On Coinductive Equivalences for Higher-Order Probabilistic Functional Programs

Ugo Dal Lago Davide Sangiorgi
Università di Bologna & INRIA
{dallago,sangio}@cs.unibo.it

Michele Alberti
Institut de Mathématiques de Luminy
CNRS & Université d'Aix-Marseille
michele.alberti@univ-amu.fr

Abstract

We study bisimulation and context equivalence in a probabilistic λ-calculus. The contributions of this paper are threefold. Firstly we show a technique for proving congruence of *probabilistic applicative bisimilarity*. While the technique follows Howe's method, some of the technicalities are quite different, relying on non-trivial "disentangling" properties for sets of real numbers. Secondly we show that, while bisimilarity is in general strictly finer than context equivalence, coincidence between the two relations is attained on pure λ-terms. The resulting equality is that induced by *Levy-Longo trees*, generally accepted as the finest extensional equivalence on pure λ-terms under a lazy regime. Finally, we derive a coinductive characterisation of context equivalence on the whole probabilistic language, via an extension in which terms akin to distributions may appear in redex position. Another motivation for the extension is that its operational semantics allows us to experiment with a different congruence technique, namely that of *logical bisimilarity*.

Categories and Subject Descriptors F.3.1 [*Logics and Meaning of Programs*]: Specifying and Verifying and Reasoning about Programs—logics of programs; F.3.2 [*Logics and Meaning of Programs*]: Semantics of Programming Languages—operational semantics.

General Terms Theory

Keywords Bisimulation; Probabilistic Lambda Calculus; Coinduction; Howe's Technique.

1. Introduction

Probabilistic models are more and more pervasive. Not only are they a formidable tool when dealing with uncertainty and incomplete information, but they sometimes are a *necessity* rather than an option, like in computational cryptography (where, e.g., secure public key encryption schemes need to be probabilistic [17]). A nice way to deal computationally with probabilistic models is to allow probabilistic choice as a primitive when designing algorithms, this way switching from usual, deterministic computation to a new paradigm, called probabilistic computation. Examples of application areas in which probabilistic computation has proved to be useful include natural language processing [29], robotics [44], computer vision [8], and machine learning [33].

This new form of computation, of course, needs to be available to programmers to be accessible. And indeed, various probabilistic programming languages have been introduced in the last years, spanning from abstract ones [24, 32, 37] to more concrete ones [18, 34], being inspired by various programming paradigms like imperative, functional or even object oriented. A quite common scheme consists in endowing any deterministic language with one or more primitives for probabilistic choice, like binary probabilistic choice or primitives for distributions.

One class of languages that cope well with probabilistic computation are functional languages. Indeed, viewing algorithms as functions allows a smooth integration of distributions into the playground, itself nicely reflected at the level of types through monads [20, 37]. As a matter of fact, many existing probabilistic programming languages [18, 34] are designed around the λ-calculus or one of its incarnations, like Scheme.

The focus of this paper are operational techniques for understanding and reasoning about program equality in higher-order probabilistic languages. Checking computer programs for equivalence is a crucial, but challenging, problem. Equivalence between two programs generally means that the programs should behave "in the same manner" under any context. Specifically, two λ-terms are *context equivalent* if they have the same convergence behavior (i.e., they do or do not terminate) in any possible context. Finding effective methods for context equivalence proofs is particularly challenging in higher-order languages.

Bisimulation has emerged as a very powerful operational method for proving equivalence of programs in various kinds of languages, due to the associated coinductive proof method. To be useful, the behavioral relation resulting from bisimulation — *bisimilarity* — should be a *congruence*, and should also be sound with respect to context equivalence. Bisimulation has been transplanted onto higher-order languages by Abramsky [1]. This version of bisimulation, called *applicative bisimulation* has received considerable attention [19, 27, 28, 35, 36, 39]. In short, two functions M and N are applicative bisimilar when their applications MP and NP are applicative bisimilar for any argument P.

Often, checking a given notion of bisimulation to be a congruence in higher-order languages is nontrivial. In the case of applicative bisimilarity, congruence proofs usually rely on Howe's method [22]. Other forms of bisimulation have been proposed, such as environmental bisimulation and logical bisimulation [25, 41, 42], with the goal of relieving the burden of the proof of congruence, and of accommodating language extensions.

In this work, we consider the pure λ-calculus extended with a probabilistic choice operator. Context equivalence of two terms

POPL '14, January 22–24, 2014, San Diego, CA, USA.
Copyright © 2014 ACM 978-1-4503-2544-8/14/01…$15.00.
http://dx.doi.org/10.1145/2535838.2535872

means that they have the same *probability of convergence* in all contexts. The objective of the paper is to understand context equivalence and bisimulation in this paradigmatic probabilistic higher-order language, called $\Lambda_\oplus$.

The paper contains three main technical contributions. The first is a proof of congruence for probabilistic applicative bisimilarity along the lines of Howe's method. This technique consists in defining, for every relation on terms $\mathcal{R}$, its Howe's lifting $\mathcal{R}^H$. The construction, essentially by definition, ensures that the relation obtained by lifting bisimilarity is a congruence; the latter is then proved to be itself a bisimulation, therefore coinciding with applicative bisimilarity. Definitionally, probabilistic applicative bisimulation is obtained by setting up a labelled Markov chain on top of λ-terms, then adapting to it the coinductive scheme introduced by Larsen and Skou in a first-order setting [26]. In the proof of congruence, the construction $(\cdot)^H$ closely reflects analogous constructions for nondeterministic extensions of the λ-calculus. The novelties are in the technical details for proving that the resulting relation is a bisimulation: in particular our proof of the so-called Key Lemma — an essential ingredient in Howe's method — relies on non-trivial "disentangling" properties for sets of real numbers, these properties themselves proved by modeling the problem as a flow network and then apply the Max-flow Min-cut Theorem. The congruence of applicative bisimilarity yields soundness with respect to context equivalence as an easy corollary. Completeness, however, fails: applicative bisimilarity is proved to be finer. A subtle aspect is also the late vs. early formulation of bisimilarity; with a choice operator the two versions are semantically different; our construction crucially relies on the late style.

In our second main technical contribution we show that the presence of higher-order functions and probabilistic choice in contexts gives context equivalence and applicative bisimilarity maximal discriminating power on pure λ-terms. We do so by proving that, on pure λ-terms, both context equivalence and applicative bisimilarity coincide with the *Levy-Longo tree equality*, which equates terms with the same Levy-Longo tree (briefly LLT), and is generally accepted as the finest extensional equivalence on pure λ-terms under a *lazy* regime. The result is in sharp contrast with what happens under a nondeterministic interpretation of choice (or in the absence of choice), where context equivalence is coarser than LLT equality.

Our third main contribution is a coinductive characterisation of probabilistic context equivalence on the whole language $\Lambda_\oplus$ (as opposed to the subset of pure λ-terms). We obtain this result by setting a bisimulation game on an extension of $\Lambda_\oplus$ in which weighted formal sums — terms akin to distributions — may appear in redex position. Thinking of distributions as sets of terms, the construction reminds us of the reduction of nondeterministic to deterministic automata. The technical details are however quite different, because we are in a higher-order language and therefore — once more — we are faced with the congruence problem for bisimulation, and because formal sums may contain an *infinite* number of terms. For the proof of congruence of bisimulation in this extended language, we have experimented the technique of logical bisimulation. In this method (and in the related method of environmental bisimulation), the clauses of applicative bisimulation are modified so to allow the standard congruence argument for bisimulations in first-order languages, where the bisimulation method itself is exploited to establish that the closure of the bisimilarity under contexts is again a bisimulation. Logical bisimilarities have two key elements. First, bisimilar functions may be tested with *bisimilar* (rather than identical) arguments (more precisely, the arguments should be in the context closure of the bisimulation; the use of contexts is necessary for soundness). Secondly, the transition system should be small-step, deterministic (or at least confluent), and the bisimulation game should also be played on internal moves. In our probabilistic setting, the ordinary logical bisimulation game has to be modified substantially. Formal sums represent possible evolutions of running terms, hence they should appear in redex position only (allowing them anywhere would complicate matters considerably). The obligation of redex position for certain terms is in contrast with the basic schema of logical bisimulation, in which related terms can be used as arguments to bisimilar functions and can therefore end up in arbitrary positions. We solve this problem by moving to *coupled logical bisimulations*, where a bisimulation is formed by a pair of relations, one on $\Lambda_\oplus$-terms, the other on terms extended with formal sums. The bisimulation game is played on both relations, but only the first relation is used to assemble input arguments for functions.

Another delicate point is the meaning of internal transitions for formal sums. In logical bisimilarity the transition system should be small-step; and formal sums should evolve into values in a finite number of steps, even if the number of terms composing the formal sum is infinite. We satisfy these requirements by defining the transition system for extended terms on top of that for $\Lambda_\oplus$-terms. The proof of congruence of coupled logical bisimilarity also exploits an "up-to distribution" bisimulation proof technique.

In the paper we adopt call-by-name evaluation. The results on applicative bisimilarity can be transported onto call-by-value; in contrast, transporting the other results is more problematic, and we leave it for future work. See Section 6 for more details.

An extended version of this paper is available [10].

1.1 Further Related Work

Research on (higher-order) probabilistic functional languages have, so far, mainly focused on either new programming constructs, or denotational semantics, or applications. The underlying operational theory, which in the ordinary λ-calculus is known to be very rich, has remained so far largely unexplored. In this section, we give some pointers to the relevant literature on probabilistic λ-calculi, without any hope of being exhaustive.

Various probabilistic λ-calculi have been proposed, starting from the pioneering work by Saheb-Djahromi [38], followed by more advanced studies by Jones and Plotkin [24]. Both these works are mainly focused on denotational semantics. More recently, there has been a revamp on this line of work, with the introduction of adequate (and sometimes also fully-abstract) denotational models for probabilistic variations of PCF [11, 16]. There is also another thread of research in which various languages derived from the λ-calculus are given types in monadic style, allowing this way to nicely model concrete problems like Bayesian inference and probability models arising in robotics [20, 32, 37]; these works however, do not study operationally based theories of program equivalence.

Nondeterministic extensions of the λ-calculus have been analysed in typed calculi [3, 27, 43] as well as in untyped calculi [6, 13, 23, 30]. The emphasis in all these works is mainly domain-theoretic. Apart from [30], all cited authors closely follow the testing theory [12], in its modalities *may* or *must*, separately or together. Ong's approach [30] inherits both testing and bisimulation elements.

Our definition of applicative bisimulation follows Larsen and Skou's scheme [26] for fully-probabilistic systems. Many other forms of probabilistic bisimulation have been introduced in the literature, but their greater complexity is usually due to the presence of *both* nondeterministic and probabilistic behaviors, or to continuous probability distributions. See surveys such as [5, 21, 31].

Contextual characterisations of LLT equality include [7], in a λ-calculus with multiplicities in which deadlock is observable, and [15], in a λ-calculus with choice, parallel composition, and both call-by-name and call-by-value applications. See [14] for a survey on observational characterisations of λ-calculus trees.

