPL 2016*. Association for Computing Machinery, New York, NY, 44–56.
[15] H. B. Curry and R. Feys. 1958. *Combinatory Logic. Vol. I.* North-Holland, Amsterdam.
[16] U. Dal Lago, C. Faggian, B. Valiron, and A. Yoshimizu. 2017. The Geometry of Parallelism: Classical, Probabilistic, and Quantum Effects. In *Proc. of POPL 2017*. Association for Computing Machinery, New York, NY, 833–845.
[17] U. Dal Lago and C. Grellois. 2019. Probabilistic Termination by Monadic Affine Sized Typing. *ACM Transactions on Programming Languages and Systems* 41, 2 (2019), 10–65.
[18] U. Dal Lago, G. Guerrieri, and W. Heijltjes. 2020. Decomposing Probabilistic Lambda-Calculi. In *Proc. of FoSSaCS 2020*. Springer, Cham, 136–156.
[19] U. Dal Lago and M. Hofmann. 2009. Bounded Linear Logic, Revisited. In *Proc. of TLCA 2009*. Springer, Berlin Heidelberg, 80–94.
[20] R. Davies and F. Pfenning. 2001. A Modal Analysis of Staged Computation. *J. ACM* 48, 3 (2001), 555–604.
[21] T. Ehrhard and C. Tasson. 2018. Probabilistic Call by Push Value. *Logical Methods in Computer Science* 15, 1 (2018), 1–46.
[22] T. Ehrhard, C. Tasson, and M. Pagani. 2014. Probabilistic Coherence Spaces Are Fully Abstract for Probabilistic PCF. In *Proc. of POPL 2014*. Association for Computing Machinery, New York, NY, 309–320.
[23] C. Faggian and S. Ronchi della Rocca. 2019. Lambda-Calculus and Probabilistic Computation. In *Proc. of LICS 2019*. IEEE, Vancouver, Canada, 1–13.
[24] R. Fagin and J.Y. Halpern. 1994. Reasoning about Knowledge and Probability. *J. ACM* 41, 2 (1994), 340–367.
[25] R. Furber, R. Mardare, and M. Mio. 2020. Probabilistic logics based on Riesz spaces. *Logical Methods in Computer Science* 16, 1 (2020), 6:1–6:45.
[26] D. R. Ghica and A. I. Smith. 2014. Bounded Linear Types in a Resource Semiring. In *Proc. of ESOP 2014*. Springer, Berlin, Heidelberg, 331–350.
[27] J.-Y. Girard. 1972. *Interprétation fonctionnelle et élimination des coupures de l'arithmetique d'ordre supérieur*. Ph.D. Dissertation. Université Paris VII.

[28] J.-Y. Girard. 2004. *Between Logic and Quantic: a Tract.* London Mathematical Society Lecture Note Series, Vol. 316. Cambridge University Press, Cambridge, 346–381.

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

[30] J.-Y. Girard, A. Scedrov, and P.J. Scott. 1992. Bounded linear logic: a Modular Approach to Polynomial-Time Computability. *Theoretical Computer Science* 97, 1 (1992), 1–66.

[31] J.Y. Halpern. 1990. An Analysis of First-Order Logics for Probability. *Artificial Intelligence* 46, 3 (1990), 311–350.

[32] J.Y. Halpern. 2003. *Reasoning About Uncertainty.* MIT Press, Cambridge MA.

[33] H. Hansson and B. Jonsson. 1994. A logic for reasoning about time and reliability. *Formal Aspects of Computing* 6, 5 (1994), 512–535.

[34] F. Henglein and J. Rehof. 2016. Modal Intersection Types, Two-Level Languages, and Staged Synthesis. In *Semantics, Logics, and Calculi: Essays Dedicated to Hanne Riis Nielson and Flemming Nielson on the Occasion of Their 60th Birthdays.* Springer International Publishing, Cham, 289–312.

[35] W.A. Howard. 1980. The Formula-as-Types Notion of Construction (1969). In *To H.B. Curry. Essays on Combinatory Logic, Lambda Calculus and Formalism.* Academic Press, Cambridge, MA.

[36] S. Katsumata. 2014. Parametric Effect Monads and Semantics of Effect Systems. In *Proc. of POPL 2014.* Association for Computing Machinery, New York, NY, 633–645.

[37] S. Katsumata. 2018. A Double Category Theoretic Analysis of Graded Linear Exponential Comonads. In *Proc. of FoSSaCS 2018.* Springer International Publishing, Cham, 110–127.

[38] N. Kobayashi, U. Dal Lago, and C. Grellois. 2019. On the Termination Problem for Probabilistic Higher-Order Recursive Programs. In *Proc. of LICS 2019.* IEEE, Vancouver, France, 1–14.