2. Preliminaries

2.1 A Pure, Untyped, Probabilistic Lambda Calculus

Let $\mathsf{X} = \{x, y, \ldots\}$ be a denumerable set of variables. The set $\Lambda_\oplus$ of *term expressions*, or *terms* is defined as follows:

$$M, N, L \ ::= \ x \ \Big| \ \lambda x.M \ \Big| \ MN \ \Big| \ M \oplus N,$$

where $x \in \mathsf{X}$. The only non-standard operator in $\Lambda_\oplus$ is probabilistic choice: $M \oplus N$ is a term which is meant to behave as either M or N, each with probability $\frac{1}{2}$. A more general construct $M \oplus_p N$ where p is any (computable) real number from $[0, 1]$, is derivable, given the universality of the λ-calculus (see, e.g., [9]). The set of free variables of a term M is indicated as $\mathsf{FV}(M)$ and is defined as usual. Given a finite set of variables $\overline{x} \subseteq \mathsf{X}$, $\Lambda_\oplus(\overline{x})$ denotes the set of terms whose free variables are among the ones in $\overline{x}$. A term M is *closed* if $\mathsf{FV}(M) = \emptyset$ or, equivalently, if $M \in \Lambda_\oplus(\emptyset)$. The (capture-avoiding) substitution of N for the free occurrences of x in M is denoted $M\{{}^{N}\!/x\}$. We sometimes use the identity term $I \overset{\text{def}}{=} \lambda x.x$, the projector $K \overset{\text{def}}{=} \lambda x.\lambda y.x$, and the purely divergent term $\Omega \overset{\text{def}}{=} (\lambda x.xx)(\lambda x.xx)$.

Terms are now given a call-by-name semantics following [9]. A term is a *value* if it is a closed λ-abstraction. We call $\mathsf{V}\Lambda_\oplus$ the set of all values. Values are ranged over by metavariables like V, W, X. Closed terms evaluates not to a single value, but to a *(partial) value distribution*, that is, a function $\mathscr{D} : \mathsf{V}\Lambda_\oplus \to \mathbb{R}_{[0,1]}$ such that $\sum_{V \in \mathsf{V}\Lambda_\oplus} \mathscr{D}(V) \leq 1$. Distributions do not necessarily sum to 1, so to model the possibility of (probabilistic) divergence. Given a value distribution $\mathscr{D}$, its *support* $\mathsf{S}(\mathscr{D})$ is the subset of $\mathsf{V}\Lambda_\oplus$ whose elements are values to which $\mathscr{D}$ attributes positive probability. Value distributions ordered pointwise form both a lower semilattice and an $\omega\mathbf{CPO}$: limits of ω-chains always exist. Given a value distribution $\mathscr{D}$, its *sum* $\sum \mathscr{D}$ is $\sum_{V \in \mathsf{V}\Lambda_\oplus} \mathscr{D}(V)$.

The call-by-name semantics of a closed term M is a value distribution $[\![M]\!]$ defined in one of the ways explained in [9]. We recall this now, though only briefly for lack of space. The first step consists in defining a formal system deriving finite *lower approximations* to the semantics of M. Big-step approximation semantics, as an example, derives judgments in the form $M \Downarrow \mathscr{D}$, where M is a term and $\mathscr{D}$ is a value distribution of finite support (see Figure 1). Small-step approximation semantics can be defined similarly, and derives judgments in the form $M \Rightarrow \mathscr{D}$. Noticeably, big-step and small-step can simulate each other, i.e. if $M \Downarrow \mathscr{D}$, then $M \Rightarrow \mathscr{E}$ where $\mathscr{E} \geq \mathscr{D}$, and *vice versa* [9]. In the second step, $[\![M]\!]$, called the *semantics* of M, is set as the least upper bound of distributions obtained in either of the two ways:

$$[\![M]\!] \overset{\text{def}}{=} \sup_{M \Downarrow \mathscr{D}} \mathscr{D} = \sup_{M \Rightarrow \mathscr{D}} \mathscr{D}.$$

Notice that the above is well-defined because for every M, the set of all distributions $\mathscr{D}$ such that $M \Downarrow \mathscr{D}$ is directed, and thus its least upper bound is a value distribution because of ω-completeness.

Example 2.1. *Consider the term $M \overset{\text{def}}{=} I \oplus (K \oplus \Omega)$. We have $M \Downarrow \mathscr{D}$, where $\mathscr{D}(I) = \frac{1}{2}$ and $\mathscr{D}(V)$ is 0 elsewhere, as well as $M \Downarrow \emptyset$, where $\emptyset$ is the empty distribution. The distribution $[\![M]\!]$ assigns $\frac{1}{2}$ to I and $\frac{1}{4}$ to K.*

The semantics of terms satisfies some useful equations, such as:

Lemma 2.2. $[\![M \oplus N]\!] = \frac{1}{2}[\![M]\!] + \frac{1}{2}[\![N]\!]$.

We are interested in context equivalence in this probabilistic setting. Typically, in a qualitative scenario as the (non)deterministic one, terms are considered context equivalent if they both converge or diverge. Here, we need to take into account quantitative information.

```
expone f n = (f n) (+) (expone f n+1)
exptwo f = (\x -> f x) (+) (exptwo (\x -> f (x+1)))
expthree k f n = foldp k n f (expthree (expone id k) f)

foldp 0 n f g = g n
foldp m n f g = (f n) (+) (foldp (m-1) (n+1) f g)
```

Figure 2. Three Higher-Order Functions

Definition 2.3 (Context Preorder and Equivalence). *The expression $M{\Downarrow_p}$ stands for $\sum[\![M]\!] = p$, i.e., the term M converges with probability p. The* context preorder *$\leq_\oplus$ stipulates $M {\leq_\oplus} N$ if $C[M]{\Downarrow_p}$ implies $C[N]{\Downarrow_q}$ with $p \leq q$, for every closing context C. The equivalence induced by $\leq_\oplus$ is* probabilistic context equivalence, *denoted as $\simeq_\oplus$.*

Remark 2.4 (Types, Open Terms). *The results in this paper are stated for an untyped language. Adapting them to a simply-typed language is straightforward; we use integers, booleans and recursion in examples. Moreover, while the results are often stated for closed terms only, they can be generalized to open terms in the expected manner. In the paper, context equivalences and preorders are defined on open terms; (bi)similarities are defined on closed terms and it is then intended that they are extended to open terms by requiring the usual closure under substitutions.*

Example 2.5. *We give some basic examples of higher-order probabilistic programs, which we will analyse using the coinductive techniques we introduce later in this paper. Consider the functions* `expone`, `exptwo`, *and* `expthree` *from Figure 2. They are written in a* Haskell*-like language extended with probabilistic choice, but can also be seen as terms in a (typed) probabilistic λ-calculus with integers and recursion akin to $\Lambda_\oplus$. Term* `expone` *takes a function* `f` *and a natural number* `n` *in input, then it proceeds by tossing a fair coin (captured here by the binary infix operator* `(+)`*) and, depending on the outcome of the toss, either calls* `f` *on* `n`*, or recursively calls itself on* `f` *and* `n+1`*. When fed with, e.g., the identity and the natural number* `1`*, the program* `expone` *evaluates to the geometric distribution assigning probability $\frac{1}{2^n}$ to any positive natural number n. A similar effect can be obtained by* `exptwo`*, which only takes* `f` *in input, then "modifying" it along the evaluation. The function* `expthree` *is more complicated, at least apparently. To understand its behavior, one should first look at the auxiliary function* `foldp`*. If* `m` *and* `n` *are two natural numbers and* `f` *and* `g` *are two functions,* `foldp m n f g` *call-by-name reduces to the following expression:*

```
(f n) (+) ((f n+1) (+) ... ((f n+m-1) (+) (g n+m))).
```

The term `expthree` *works by forwarding its three arguments to* `foldp`*. The fourth argument is a recursive call to* `expthree` *where, however,* `k` *is replaced by any number greater or equal to it, chosen according to a geometric distribution. The functions above can all be expressed in $\Lambda_\oplus$, using fixed-point combinators. As we will see soon,* `expone`, `exptwo`, *and* `expthree k` *are context equivalent whenever* `k` *is a natural number.*

2.2 Probabilistic Bisimulation

In this section we recall the definition and a few basic notions of bisimulation for labelled Markov chains, following Larsen and Skou [26]. In Section 3 we will then adapt this form of bisimilarity to the probabilistic λ-calculus $\Lambda_\oplus$ by combining it with Abramsky's applicative bisimilarity.

Definition 2.6. *A* labelled Markov chain *is a triple $(\mathcal{S}, \mathcal{L}, \mathcal{P})$ such that:*

- *$\mathcal{S}$ is a countable set of* states*;*
- *$\mathcal{L}$ is set of* labels*;*

$$\frac{}{M \Downarrow \emptyset}\ \mathsf{bt} \qquad \frac{}{V \Downarrow \{V\}}\ \mathsf{bv} \qquad \frac{M \Downarrow \mathscr{D} \quad \{P\{^{N}/x\} \Downarrow \mathscr{E}_{P,N}\}_{\lambda x.P \in \mathsf{S}(\mathscr{D})}}{MN \Downarrow \sum_{\lambda x.P \in \mathsf{S}(\mathscr{D})} \mathscr{D}(\lambda x.P) \cdot \mathscr{E}_{P,N}}\ \mathsf{ba} \qquad \frac{M \Downarrow \mathscr{D} \quad N \Downarrow \mathscr{E}}{M \oplus N \Downarrow \frac{1}{2} \cdot \mathscr{D} + \frac{1}{2} \cdot \mathscr{E}}\ \mathsf{bs}$$

Figure 1. Big-step call-by-name approximation semantics for $\Lambda_\oplus$.

- *$\mathcal{P}$ is a* transition probability matrix*, i.e. a function*

  $$\mathcal{P} : \mathcal{S} \times \mathcal{L} \times \mathcal{S} \to \mathbb{R}_{[0,1]}$$

  such that the following normalization condition holds:

  $$\forall \ell \in \mathcal{L}. \forall s \in \mathcal{S}. \mathcal{P}(s, \ell, \mathcal{S}) \leq 1$$

  where, as usual $\mathcal{P}(s, \ell, X)$ stands for $\sum_{t \in X} \mathcal{P}(s, \ell, t)$ whenever $X \subseteq \mathcal{S}$.

If $\mathcal{R}$ is an equivalence relation on $\mathcal{S}$, $\mathcal{S}/\mathcal{R}$ denotes the *quotient* of $\mathcal{S}$ modulo $\mathcal{R}$, i.e., the set of all equivalence classes of $\mathcal{S}$ modulo $\mathcal{R}$. Given any binary relation $\mathcal{R}$, its reflexive and transitive closure is denoted as $\mathcal{R}^*$.

DEFINITION 2.7. *Given a labelled Markov chain $(\mathcal{S}, \mathcal{L}, \mathcal{P})$, a* probabilistic bisimulation *is an equivalence relation $\mathcal{R}$ on $\mathcal{S}$ such that $(s, t) \in \mathcal{R}$ implies that for every $\ell \in \mathcal{L}$ and for every $\mathtt{E} \in \mathcal{S}/\mathcal{R}$, $\mathcal{P}(s, \ell, \mathtt{E}) = \mathcal{P}(t, \ell, \mathtt{E})$.*

Note that a probabilistic bisimulation has to be, by definition, an *equivalence relation*. This means that, in principle, we are not allowed to define probabilistic bisimilarity simply as the union of all probabilistic bisimulations. As a matter of fact, given $\mathcal{R}, \mathcal{T}$ two equivalence relations, $\mathcal{R} \cup \mathcal{T}$ is not necessarily an equivalence relation. The following is a standard way to overcome the problem:

LEMMA 2.8. *If $\{\mathcal{R}_i\}_{i \in I}$, is a collection of probabilistic bisimulations, then also their reflexive and transitive closure $(\bigcup_{i \in I} \mathcal{R}_i)^*$ is a probabilistic bisimulation.*

Lemma 2.8 allows us to define the largest probabilistic bisimulation, called *probabilistic bisimilarity*. It is $\sim \overset{\text{def}}{=} \bigcup\{\mathcal{R} \mid \mathcal{R}$ is a probabilistic bisimulation$\}$. Indeed, by Lemma 2.8, $(\sim)^*$ is a probabilistic bisimulation too; we now claim that $\sim\, = (\sim)^*$. The inclusion $\sim\, \subseteq (\sim)^*$ is obvious. The other way around, $\sim \supseteq (\sim)^*$, follows by $(\sim)^*$ being a probabilistic bisimulation and hence included in $\sim$.

In the notion of a probabilistic simulation, preorders play the role of equivalence relations: given a labelled Markov chain $(\mathcal{S}, \mathcal{L}, \mathcal{P})$, a *probabilistic simulation* is a preorder relation $\mathcal{R}$ on $\mathcal{S}$ such that $(s, t) \in \mathcal{R}$ implies that for every $\ell \in \mathcal{L}$ and for every $X \subseteq \mathcal{S}$, $\mathcal{P}(s, \ell, X) \leq \mathcal{P}(t, \ell, \mathcal{R}(X))$, where as usual $\mathcal{R}(X)$ stands for the $\mathcal{R}$*-closure* of X, namely the set $\{y \in \mathcal{S} \mid \exists x \in X.\ x\ \mathcal{R}\ y\}$. Lemma 2.8 holds for probabilistic simulations, and as a consequence, we are allowed to define *similarity* simply as $\lesssim \overset{\text{def}}{=} \bigcup\{\mathcal{R} \mid \mathcal{R}$ is a probabilistic simulation$\}$.

Any symmetric probabilistic simulation is a probabilistic bisimulation. Contrary to the nondeterministic case, however, simulation equivalence coincides with bisimulation:

PROPOSITION 2.9. *$\sim$ coincides with $\lesssim \cap \lesssim^{op}$.*

For technical reasons that will become apparent soon, it is convenient to consider Markov chains in which the state space is partitioned into disjoint sets, in such a way that comparing states coming from different components is not possible. Remember that the disjoint union $\biguplus_{i \in I} X_i$ of a family of sets $\{X_i\}_{i \in I}$ is defined as $\{(a, i) \mid i \in I \wedge a \in X_i\}$. If the set of states $\mathcal{S}$ of a labelled Markov chain is a disjoint union $\biguplus_{i \in I} X_i$, one wants that (bi)simulation relations only compare elements coming from the same X_i, i.e. $(a, i)\mathcal{R}(b, j)$ implies $i = j$. In this case, we say that the underlying labelled Markov chain is *multisorted*.

3. Probabilistic Applicative Bisimulation and Howe's technique

In this section, notions of similarity and bisimilarity for $\Lambda_\oplus$ are introduced, in the spirit of Abramsky's work on applicative bisimulation [1]. Definitionally, this consists in seeing $\Lambda_\oplus$'s operational semantics as a labelled Markov chain, then giving the Larsen and Skou's notion of (bi)simulation for it. States will be terms, while labels will be of two kinds: one can either *evaluate* a term, obtaining (a distribution of) values, or *apply* a term to a value.

The resulting bisimulation (probabilistic applicative bisimulation) will be shown to be a congruence, thus included in probabilistic context equivalence. This will be done by a non-trivial generalization of Howe's technique [22], which is a well-known methodology to get congruence results in presence of higher-order functions, but which has not been applied to probabilistic calculi so far.

Formalizing probabilistic applicative bisimulation requires some care. As usual, two values $\lambda x.M$ and $\lambda x.N$ are defined to be bisimilar if for every L, $M\{^{L}/x\}$ and $N\{^{L}/x\}$ are themselves bisimilar. But how if we rather want to compare two arbitrary closed terms M and N? The simplest solution consists in following Larsen and Skou and stipulate that every equivalence class of $\mathsf{V}\Lambda_\oplus$ modulo bisimulation is attributed the same measure by both $[\![M]\!]$ and $[\![N]\!]$. Values are thus treated in two different ways (they are both terms and values), and this is the reason why each of them corresponds to *two* states in the underlying Markov chain.

DEFINITION 3.1. *$\Lambda_\oplus$ can be seen as a multisorted labelled Markov chain $(\Lambda_\oplus(\emptyset) \uplus \mathsf{V}\Lambda_\oplus, \Lambda_\oplus(\emptyset) \uplus \{\tau\}, \mathcal{P}_\oplus)$ that we denote with $\Lambda_\oplus$. Labels are either closed terms, which model parameter passing, or τ, that models evaluation. Please observe that the states of the labelled Markov chain we have just defined are elements of the disjoint union $\Lambda_\oplus(\emptyset) \uplus \mathsf{V}\Lambda_\oplus$. Two distinct states correspond to the same value V, and to avoid ambiguities, we call the second one (i.e. the one coming from $\mathsf{V}\Lambda_\oplus$) a* distinguished value*. When we want to insist on the fact that a value $\lambda x.M$ is distinguished, we indicate it with $\nu x.M$. We define the transition probability matrix $\mathcal{P}_\oplus$ as follows:*

- *For every term M and for every distinguished value $\nu x.N$,*

  $$\mathcal{P}_\oplus(M, \tau, \nu x.N) \overset{\text{def}}{=} [\![M]\!](\nu x.N);$$

- *For every term M and for every distinguished value $\nu x.N$,*

  $$\mathcal{P}_\oplus(\nu x.N, M, N\{^{M}/x\}) \overset{\text{def}}{=} 1;$$

- *In all other cases, $\mathcal{P}_\oplus$ returns 0.*

Terms seen as states only interact with the environment by performing τ, while distinguished values only take other closed terms as parameters.

Simulation and bisimulation relations can be defined for $\Lambda_\oplus$ as for any labelled Markov chain. Even if, strictly speaking, these are binary relations on $\Lambda_\oplus(\emptyset) \uplus \mathsf{V}\Lambda_\oplus$, we often see them just as their restrictions to $\Lambda_\oplus(\emptyset)$. Formally, a *probabilistic applicative bisimulation* (a PAB) is simply a probabilistic bisimulation on $\Lambda_\oplus$. This way one can define *probabilistic applicative bisimilarity*, which is denoted $\sim$. Similarly for *probabilistic applicative simulation* (PAS) and *probabilistic applicative similarity*, denoted $\lesssim$.

REMARK 3.2 (Early vs. Late). *Technically, the distinction between terms and values in Definition 3.1 means that our bisimulation is in* late *style. In bisimulations for value-passing concurrent languages, "late" indicates the explicit manipulation of functions in the clause for input actions: functions are chosen first, and only later, the input value received is taken into account [40]. Late-style is used in contraposition to* early *style, where the order of quantifiers is exchanged, so that the choice of functions may depend on the specific input value received. In our setting, adopting an early style would mean having transitions such as $\lambda x.M \xrightarrow{N} M\{^{N}/_{x}\}$, and then setting up a probabilistic bisimulation on top of the resulting transition system. We leave for future work a study of the comparison between the two styles. In this paper, we stick to the late style because easier to deal with, especially under Howe's technique. Previous works on applicative bisimulation for nondeterministic functions also focus on the late approach [30, 36].*

REMARK 3.3. *Defining applicative bisimulation in terms of multisorted labelled Markov chains has the advantage of recasting the definition in a familiar framework; most importantly, this formulation will be useful when dealing with Howe's method. To spell out the explicit operational details of the definition, a probabilistic applicative bisimulation can be seen as an equivalence relation $\mathcal{R} \subseteq \Lambda_\oplus(\emptyset) \times \Lambda_\oplus(\emptyset)$ such that whenever $M\ \mathcal{R}\ N$:*

1. $[\![M]\!](\mathtt{E} \cap \mathsf{V}\Lambda_\oplus) = [\![N]\!](\mathtt{E} \cap \mathsf{V}\Lambda_\oplus)$, *for any equivalence class* $\mathtt{E}$ *of* $\mathcal{R}$ *(that is, the probability of reaching a value in* $\mathtt{E}$ *is the same for the two terms);*
2. *if M and N are values, say $\lambda x.P$ and $\lambda x.Q$, then $P\{^{L}/_{x}\}\ \mathcal{R}\ Q\{^{L}/_{x}\}$, for all $L \in \Lambda_\oplus(\emptyset)$.*

The special treatment of values, in Clause 2., motivates the use of multisorted *labelled Markov chains in Definition 3.1.*

Terms with the same semantics are indistinguishable:

LEMMA 3.4. *The binary relation $\mathcal{R} = \{(M,N) \in \Lambda_\oplus(\emptyset) \times \Lambda_\oplus(\emptyset)$ s.t. $[\![M]\!] = [\![N]\!]\} \uplus \{(V,V) \in \mathsf{V}\Lambda_\oplus \times \mathsf{V}\Lambda_\oplus\}$ is a* PAB.

Conversely, knowing that two terms M and N are (bi)similar means knowing quite a lot about their convergence probability:

LEMMA 3.5 (Adequacy of Bisimulation). *If $M \sim N$, then $\sum[\![M]\!] = \sum[\![N]\!]$. Moreover, if $M \lesssim N$, then $\sum[\![M]\!] \leq \sum[\![N]\!]$.*

EXAMPLE 3.6. *The semantics of the terms:*

$$M \overset{\mathrm{def}}{=} ((\lambda x.(x \oplus x)) \oplus (\lambda x.x)) \oplus \Omega;$$
$$N \overset{\mathrm{def}}{=} \Omega \oplus (\lambda x.Ix);$$

differ, as for every value V, we have:

$$[\![M]\!](V) \quad = \quad \begin{cases} \frac{1}{4} & \textit{if } V \textit{ is } \lambda x.(x \oplus x) \textit{ or } \lambda x.x; \\ 0 & \textit{otherwise}; \end{cases}$$
$$[\![N]\!](V) \quad = \quad \begin{cases} \frac{1}{2} & \textit{if } V \textit{ is } \lambda x.Ix; \\ 0 & \textit{otherwise}. \end{cases}$$

Nonetheless, we can prove $M \sim N$. Indeed, $\nu x.(x \oplus x) \sim \nu x.x \sim \nu x.Ix$ because, for every $L \in \Lambda_\oplus(\emptyset)$, the three terms L, $L \oplus L$ and IL all have the same semantics, i.e., $[\![L]\!]$. Now, consider any equivalence class $\mathtt{E}$ of distinguished values modulo $\sim$. If $\mathtt{E}$ includes the three distinguished values above, then

$$\mathcal{P}_\oplus(M, \tau, \mathtt{E}) = \sum_{V \in \mathtt{E}} [\![M]\!](V) = \frac{1}{2} = \sum_{V \in \mathtt{E}} [\![N]\!](V) = \mathcal{P}_\oplus(N, \tau, \mathtt{E}).$$

Otherwise, $\mathcal{P}_\oplus(M, \tau, \mathtt{E}) = 0 = \mathcal{P}_\oplus(N, \tau, \mathtt{E})$.

3.1 Probabilistic Applicative Bisimulation is a Congruence

In this section, we prove that probabilistic applicative bisimulation is indeed a congruence, and that its non-symmetric sibling is a precongruence. The structure of the proof follows Howe [22], so we will not give all the details. The main idea consists in defining a way to turn an arbitrary relation $\mathcal{R}$ on (possibly open) terms to another one, $\mathcal{R}^H$, in such a way that, if $\mathcal{R}$ satisfies a few simple conditions, then $\mathcal{R}^H$ is a (pre)congruence including $\mathcal{R}$. The key step, then, is to prove that $\mathcal{R}^H$ is indeed a (bi)simulation. In view of Proposition 2.9, considering similarity suffices here.