[39] D. Kozen. 1981. Semantics of probabilistic programs. *Journal of Computer and System Science* 22, 3 (1981), 328 – 350.

[40] D. Lehmann and S. Shelah. 1982. Reasoning with time and chance. *Information and Control* 53, 3 (1982), 165 – 198.

[41] P. Martin-Löf. 1975. An Intuitionistic Theory of Types: Predicative Part. In *Proc. of Logic Colloquium '73*, Vol. 80. North-Holland, Amsterdam, 73–118.

[42] H. Michalewski and M. Mio. 2016. Measure Quantifiers in Monadic Second Order Logic. In *Proc. of LFCS 2016.* Springer, Cham, 267–282.

[43] M. Mio, M. Skrzypczak, and H. Michalewski. 2012. Monadic Second Order Logic with Measure and Category Quantifiers. *Logical Methods in Computer Science* 8, 2 (2012), 1–29.

[44] C.F. Morgenstern. 1979. The Measure Quantifier. *Journal of Symbolic Logic* 44, 1 (1979), 103–108.

[45] N. J. Nilsson. 1986. Probabilistic Logic. *Artificial Intelligence* 28, 1 (1986), 71–87.

[46] N. J. Nilsson. 1993. Probabilistic Logic Revisited. *Artificial Intelligence* 59, 1/2 (1993), 39–42.

[47] C.H. Papadimitriou. 1985. Games Against Nature. *Journal of Computer and System Science* 31, 2 (1985), 288–301.

[48] M. Parigot. 1992. $\lambda\mu$-Calculus: An Algorithmic Interpretation of Classical Natural Deduction. In *Proc. of LPAR 1992.* Springer, Berlin Heidelberg, 190–201.

[49] G. Plotkin and C. Stirling. 1986. A Framework for Intuitionistic Modal Logics: Extended Abstract. In *Proc. of TARK '86.* Morgan Kaufmann Publishers Inc., San Francisco, CA, 399–406.

[50] M. L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, Inc., USA.

[51] M. Richardson and P. Domingos. 2006. Markov Logic Networks. *Machine Learning* 62, 1 (2006), 107–136.

[52] A.K. Simpson. 1994. *The Proof Theory and Semantics of Intuitionistic Modal Logic.* Ph.D. Dissertation. University of Edinburgh.

[53] M.H. Sorensen and P. Urzyczyn. 2006. *Lectures on the Curry-Howard isomorphism.* Studies in Logic and the Foundations of Mathematics, Vol. 149. Elsevier, Amsterdam.

[54] C.I. Steinhorn. 1985. Borel Structures and Measure and Category Logics. In *Model-Theoretic Logics.* Vol. 8. Springer-Verlag, New York, Berlin, Heidelberg, Tokyo, 579–596.

[55] P. Wadler. 2012. Propositions as Sessions. In *Proc. of ICFP 2012.* Association for Computing Machinery, New York, NY, 273–286.

[56] K.W. Wagner. 1984. Compact descriptions and the counting polynomial-time hierarchy. In *Frege Conference 1984: Proceedings of the International Conference held at Schwerin (Mathematische Forschung, Vol. 20).* Akademie-Verlag, Berlin, 383–392.

[57] K.W. Wagner. 1986. Some Observations on the Connection Between Counting and Recursion. *Theoretical Computer Science* 47 (1986), 131–147.

[58] K.W. Wagner. 1986. The Complexity of Combinatorial Problems with Succinct Input Representation. *Acta Informatica* 23 (1986), 325–356.

[59] D. Wang, D. M. Kahn, and J. Hofmann. 2020. Raising Expectations: Automating Expected Cost Analysis with Types. In *Proc. of ICFP 2020*, Vol. 4. Association for Computing Machinery, New York, NY, 1–31.

[60] J. Warrell and M. B. Gerstein. 2018. Dependent Type Networks: A Probabilistic Logic via the Curry-Howard Correspondence in a System of Probabilistic Dependent Types. (2018). unpublished manuscript, http://papers.gersteinlab.org/papers/UDL-19/index-all.html.

[61] S. Zachos. 1988. Probabilistic Quantifiers and Games. *Journal of Computer and System Science* 36, 3 (1988), 433–451.

[62] S. Zachos and H. Heller. 1986. A decisive characterization of BPP. *Information and Control* 1–3 (1986), 125–135.

[63] N. Zyuzin and A. Nanevski. 2021. Contextual Modal Types for Algebraic Effects and Handlers. In *Proc. of ICFP 2021.* Association for Computing Machinery, New York, NY, 1–29.