It is here convenient to work with generalizations of relations called $\Lambda_\oplus$-*relations*, i.e. sets of triples in the form $(\overline{x}, M, N)$, where $M, N \in \Lambda_\oplus(\overline{x})$. Thus if a relation has the pair (M, N) with $M, N \in \Lambda_\oplus(\overline{x})$, then the corresponding $\Lambda_\oplus$-relation will include $(\overline{x}, M, N)$. (Recall that applicative (bi)similarity is extended to open terms by considering all closing substitutions.) Given any $\Lambda_\oplus$-relation $\mathcal{R}$, we write $\overline{x} \vdash M\ \mathcal{R}\ N$ if $(\overline{x}, M, N) \in \mathcal{R}$. A $\Lambda_\oplus$-relation $\mathcal{R}$ is said to be *compatible* iff the four conditions below hold:

($\mathsf{Com1}$) $\forall \overline{x} \in \mathcal{P}_{\mathsf{FIN}}(\mathsf{X}), x \in \overline{x}$: $\overline{x} \vdash x\ \mathcal{R}\ x$,

($\mathsf{Com2}$) $\forall \overline{x} \in \mathcal{P}_{\mathsf{FIN}}(\mathsf{X}), \forall x \in \mathsf{X} - \overline{x}, \forall M, N \in \Lambda_\oplus(\overline{x} \cup \{x\})$: $\overline{x} \cup \{x\} \vdash M\ \mathcal{R}\ N \Rightarrow \overline{x} \vdash \lambda x.M\ \mathcal{R}\ \lambda x.N$,

($\mathsf{Com3}$) $\forall \overline{x} \in \mathcal{P}_{\mathsf{FIN}}(\mathsf{X}), \forall M, N, L, P \in \Lambda_\oplus(\overline{x})$: $\overline{x} \vdash M\ \mathcal{R}\ N \wedge \overline{x} \vdash L\ \mathcal{R}\ P \Rightarrow \overline{x} \vdash ML\ \mathcal{R}\ NP$,

($\mathsf{Com4}$) $\forall \overline{x} \in \mathcal{P}_{\mathsf{FIN}}(\mathsf{X}), \forall M, N, L, P \in \Lambda_\oplus(\overline{x})$: $\overline{x} \vdash M\ \mathcal{R}\ N \wedge \overline{x} \vdash L\ \mathcal{R}\ P \Rightarrow \overline{x} \vdash M \oplus L\ \mathcal{R}\ N \oplus P$.

The notions of an equivalence relation and of a preorder can be straightforwardly generalized to $\Lambda_\oplus$-relations, and any compatible $\Lambda_\oplus$-relation that is an equivalence relation (respectively, a preorder) is said to be a *congruence* (respectively, a *precongruence*).

If bisimilarity is a congruence, then $C[M]$ is bisimilar to $C[N]$ whenever $M \sim N$ and C is a context. In other words, terms can be replaced by equivalent ones in any context. This is a crucial sanity-check any notion of equivalence is expected to pass.

It is well-known that proving bisimulation to be a congruence may be nontrivial when the underlying language contains higher-order functions. This is also the case here. Proving ($\mathsf{Com1}$), ($\mathsf{Com2}$) and ($\mathsf{Com4}$) just by inspecting the operational semantics of the involved terms is indeed possible, but the method fails for ($\mathsf{Com3}$), when the involved contexts contain applications. We thus need to introduce nontrivial technical tools.

Actually, the Howe's lifting of any $\Lambda_\oplus$-relation $\mathcal{R}$ is the relation $\mathcal{R}^H$ defined by the rules in Figure 3. The reader familiar with Howe's method should have a sense of *déjà vu* here: indeed, this is the same definition one finds in the realm of *nondeterministic* λ-calculi. The language of terms, after all, is the same. This facilitates the first part of the proof. Indeed, one already knows that if $\mathcal{R}$ is a preorder, then $\mathcal{R}^H$ is compatible, closed under term-substitution and includes $\mathcal{R}$, since all these properties are already known (see, e.g. [36]) and only depend on the shape of terms and not on their operational semantics. For the reader's convenience, all these proofs can be found in [10]. Something is missing, however, before we can conclude that $\lesssim^H$ is a precongruence, namely transitivity. We also follow Howe here and consider the transitive closure $(\lesssim^H)^+$, which is a preorder by construction.

This is just the first half of the story: we also need to prove that $(\lesssim^H)^+$ is a simulation. As we already know it is a preorder, the following lemma gives us the missing bit:

LEMMA 3.7 (Key Lemma). *If $M \lesssim^H N$, then for every $X \subseteq \Lambda_\oplus(x)$ it holds that $[\![M]\!](\lambda x.X) \leq [\![N]\!](\lambda x.(\lesssim^H(X)))$.*

The proof of this lemma is delicate and is discussed in the next section. From the lemma, using a standard argument we derive the needed substitutivity results, and ultimately the most important result of this section.

THEOREM 3.8. *On $\Lambda_\oplus$-terms, $\lesssim$ is a precongruence and $\sim$ is a congruence.*

$$\frac{\overline{x} \vdash x \mathrel{\mathcal{R}} M}{\overline{x} \vdash x \mathrel{\mathcal{R}^H} M}\ (\mathsf{How1}) \qquad \frac{\overline{x} \cup \{x\} \vdash M \mathrel{\mathcal{R}^H} L \qquad \overline{x} \vdash \lambda x.L \mathrel{\mathcal{R}} N \qquad x \notin \overline{x}}{\overline{x} \vdash \lambda x.M \mathrel{\mathcal{R}^H} N}\ (\mathsf{How2})$$

$$\frac{\overline{x} \vdash M \mathrel{\mathcal{R}^H} P \qquad \overline{x} \vdash N \mathrel{\mathcal{R}^H} Q \qquad \overline{x} \vdash PQ \mathrel{\mathcal{R}} L}{\overline{x} \vdash MN \mathrel{\mathcal{R}^H} L}\ (\mathsf{How3}) \qquad \frac{\overline{x} \vdash M \mathrel{\mathcal{R}^H} P \qquad \overline{x} \vdash N \mathrel{\mathcal{R}^H} Q \qquad \overline{x} \vdash P \oplus Q \mathrel{\mathcal{R}} L}{\overline{x} \vdash M \oplus N \mathrel{\mathcal{R}^H} L}\ (\mathsf{How4})$$

Figure 3. Howe's Lifting for $\Lambda_\oplus$.

3.2 Proof of the Key Lemma

Proving the Key Lemma 3.7 turns out to be much more difficult than for deterministic or nondeterministic cases. In particular, the case when M is an application relies on another technical lemma we are now going to give, which itself can be proved by tools from linear programming.

The combinatorial problem we will face while proving the Key Lemma can actually be decontextualized and understood independently. Suppose we have $n = 3$ non-disjoint sets X_1, X_2, X_3 whose elements are labelled with real numbers. As an example, we could be in a situation like the one in Figure 4(a) (where for the sake of simplicity only the labels are indicated). We fix three real numbers $p_1 \stackrel{\text{def}}{=} \frac{5}{64}, p_2 \stackrel{\text{def}}{=} \frac{3}{16}, p_3 \stackrel{\text{def}}{=} \frac{5}{64}$. It is routine to check that for every $I \subseteq \{1, 2, 3\}$ it holds that

$$\sum_{i \in I} p_i \leq || \bigcup_{i \in I} X_i ||,$$

where $||X||$ is the sum of the labels of the elements of X. Let us observe that it is of course possible to turn the three sets X_1, X_2, X_3 into three disjoint sets Y_1, Y_2 and Y_3 where each Y_i contains (copies of) the elements of X_i whose labels, however, are obtained by splitting the ones of the original elements. Examples of those sets are in Figure 4(b): if you superpose the three sets, you obtain the Venn diagram we started from. Quite remarkably, however, the examples from Figure 4 have an additional property, namely that for every $i \in \{1, 2, 3\}$ it holds that $p_i \leq ||Y_i||$. We now show that finding sets satisfying the properties above is always possible, even when n is arbitrary.

Suppose $p_1, \ldots, p_n \in \mathbb{R}_{[0,1]}$, and suppose that for each $I \subseteq \{1, \ldots, n\}$ a real number $r_I \in \mathbb{R}_{[0,1]}$ is defined such that for every such I it holds that $\sum_{i \in I} p_i \leq \sum_{J \cap I \neq \emptyset} r_J \leq 1$. Then $(\{p_i\}_{1 \leq i \leq n}, \{r_I\}_{I \subseteq \{1,\ldots,n\}})$ is said to be a *probability assignment* for $\{1, \ldots, n\}$. Is it always possible to "disentangle" probability assignments? The answer is positive.

LEMMA 3.9 (Disentangling Probability Assignments). *Let* $\mathbf{P} \stackrel{\text{def}}{=} (\{p_i\}_{1 \leq i \leq n}, \{r_I\}_{I \subseteq \{1,\ldots,n\}})$ *be a probability assignment. Then for every nonempty* $I \subseteq \{1, \ldots, n\}$ *and for every* $k \in I$ *there is* $s_{k,I} \in \mathbb{R}_{[0,1]}$ *such that the following conditions all hold:*

1. *for every* I*, it holds that* $\sum_{k \in I} s_{k,I} \leq 1$*;*
2. *for every* $k \in \{1, \ldots, n\}$*, it holds that* $p_k \leq \sum_{k \in I} s_{k,I} \cdot r_I$*.*

Proof. Any probability assignment $\mathbf{P}$ can be seen as a flow network, where nodes are the nonempty subsets of $\{1, \ldots, n\}$, plus a distinguished source s and target t. Edges, then, go from s to each singleton $\{i\}$ (with capacity p_i), from every nonempty I to $I \cup \{i\} \subseteq \{1, \ldots, n\}$ whenever $i \notin I$ (with capacity 1) and from every such I to t (with capacity r_I). The thesis, then, is easily proved equivalent to showing that such a net supports a flow of value $\sum p_i$. And, indeed, the fact that this is the value of the maximum flow from s to t can be proved via the Max-Flow Min-Cut Theorem. □

REMARK 3.10. *Throughout the following proof we will implicitly use a routine result stating that* $M \lesssim N$ *implies* $[\![M]\!](\lambda x.X) \leq [\![N]\!](\lambda x.{\lesssim}(X))$*, for every* $X \subseteq \Lambda_\oplus(x)$*. The property needed by the latter is precisely the reason why we have formulated* $\Lambda_\oplus$ *as a multisorted labelled Markov chain:* ${\lesssim}(\nu x.X)$ *consists of distinguished values only, and is nothing but* $\nu x.{\lesssim}(X)$*.*

Proof. [of Lemma 3.7] This is equivalent to proving that if $M \lesssim^H N$, then for every $X \subseteq \Lambda_\oplus(x)$ the following implication holds: if $M \Downarrow \mathscr{D}$, then $\mathscr{D}(\lambda x.X) \leq [\![N]\!](\lambda x.({\lesssim^H}(X)))$. This is an induction on the structure of the proof of $M \Downarrow \mathscr{D}$. Some interesting cases follow:

- If M is a value $\lambda x.L$ and $\mathscr{D}(\lambda x.L) = 1$, then the proof of $M \lesssim^H N$ necessarily ends as follows:

$$\frac{\{x\} \vdash L \lesssim^H P \qquad \emptyset \vdash \lambda x.P \lesssim N}{\emptyset \vdash \lambda x.L \lesssim^H N}$$

Let X be any subset of $\Lambda_\oplus(x)$. Now, if $L \notin X$, then $\mathscr{D}(\lambda x.X) = 0$ and the inequality trivially holds. If, on the contrary, $L \in X$, then $P \in {\lesssim^H}(X)$. Consider $\lesssim(P)$, the set of terms that are in relation with P via $\lesssim$. We have that for every $Q \in {\lesssim}(P)$, both $\{x\} \vdash L \lesssim^H P$ and $\{x\} \vdash P \lesssim Q$ hold, and as a consequence $\{x\} \vdash L \lesssim^H Q$ does (this is a consequence of a property of $(\cdot)^H$, see [10]). In other words, ${\lesssim}(P) \subseteq {\lesssim^H}(X)$. But then,

$$[\![N]\!](\lambda x.{\lesssim^H}(X)) \geq [\![N]\!](\lambda x. \lesssim(P)) \geq [\![\lambda x.P]\!](\lambda x.P) = 1.$$

- If M is an application LP, then $M \Downarrow \mathscr{D}$ is obtained as follows:

$$\frac{L \Downarrow \mathscr{F} \qquad \{Q\{P/x\} \Downarrow \mathscr{H}_{Q,P}\}_{Q,P}}{LP \Downarrow \sum_Q \mathscr{F}(\lambda x.Q) \cdot \mathscr{H}_{Q,P}}$$

Moreover, the proof of $\emptyset \vdash M \lesssim^H N$ must end as follows:

$$\frac{\emptyset \vdash L \lesssim^H R \qquad \emptyset \vdash P \lesssim^H S \qquad \emptyset \vdash RS \lesssim N}{\emptyset \vdash LP \lesssim^H N}$$

Now, since $L \Downarrow \mathscr{F}$ and $\emptyset \vdash L \lesssim^H R$, by induction hypothesis we get that for every $Y \subseteq \Lambda_\oplus(x)$ it holds that $\mathscr{F}(\lambda x.Y) \leq [\![R]\!](\lambda x.{\lesssim^H}(Y))$. Let us now take a look at the distribution

$$\mathscr{D} = \sum_Q \mathscr{F}(\lambda x.Q) \cdot \mathscr{H}_{Q,P}.$$

Since $\mathscr{F}$ is a *finite* distribution, the sum above is actually the sum of finitely many summands. Let the support $\mathsf{S}(\mathscr{F})$ of $\mathscr{F}$ be $\{\lambda x.Q_1, \ldots, \lambda x.Q_n\}$. It is now time to put the above into a form that is amenable to treatment by Lemma 3.9. Let us consider the n sets ${\lesssim^H}(Q_1), \ldots, {\lesssim^H}(Q_n)$; to each term U in them we can associate the probability $[\![R]\!](\lambda x.U)$. We are then in the scope of Lemma 3.9, since by induction hypothesis we know that for every $Y \subseteq \Lambda_\oplus(x)$, $\mathscr{F}(\lambda x.Y) \leq [\![R]\!](\lambda x.{\lesssim^H}(Y))$. We can then conclude that for every

$$U \in {\lesssim^H}(\{Q_1, \ldots, Q_n\}) = \bigcup_{1 \leq i \leq n} {\lesssim^H}(Q_i)$$

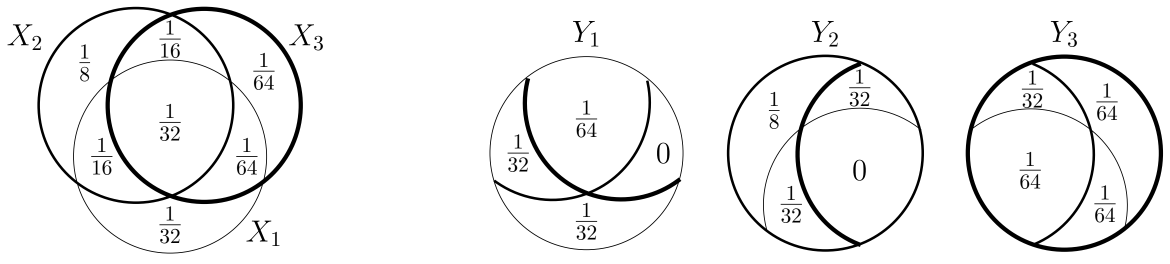


Figure 4. Disentangling Sets

there are n real numbers $r_1^{U,R}, \dots, r_n^{U,R}$ such that:

$$\llbracket R \rrbracket(\lambda x.U) \geq \sum_{1 \leq i \leq n} r_i^{U,R} \qquad \forall U \in \bigcup_{1 \leq i \leq n} \lesssim^H(Q_i);$$

$$\mathscr{F}(\lambda x.Q_i) \leq \sum_{U \in \lesssim^H(Q_i)} r_i^{U,R} \qquad \forall 1 \leq i \leq n.$$

So, we can conclude that

$$\mathscr{D} \leq \sum_{1 \leq i \leq n} \left(\sum_{U \in \lesssim^H(Q_i)} r_i^{U,R} \right) \cdot \mathscr{H}_{Q_i,P}$$
$$= \sum_{1 \leq i \leq n} \sum_{U \in \lesssim^H(Q_i)} r_i^{U,R} \cdot \mathscr{H}_{Q_i,P}.$$

Now, whenever $Q_i \lesssim^H U$ and $P \lesssim^H S$, it follows that $Q_i\{P/x\} \lesssim^H U\{S/x\}$. We can then apply the inductive hypothesis to the n derivations of $Q_i\{P/x\} \Downarrow \mathscr{H}_{Q_i,P}$, obtaining that, for every $X \subseteq \Lambda_\oplus(x)$,

$$\begin{aligned}
\mathscr{D}(\lambda x.X) &\leq \sum_{1 \leq i \leq n} \sum_{U \in \lesssim^H(Q_i)} r_i^{U,R} \cdot \llbracket U\{S/x\} \rrbracket(\lambda x. \lesssim^H(X)) \\
&\leq \sum_{1 \leq i \leq n} \sum_{U \in \lesssim^H(\{Q_1,\dots,Q_n\})} r_i^{U,R} \cdot \llbracket U\{S/x\} \rrbracket(\lambda x. \lesssim^H(X)) \\
&= \sum_{U \in \lesssim^H(\{Q_1,\dots,Q_n\})} \sum_{1 \leq i \leq n} r_i^{U,R} \cdot \llbracket U\{S/x\} \rrbracket(\lambda x. \lesssim^H(X)) \\
&= \sum_{U \in \lesssim^H(\{Q_1,\dots,Q_n\})} \left(\sum_{1 \leq i \leq n} r_i^{U,R} \right) \cdot \llbracket U\{S/x\} \rrbracket(\lambda x. \lesssim^H(X)) \\
&\leq \sum_{U \in \lesssim^H(\{Q_1,\dots,Q_n\})} \llbracket R \rrbracket(\lambda x.U) \cdot \llbracket U\{S/x\} \rrbracket(\lambda x. \lesssim^H(X)) \\
&\leq \sum_{U \in \Lambda_\oplus(x)} \llbracket R \rrbracket(\lambda x.U) \cdot \llbracket U\{S/x\} \rrbracket(\lambda x. \lesssim^H(X)) \\
&= \llbracket RS \rrbracket(\lambda x. \lesssim^H(X)) \leq \llbracket N \rrbracket(\lambda x. \lesssim ((\lesssim^H)(X))) \\
&\leq \llbracket N \rrbracket(\lambda x. \lesssim^H(X)),
\end{aligned}$$

which is the thesis.

This concludes the proof. □

3.3 Relating Applicative Bisimulation and Context Equivalence

The congruence of applicative bisimilarity yields the inclusion in context equivalence.

THEOREM 3.11. *For all M, $N \in \Lambda_\oplus$, $M \sim N$ implies $M \simeq_\oplus N$.*

The converse inclusion fails. A counterexample is described in the following.

EXAMPLE 3.12. *For $M \stackrel{\text{def}}{=} \lambda x.L \oplus P$ and $N \stackrel{\text{def}}{=} (\lambda x.L) \oplus (\lambda x.P)$ (where L is $\lambda y.\Omega$ and P is $\lambda y.\lambda z.\Omega$), we have $M \not\lesssim N$, hence $M \not\sim N$, but $M \simeq_\oplus N$.*

We prove that the above two terms are context equivalent by means of *CIU-equivalence*. This is a relation that can be shown to coincide with context equivalence by a Context Lemma, itself proved by the Howe's technique. More details are in [10]. See also Section 7.

EXAMPLE 3.13. *We consider again the programs from Example 2.5. Terms* `expone` *and* `exptwo` *only differ because the former performs all probabilistic choices on natural numbers obtained by applying a function to its argument, while in the latter choices are done at the functional level, and the argument to those functions is provided only at a later stage. As a consequence, the two terms are not applicative bisimilar, and the reason is akin to that for the inequality of the terms in Example 3.12. In contrast, the bisimilarity between* `expone` *and* `expthree k`*, where* `k` *is any natural number, intuitively holds because both* `expone` *and* `expthree k` *evaluate to a single term when fed with a function, while they start evolving in a genuinely probabilistic way only after the second argument is provided. At that point, the two functions evolve in very different ways, but their semantics (in the sense of Section 2) is the same (cf., Lemma 3.4). As a bisimulation one can use the equivalence generated by the relation*

$$\left(\bigcup_{\mathtt{k}} \{(\mathtt{expone}, \mathtt{expthree\ k})\} \right) \cup \{(M, N) \mid \llbracket M \rrbracket = \llbracket N \rrbracket\}$$
$$\cup \left(\bigcup_{L} \{(\lambda \mathtt{n}.\mathtt{B}\{L/\mathtt{f}\}, \lambda \mathtt{n}.\mathtt{C}\{L/\mathtt{f}\})\} \right)$$

using `B` *and* `C` *for the body of* `expone` *and* `expthree` *respectively.*

4. The Discriminating Power of Probabilistic Contexts

We show here that applicative bisimilarity and context equivalence collapse if the tested terms are pure, *deterministic*, λ-terms. In other words, if the probabilistic choices are brought into the terms only through the inputs supplied to the tested functions, applicative bisimilarity and context equivalence yield exactly the same discriminating power. To show this, we prove that, on pure λ-terms, both relations coincide with the *Levy-Longo tree equality*, which equates terms with the same Levy-Longo tree (briefly LLT) [14].

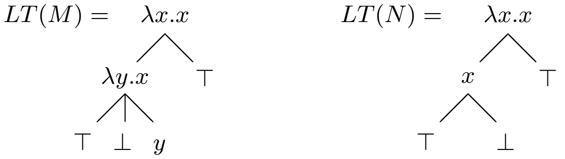
LLT's are the lazy variant of Böhm Trees (briefly BT), the most popular tree structure in the λ-calculus. BT's only correctly express the computational content of λ-terms in a *strong* regime, while they fail to do so in the lazy one. For instance, the term $\lambda x.\Omega$ and Ω, as both *unsolvable* [4], have identical BT's, but in a lazy regime we would always distinguish between them; hence they have different LLT's. The *Levy-Longo tree* of M, $LT(M)$, is coinductively constructed as follows: $LT(M) \stackrel{\text{def}}{=} \lambda x_1 \dots x_n.\bot$

if M is an unsolvable of order n; $LT(M) \overset{\text{def}}{=} \top$ if M is an unsolvable of order ∞; finally if M has principal head normal form $\lambda x_1 .\dots x_n .yM_1 \dots M_m$, then $LT(M)$ is a tree with root $\lambda x_1 .\dots x_n .y$ and with $LT(M_1), \dots , LT(M_m)$ as subtrees. Being defined coinductively, LLT's can of course be infinite. We write $M =_{\mathrm{LL}} N$ iff $LT(M) = LT(N)$.

EXAMPLE 4.1. *Let Ξ be an unsolvable of order ∞ such as $\Xi \overset{\text{def}}{=} (\lambda x.\lambda y.(xx))(\lambda x.\lambda y.(xx))$, and consider the terms*

$$M \overset{\text{def}}{=} \lambda x.(x(\lambda y.(x\Xi\Omega y))\Xi); \qquad N \overset{\text{def}}{=} \lambda x.(x(x\Xi\Omega)\Xi).$$

These terms have been used to prove non-full-abstraction results in a canonical model for the lazy λ-calculus by Abramsky and Ong [2]. For this, they show that in the model the convergence test is definable (this operator, when it receives an argument, would return the identity function if the supplied argument is convergent, and would diverge otherwise). The convergence test, ∇, can distinguish between the two terms, as $M\nabla$ reduces to an abstraction, whereas $N\nabla$ diverges. However, no pure λ-term can make the same distinction. The two terms also have different LL trees:



Although in $\Lambda_\oplus$, as in Λ, the convergence test operator is not definable, M and N can be separated using probabilities by running them in a context C that would feed $\Omega \oplus \lambda z.\lambda u.z$ as argument; then $C[M]\Downarrow_{\frac{1}{2}}$ whereas $C[N]\Downarrow_{\frac{1}{4}}$.

EXAMPLE 4.2. *Abramsky's canonical model is itself coarser than LLT equality. For instance, the terms $M \overset{\text{def}}{=} \lambda x.xx$ and $N \overset{\text{def}}{=} \lambda x.(x\lambda y.(xy))$, have different LLT's but are equal in Abramsky's model (and hence equal for context equivalence in Λ). They are separated by context equivalence in $\Lambda_\oplus$, for instance using the context $C \overset{\text{def}}{=} [\cdot](I \oplus \Omega)$, since $C[M]\Downarrow_{\frac{1}{4}}$ whereas $C[N]\Downarrow_{\frac{1}{2}}$.*

We already know that on full $\Lambda_\oplus$, applicative bisimilarity ($\sim$) implies context equivalence ($\simeq_\oplus$). Hence, to prove that *on pure λ-terms* the two equivalences collapse to LLT equality ($=_{\mathrm{LL}}$), it suffices to prove that, for those pure terms, $\simeq_\oplus$ implies $=_{\mathrm{LL}}$, and that $=_{\mathrm{LL}}$ implies $\sim$.

The first implication is obtained by a variation on the Böhm-out technique, a powerful methodology for separation results in the λ-calculus, often employed in proofs about local structure characterisation theorems of λ-models. For this we exploit an inductive characterisation of LLT equality via stratification approximants. The n-approximant (for $n \geq 1$), written $=^n_{\mathrm{LL}}$, is essentially obtained by observing a tree only up-to level n. The key Lemma 4.3 shows that any difference on the trees of two λ-terms within level n can be observed by a suitable context of the probabilistic λ-calculus.

We write $\uplus M$ as an abbreviation for the term $\Omega \oplus M$. We denote by Q_n, $n > 0$, the term $\lambda x_1 .\dots\lambda x_n .x_n x_1 x_2 \cdots x_{n-1}$. This is usually called the *Böhm permutator* of degree n. Böhm permutators play a key role in the Böhm-out technique. A variant of them, the $\uplus$-permutators, play a pivotal role in Lemma 4.3 below. A term $M \in \Lambda_\oplus$ is a *$\uplus$-permutator of degree n* if either $M = Q_n$ or there exists $0 \leq r < n$ such that

$$M = \lambda x_1 .\dots\lambda x_r . \uplus \lambda x_{r+1} \cdots \lambda x_n .x_n x_1 \cdots x_{n-1} .$$

A function f from the positive integers to λ-terms is a *$\uplus$-permutator function* if, for all n, $f(n)$ is a $\uplus$-permutator of degree n.

LEMMA 4.3. *Suppose $M \neq^n_{\mathrm{LL}} N$ for some n, and let $\{x_1, \dots , x_r\}$ be the free variables in M, N. Then there are integers $m_{x_1}, \dots , m_{x_r}$ and k, and permutator functions $f_{x_1}, \dots , f_{x_r}$ such that, for all $m > k$, there are closed terms $\overline{R_m}$ such that the following holds: if $M\{f_{x_1}(m + m_{x_1})/x_1\} \dots \{f_{x_r}(m + m_{x_r})/x_r\}\overline{R_m}\Downarrow_r$ and $N\{f_{x_1}(m + m_{x_1})/x_1\} \dots \{f_{x_r}(m + m_{x_r})/x_r\}\overline{R_m}\Downarrow_s$, then $r \neq s$.*

The proof goes by induction on the least n such that $M \neq^n_{\mathrm{LL}} N$.

The fact the Böhm-out technique actually works implies that the discriminating power of probabilistic contexts is at least as strong as the one of LLT's.

COROLLARY 4.4. *For $M, N \in \Lambda$, $M \simeq_\oplus N$ implies $M =_{\mathrm{LL}} N$.*

To show that LLT equality is included in probabilistic applicative bisimilarity, we proceed as follows. First we define a refinement of the latter, essentially one in which we *observe* all probabilistic choices. As a consequence, the underlying bisimulation game may ignore probabilities. The obtained notion of equivalence is strictly finer than probabilistic applicative bisimilarity. The advantage of the refinement is that *both* the inclusion of LLT equality in the refinement, and the inclusion of the latter in probabilistic applicative bisimilarity turn out to be relatively easy to prove. A *direct* proof of the inclusion of LLT equality in probabilistic applicative bisimilarity would have been harder, as it would have required extending the notion of a Levy-Longo tree to $\Lambda_\oplus$, then reasoning on substitution closures of such trees. The definition below relies on two notions of reduction: $M \longrightarrow_p N$ means that M call-by-name reduces to N in one step with probability p. (As a matter of fact, p can be either 1 or $\frac{1}{2}$.) Then $\Longrightarrow$ is obtained by composing $\longrightarrow$ zero or more times (and multiplying the corresponding real numbers).

DEFINITION 4.5. *A relation $\mathcal{R} \subseteq \Lambda_\oplus(\emptyset) \times \Lambda_\oplus(\emptyset)$ is a* strict applicative bisimulation *whenever $M \mathrel{\mathcal{R}} N$ implies*

1. *if $M \longrightarrow_1 P$, then $N \Longrightarrow_1 Q$ and $P \mathrel{\mathcal{R}} Q$;*
2. *if $M \longrightarrow_{\frac{1}{2}} P$, then $N \Longrightarrow_{\frac{1}{2}} Q$ and $P \mathrel{\mathcal{R}} Q$;*
3. *if $M = \lambda x.P$, then $N \Longrightarrow_1 \lambda x.Q$ and $P\{L/x\} \mathrel{\mathcal{R}} Q\{L/x\}$ for all $L \in \Lambda_\oplus(\emptyset)$;*
4. *the converse of 1., 2. and 3..*

Strict applicative bisimilarity *is the union of all strict applicative bisimulations.*

If two terms have the same LLT, then passing them the same argument $M \in \Lambda_\oplus$ produces *exactly* the same choice structure: intuitively, whenever the first term finds (a copy of) M in head position, also the second will find M.

LEMMA 4.6. *If $M =_{\mathrm{LL}} N$ then $M \mathrel{\mathcal{R}} N$, for some strict applicative bisimulation $\mathcal{R}$.*

Terms which are strict applicative bisimilar cannot be distinguished by applicative bisimilarity proper, since the requirements induced by the latter are less strict than the ones the former imposes:

LEMMA 4.7. *Strict applicative bisimilarity is included in applicative bisimilarity.*

Since we now know that for *pure, deterministic* λ-terms, $=_{\mathrm{LL}}$ is included in $\sim$ (by Lemma 4.6 and Lemma 4.7), that $\sim$ is included in $\simeq_\oplus$ (by Theorem 3.11) and that the latter is included in $=_{\mathrm{LL}}$ (Corollary 4.4), we can conclude:

COROLLARY 4.8. *The relations $=_{\mathrm{LL}}$, $\sim$, and $\simeq_\oplus$ coincide in Λ.*

5. Coupled Logical Bisimulation

In this section we derive a coinductive characterisation of probabilistic context equivalence on the whole language $\Lambda_\oplus$ (as opposed to the subset of sum-free λ-terms as in Section 4). For this, we need to

$$\frac{}{M \oplus N \rightsquigarrow \langle M, \frac{1}{2}\rangle + \langle N, \frac{1}{2}\rangle}\ \textsf{ss}$$

$$\frac{}{\lambda x.M \rightsquigarrow \langle \lambda x.M, 1\rangle}\ \textsf{sl} \qquad \frac{[\![M_i]\!] = \mathscr{D}_i}{\Sigma_i\langle M_i, p_i\rangle \rightsquigarrow \Sigma_i\langle \mathscr{D}_i, p_i\rangle}\ \textsf{spc}$$

$$\frac{}{ZM \rightsquigarrow Z \bullet M}\ \textsf{sp} \qquad \frac{E \rightsquigarrow F}{EM \rightsquigarrow FM}\ \textsf{sa}$$

Figure 5. Reduction Rules for $\Lambda_\oplus^{\mathrm{FS}}$

manipulate formal weighted sums. Thus we work with an extension of $\Lambda_\oplus$ in which such weighted sums may appear in redex position. An advantage of having formal sums is that the transition system on the extended language can be small-step and deterministic — any closed term that is not a value will have exactly one possible internal transition.

This will make it possible to pursue the *logical bisimulation* method, in which the congruence of bisimilarity is proved using a standard induction argument over all contexts. The refinement of the method handling probabilities, called *coupled* logical bisimulation, uses *pairs* of relations, as we need to distinguish between ordinary terms and terms possibly containing formal sums.

We preferred to follow logical bisimulations rather then environmental bisimulations because the former admit a simpler definition (in the latter, each pair of terms is enriched with an environment, that is, an extra set of pairs of terms). Moreover it is unclear what environments should be when one also considers formal sums. We leave this for future work.

Formal sums are a tool for representing the behaviour of running $\Lambda_\oplus$ terms. Thus, on terms with formal sums, only the results for closed terms interest us. However, the characterization of contextual equivalence in $\Lambda_\oplus$ as coupled logical bisimulation also holds on open terms.

5.1 Notation and Terminology

We write $\Lambda_\oplus^{\mathrm{FS}}$ for the extension of $\Lambda_\oplus$ in which formal sums may appear in redex position. Terms of $\Lambda_\oplus^{\mathrm{FS}}$ are defined as follows (M, N being $\Lambda_\oplus$-terms):

$$E, F ::= EM \;\Big|\; \Sigma_{i\in I}\langle M_i, p_i\rangle \;\Big|\; M \oplus N \;\Big|\; \lambda x.M.$$

In a formal sum $\Sigma_{i\in I}\langle M_i, p_i\rangle$, I is a countable (possibly empty) set of indices such that $\sum_{i\in I} p_i \leq 1$. We use $+$ for binary formal sums. Formal sums are ranged over by metavariables like H, K. When each M_i is a value (i.e., an abstraction) then $\Sigma_{i\in I}\langle M_i, p_i\rangle$ is a (*formally summed*) *value*; such values are ranged over by Z, Y, X. If $H = \Sigma_{i\in I}\langle M_i, p_i\rangle$ and $K = \Sigma_{j\in J}\langle M_j, p_j\rangle$ where I and J are disjoint, then $H \oplus K$ abbreviates $\Sigma_{r\in I\cup J}\langle M_r, \frac{p_r}{2}\rangle$. Similarly, if for every $j \in J$ H_j is $\Sigma_{i\in I}\langle M_{i,j}, p_{i,j}\rangle$, then $\Sigma_j\langle H_j, p_j\rangle$ stands for $\Sigma_{(i,j)}\langle M_{i,j}, p_{i,j} \cdot p_j\rangle$. For $H = \Sigma_i\langle M_i, p_i\rangle$ we write $\mathbf{\Sigma}(H)$ for the real number $\sum_i p_i$. If $Z = \Sigma_i\langle \lambda x.M_i, p_i\rangle$, then $Z \bullet N$ stands for $\Sigma_i\langle M_i\{^N/x\}, p_i\rangle$. The set of closed terms is $\Lambda_\oplus^{\mathrm{FS}}(\emptyset)$.

Any partial value distribution $\mathscr{D}$ (in the sense of Section 2) can be seen as the formal sum $\Sigma_{V\in \mathsf{V}\Lambda_\oplus}\langle V, \mathscr{D}(V)\rangle$. Similarly, any formal sum $H = \Sigma_{i\in I}\langle M_i, p_i\rangle$ can be mapped to the distribution $\sum_{i\in I} p_i \cdot [\![M_i]\!]$, that we indicate with $[\![H]\!]$.

Reduction between $\Lambda_\oplus^{\mathrm{FS}}$ terms, written $E \rightsquigarrow F$, is defined by the rules in Figure 5; these rules are given on top of the operational semantics for $\Lambda_\oplus$ as defined in Section 2, which is invoked in the premise of rule $\textsf{spc}$ (if there is a i with M_i not a value). The reduction relation $\rightsquigarrow$ is deterministic and strongly normalizing. We use $\overset{\rightsquigarrow}{\rightsquigarrow}$ for its reflexive and transitive closure. Lemma 5.1 shows the agreement between the new reduction relation and the original one.

LEMMA 5.1. *For all $M \in \Lambda_\oplus(\emptyset)$ there is a value Z such that $M \overset{\rightsquigarrow}{\rightsquigarrow} Z$ and $[\![M]\!] = [\![Z]\!]$.*

Proof. One first show that for all E there is n such that $E \rightsquigarrow^n Z$. Then one reasons with a double induction: an induction on n, and a transition induction, exploiting the determinism of $\rightsquigarrow$. □

5.2 Context Equivalence and Bisimulation

In $\Lambda_\oplus^{\mathrm{FS}}$ certain terms (i.e., formal sums) may only appear in redex position; ordinary terms (i.e., terms in $\Lambda_\oplus$), by contrast, may appear in arbitrary position. When extending context equivalence to $\Lambda_\oplus^{\mathrm{FS}}$ we therefore have to distinguish these two cases. Moreover, as our main objective is the characterisation of context equivalence in $\Lambda_\oplus$, we set a somewhat constrained context equivalence in $\Lambda_\oplus^{\mathrm{FS}}$ in which contexts may not contain formal sums (thus the $\Lambda_\oplus^{\mathrm{FS}}$ contexts are the same as the $\Lambda_\oplus$ contexts). We call these *simple* $\Lambda_\oplus^{\mathrm{FS}}$ *contexts*, whereas we call *general* $\Lambda_\oplus^{\mathrm{FS}}$ *context* an unconstrained context, i.e., a $\Lambda_\oplus^{\mathrm{FS}}$ term in which the hole $[\cdot]$ may appear in any places where a term from $\Lambda_\oplus$ was expected — including within a formal sum. (Later we will see that allowing general contexts does not affect the resulting context equivalence.) Terms possibly containing formal sums are tested in evaluation contexts, i.e., contexts of the form $[\cdot]\overline{M}$. We write $E{\downdownarrows_p}$ if $E \overset{\rightsquigarrow}{\rightsquigarrow} Z$ and $\mathbf{\Sigma}(Z) = p$ (recall that Z is unique, for a given E).

DEFINITION 5.2 (Context Equivalence in $\Lambda_\oplus^{\mathrm{FS}}$). *Two $\Lambda_\oplus$-terms M and N are* context equivalent in $\Lambda_\oplus^{\mathrm{FS}}$, *written $M \simeq_\oplus^{\mathrm{FS}} N$, if for all (closing) simple $\Lambda_\oplus^{\mathrm{FS}}$ contexts C, we have $C[M]{\downdownarrows_p}$ iff $C[N]{\downdownarrows_p}$. Two $\Lambda_\oplus^{\mathrm{FS}}$-terms E and F are* evaluation-context equivalent, *written $E \approxeq_\oplus^{\mathrm{FS}} F$, if for all (closing) $\Lambda_\oplus^{\mathrm{FS}}$ evaluation contexts C, we have $C[E]{\downdownarrows_p}$ iff $C[F]{\downdownarrows_p}$.*

In virtue of Lemma 5.1, context equivalence in $\Lambda_\oplus$ coincides with context equivalence in $\Lambda_\oplus^{\mathrm{FS}}$.

We now introduce a bisimulation that yields a coinductive characterisation of context equivalence (and also of evaluation-context equivalence). A *coupled relation* is a pair $(\mathcal{V}, \mathcal{E})$ where: $\mathcal{V} \subseteq \Lambda_\oplus(\emptyset) \times \Lambda_\oplus(\emptyset)$, $\mathcal{E} \subseteq \Lambda_\oplus^{\mathrm{FS}}(\emptyset) \times \Lambda_\oplus^{\mathrm{FS}}(\emptyset)$, and $\mathcal{V} \subseteq \mathcal{E}$. Intuitively, we place in $\mathcal{V}$ the pairs of terms that should be preserved by all contexts, and in $\mathcal{E}$ those that should be preserved by evaluation contexts. For a coupled relation $\mathcal{R} = (\mathcal{V}, \mathcal{E})$ we write $\mathcal{R}_1$ for $\mathcal{V}$ and $\mathcal{R}_2$ for $\mathcal{E}$. The union of coupled relations is defined componentwise: e.g., if $\mathcal{R}$ and $\mathcal{S}$ are coupled relations, then the coupled relation $\mathcal{R} \cup \mathcal{S}$ has $(\mathcal{R} \cup \mathcal{S})_1 \overset{\mathrm{def}}{=} \mathcal{R}_1 \cup \mathcal{S}_1$ and $(\mathcal{R} \cup \mathcal{S})_2 \overset{\mathrm{def}}{=} \mathcal{R}_2 \cup \mathcal{S}_2$. If $\mathcal{V}$ is a relation on $\Lambda_\oplus$, then $\mathcal{V}^{\mathrm{C}}$ is the context closure of $\mathcal{V}$ in $\Lambda_\oplus$, i.e., the set of all (closed) terms of the form $(C[\overline{M}], C[\overline{N}])$ where C is a multi-hole $\Lambda_\oplus$ context and $\overline{M}\ \mathcal{V}\ \overline{N}$.

DEFINITION 5.3. *A coupled relation $\mathcal{R}$ is a* coupled logical bisimulation *if whenever $E\ \mathcal{R}_2\ F$ we have:*

1. if $E \rightsquigarrow D$, then $F \overset{\rightsquigarrow}{\rightsquigarrow} G$, where $D\ \mathcal{R}_2\ G$;

2. if E is a formally summed value, then $F \overset{\rightsquigarrow}{\rightsquigarrow} Y$ with $\mathbf{\Sigma}(E) = \mathbf{\Sigma}(Y)$, and for all $M\ \mathcal{R}_1^{\mathrm{C}}\ N$ we have $(E \bullet M)\ \mathcal{R}_2\ (Y \bullet N)$;

3. the converse of 1. and 2..

Coupled logical bisimilarity, $\approx$, *is the union of all coupled logical bisimulations (hence $\approx_1$ is the union of the first component of all coupled logical bisimulations, and similarly for $\approx_2$).*

In a coupled bisimulation $(\mathcal{R}_1, \mathcal{R}_2)$, the bisimulation game is only played on the pairs in $\mathcal{R}_2$. However, the first relation $\mathcal{R}_1$ is relevant, as inputs for tested functions are built using $\mathcal{R}_1$ (Clause 2. of Definition 5.3). Actually, also the pairs in $\mathcal{R}_1$ are tested, because in any coupled relations it must be $\mathcal{R}_1 \subseteq \mathcal{R}_2$. The values produced by the bisimulation game for coupled bisimulation on $\mathcal{R}_2$ are formal sums (not plain λ-terms), and this is why we do not require them to be in $\mathcal{R}_1$: formal sums should only appear in redex position, but terms in $\mathcal{R}_1$ can be used as arguments to bisimilar functions and can therefore end up in arbitrary positions.

We will see below another aspect of the relevance of $\mathcal{R}_1$: the proof technique of logical bisimulation only allows us to prove substitutivity of the bisimilarity in arbitrary contexts *for the pairs of terms* in $\mathcal{R}_1$. For pairs in $\mathcal{R}_2$ but not in $\mathcal{R}_1$ the proof technique only allows us to derive preservation in evaluation contexts.

In the proof of congruence of coupled logical bisimilarity we will push "as many terms as possible" into the first relation, i.e., the first relation will be as large as possible. However, in proofs of bisimilarity for concrete terms, the first relation may be very small, possibly a singleton or even empty. Then the bisimulation clauses become similar to those of applicative bisimulation (as inputs of tested function are "almost" identical). Summing up, in coupled logical bisimulation the use of two relations gives us more flexibility than in ordinary logical bisimulation: depending on the needs, we can tune the size of the first relation. It is possible that some of the above aspects of coupled logical bisimilarity be specific to call-by-name, and that the call-by-value version would require non-trivial modifications.

REMARK 5.4. *In a coupled logical bisimulation, the first relation is used to construct the inputs for the tested functions (the formally summed values produced in the bisimulation game for the second relation). Therefore, such first relation may be thought of as a "global" environment— global because it is the same for each pair of terms on which the bisimulation game is played. As a consequence, coupled logical bisimulation remains quite different from environmental bisimulation [42], where the "environment" for constructing inputs is local to each pair of tested terms. Coupled logical bisimulation follows ordinary logical bisimulation [41], in which there is only one global environment; in ordinary logical bisimulation, however, the global environment coincides with the set of tested terms. The similarity with logical bisimulation is also revealed by non-monotonicity of the associated functional (in contrast, the functional associated to environmental bisimulation is monotone); see Remark 5.11.*

As an example of use of coupled logical bisimulation, we revisit the counterexample 3.12 to the completeness of applicative bisimilarity with respect to contextual equivalence.

EXAMPLE 5.5. *We consider the terms of Example 3.12 and show that they are in $\approx_1$, hence also in $\simeq_\oplus$ (contextual equivalence of $\Lambda_\oplus$), by Corollary 5.9 and $\simeq_\oplus^{\mathrm{FS}} = \simeq_\oplus$. Recall that the terms are $M \overset{\text{def}}{=} \lambda x.(L \oplus P)$ and $N \overset{\text{def}}{=} (\lambda x.L) \oplus (\lambda x.P)$ for $L \overset{\text{def}}{=} \lambda z.\Omega$ and $P \overset{\text{def}}{=} \lambda y.\lambda z.\Omega$. We set $\mathcal{R}_1$ to contain only (M, N) (this is the pair that interests us), and R_2 to contain the pairs $(M, N), (\langle M, 1\rangle, \langle \lambda x.L, \frac{1}{2}\rangle + \langle \lambda x.P, \frac{1}{2}\rangle), (\langle L + P, 1\rangle, \langle L, \frac{1}{2}\rangle + \langle P, \frac{1}{2}\rangle)$, and a set of pairs with identical components, namely $(\langle L, \frac{1}{2}\rangle + \langle P, \frac{1}{2}\rangle, \langle L, \frac{1}{2}\rangle + \langle P, \frac{1}{2}\rangle)$, $(\langle \Omega, \frac{1}{2}\rangle + \langle \lambda u.\Omega, \frac{1}{2}\rangle, \langle \Omega, \frac{1}{2}\rangle + \langle \lambda u.\Omega, \frac{1}{2}\rangle)$, $(\langle \lambda u.\Omega, \frac{1}{2}\rangle, \langle \lambda u.\Omega, \frac{1}{2}\rangle)$, $(\langle \Omega, \frac{1}{2}\rangle, \langle \Omega, \frac{1}{2}\rangle)$, $(\emptyset, \emptyset)$, where $\emptyset$ is the empty formal sum. Thus $(\mathcal{R}_1, \mathcal{R}_2)$ is a coupled logical bisimulation.*

The main challenge towards the goal of relating coupled logical bisimilarity and context equivalence is the substitutivity of bisimulation. We establish the latter exploiting some *up-to techniques* for bisimulation. We only give the definitions of the techniques, omitting the statements about their soundness. The first up-to technique allows us to drop the bisimulation game on silent actions:

DEFINITION 5.6 (Big-Step Bisimulation). *A coupled relation $\mathcal{R}$ is a* big-step coupled logical bisimulation *if whenever $E \; \mathcal{R}_2 \; F$, the following holds: if $E \overset{\rightsquigarrow}{\rightsquigarrow} Z$ then $F \overset{\rightsquigarrow}{\rightsquigarrow} Y$ with $\mathbf{\Sigma}(Z) = \mathbf{\Sigma}(Y)$, and for all $M \; \mathcal{R}_1^{\mathrm{C}} \; N$ we have $(Z \bullet M) \; \mathcal{R}_2 \; (Y \bullet N)$.*

In the reduction $\rightsquigarrow$, computation is performed at the level of formal sums; and this is reflected, in coupled bisimulation, by the application of values to formal sums only. The following up-to technique allows computation, and application of input values, also with ordinary terms. In the definition, we extract a formal sum from a term E in $\Lambda_\oplus^{\mathrm{FS}}$ using the function $\mathcal{D}(\cdot)$ inductively as follows:

$$\mathcal{D}(EM) \overset{\text{def}}{=} \Sigma_i \langle M_i M, p_i\rangle \text{ whenever } \mathcal{D}(E) = \Sigma_i \langle M_i, p_i\rangle;$$

$$\mathcal{D}(M) \overset{\text{def}}{=} \langle M, 1\rangle; \qquad\qquad \mathcal{D}(H) \overset{\text{def}}{=} H.$$

DEFINITION 5.7. *A coupled relation $\mathcal{R}$ is a* bisimulation up-to formal sums *if, whenever $E \; \mathcal{R}_2 \; F$, then either (one of the bisimulation clauses of Definition 5.3 applies), or ($E, F \in \Lambda_\oplus$ and one of the following clauses applies):*

1. *$E \rightsquigarrow D$ with $\mathcal{D}(D) = \langle M, \frac{1}{2}\rangle + \langle N, \frac{1}{2}\rangle$, and $F \rightsquigarrow G$ with $\mathcal{D}(G) = \langle L, \frac{1}{2}\rangle + \langle P, \frac{1}{2}\rangle$, $M \; \mathcal{R}_2 \; L$, and $N \; \mathcal{R}_2 \; P$;*
2. *$E = \lambda x.M$ and $F = \lambda x.N$, and for all $P \; \mathcal{R}_1^{\mathrm{C}} \; Q$ we have $M\{P/x\} \; \mathcal{R}_2 \; N\{Q/x\}$;*
3. *$E = (\lambda x.M)P\overline{M}$ and $F = (\lambda x.N)Q\overline{N}$, and $M\{P/x\}\overline{M} \; \mathcal{R}_2 \; N\{Q/x\}\overline{N}$.*

According to Definition 5.7, in the bisimulation game for a coupled relation, given a pair $(E, F) \in \mathcal{R}_2$, we can either choose to follow the bisimulation game in the original Definition 5.3; or, if E and F do not contain formal sums, we can try one of the new clauses above. The advantage of the first new clause is that it allows us to make a split on the derivatives of the original terms. The advantage of the other two new clauses is that they allow us to directly handle the given λ-terms, without using the operational rules of Figure 5 and therefore without introducing formal sums. To understand the first clause, suppose $E \overset{\text{def}}{=} (M \oplus N)L$ and $F \overset{\text{def}}{=} P \oplus Q$. We have $E \rightsquigarrow (\langle M, \frac{1}{2}\rangle + \langle N, \frac{1}{2}\rangle)L \overset{\text{def}}{=} G$ with $\mathcal{D}(G) = \langle ML, \frac{1}{2}\rangle + \langle NL, \frac{1}{2}\rangle$, and $F \rightsquigarrow \langle P, \frac{1}{2}\rangle + \langle Q, \frac{1}{2}\rangle \overset{\text{def}}{=} H$, with $\mathcal{D}(H) = H$, and it is sufficient now to ensure $(ML) \; \mathcal{R}_2 \; P$, and $(NL) \; \mathcal{R}_2 \; Q$.

Using the above proof technique, we can prove the necessary substitutivity property for bisimulation. The use of up-to techniques, and the way bisimulation is defined (in particular the presence of a clause for τ-steps and the possibility of using the pairs in the bisimulation itself to construct inputs for functions), make it possible to use a standard argument by induction over contexts.

LEMMA 5.8. *If $\mathcal{R}$ is a bisimulation then the context closure $\mathcal{S}$ with*

$$\mathcal{S}_1 \overset{\text{def}}{=} \mathcal{R}_1^{\mathrm{C}};$$

$$\mathcal{S}_2 \overset{\text{def}}{=} \mathcal{R}_2 \cup \mathcal{R}_1^{\mathrm{C}} \cup \{(E\overline{M}, F\overline{N}) \text{ s.t. } E \; \mathcal{R}_2 \; F \text{ and } M_i \; \mathcal{R}_1^{\mathrm{C}} \; N_i\};$$

is a bisimulation up-to formal sums.

Using Lemma 5.8 we can prove the inclusion in context equivalence.

COROLLARY 5.9. *If $M \approx_1 N$ then $M \simeq_\oplus^{\mathrm{FS}} N$. Moreover, if $E \approx_2 F$ then $E \approxeq_\oplus^{\mathrm{FS}} F$.*

The converse of Corollary 5.9 is proved exploiting a few simple properties of $\approxeq_\oplus^{\mathrm{FS}}$ (e.g., its transitivity, the inclusion $\rightsquigarrow \subseteq \approxeq_\oplus^{\mathrm{FS}}$).

THEOREM 5.10. *We have $\simeq_\oplus^{\mathrm{FS}} \subseteq \approx_1$, and $\approxeq_\oplus^{\mathrm{FS}} \subseteq \approx_2$.*

It also holds that coupled logical bisimilarity is preserved by the formal sum construct; i.e., $M_i \approx_1 N_i$ for each $i \in I$ implies $\Sigma_{i \in I}\langle M_i, p_i\rangle \approx_2 \Sigma_{i \in I}\langle N_i, p_i\rangle$. As a consequence, context equivalence defined on general $\Lambda_\oplus^{\mathrm{FS}}$ contexts is the same as that set on simple contexts (Definition 5.2).

REMARK 5.11. *The functional induced by coupled logical bisimulation is* not *monotone. For instance, if $\mathcal{V} \subseteq \mathcal{W}$, then a pair of terms may satisfy the bisimulation clauses on $(\mathcal{V}, \mathcal{E})$, for some $\mathcal{E}$,* *but not on $(\mathcal{W}, \mathcal{E})$, because the input for functions may be taken from the larger relation $\mathcal{W}$. (Recall that coupled relations are* pairs *of relations. Hence operations on coupled relations, such as union and inclusion, are defined component-wise.) However, Corollary 5.9 and Theorem 5.10 tell us that there is indeed a largest bisimulation, namely the pair $(\simeq_{\oplus}^{\mathrm{FS}}, \approxeq_{\oplus}^{\mathrm{FS}})$.*

With logical (as well as environmental) bisimulations, up-to techniques are particularly important to relieve the burden of proving concrete equalities. A powerful up-to technique in higher-order languages is *up-to contexts*. We present a form of up-to contexts combined with the big-step version of logical bisimilarity. Below, for a relation $\mathcal{R}$ on $\Lambda_{\oplus}$, we write $\mathcal{R}^{\mathrm{CFS}}$ for the closure of the relation under general (closing) $\Lambda_{\oplus}^{\mathrm{FS}}$ contexts.

DEFINITION 5.12. *A coupled relation $\mathcal{R}$ is a* big-step coupled logical bisimulation up-to contexts *if whenever $E \;\mathcal{R}_2\; F$, the following holds: if $E \rightsquigarrow Z$ then $F \rightsquigarrow Y$ with $\mathbf{\Sigma}(Z) = \mathbf{\Sigma}(Y)$, and for all $M \;\mathcal{R}_1^{\mathrm{C}}\; N$, we have $(Z \bullet M) \;\mathcal{R}_1^{\mathrm{CFS}}\; (Y \bullet N)$.*

For the soundness proof, we first derive the soundness of a small-step up-to context technique, whose proof, in turn, is similar to that of Lemma 5.8 (the up-to-formal-sums technique of Definition 5.7 already allows some context manipulation; we need this technique for the proof of the up-to-contexts technique).

EXAMPLE 5.13. *We have seen that the terms* `expone` *and* `exptwo` *of Example 2.5 are not applicative bisimilar. We can show that they are context equivalent, by proving that they are coupled bisimilar. We sketch a proof of this, in which we employ the up-to technique from Definition 5.12. We use the coupled relation $\mathcal{R}$ in which $\mathcal{R}_1 \overset{\text{def}}{=} \{(\texttt{expone}, \texttt{exptwo})\}$, and $\mathcal{R}_2 \overset{\text{def}}{=} \mathcal{R}_1 \cup \{(A_M, B_N) \mid M \;\mathcal{R}_1^{\mathrm{C}}\; N\}$ where $A_M \overset{\text{def}}{=} \lambda n.((Mn) \oplus (\texttt{expone}\, M\, (n+1)))$, and $B_N \overset{\text{def}}{=} (\lambda x.Nx) \oplus (\texttt{exptwo}\, (\lambda x.N(x+1)))$. This is a big-step coupled logical bisimulation up-to contexts. The interesting part is the matching argument for the terms A_M, B_N; upon receiving an argument m they yield the summed values $\Sigma_i \langle M(m+1), p_i \rangle$ and $\Sigma_i \langle N(m+1), p_i \rangle$ (for some p_i's), and these are in $\mathcal{R}_1^{\mathrm{CFS}}$.*

6. Beyond Call-by-Name Reduction

So far, we have studied the problem of giving sound (and sometime complete) coinductive methods for program equivalence in a probabilistic λ-calculus endowed with *call-by-name* reduction. One may wonder whether what we have obtained can be adapted to other notions of reduction, and in particular to *call-by-value* reduction (e.g., the call-by-value operational semantics of $\Lambda_{\oplus}$ from [9]).

Since our construction of a labelled Markov chain for $\Lambda_{\oplus}$ is somehow independent on the underlying operational semantics, defining a call-by-value probabilistic applicative bisimulation is effortless. The proofs of congruence of the bisimilarity and its soundness can also be transplanted to call-by-value.

When we restrict our attention to pure λ-terms, as we do in Section 4, we are strongly relying on call-by-name evaluation: LLT's only reflect term equivalence in a call-by-name lazy regime. We leave the task of generalizing the results to eager evaluation to future work, but we conjecture that, in that setting, probabilistic choice *alone* does not give contexts the same discriminating power as probabilistic bisimulation. Similarly we have not investigated the call-by-value version of coupled logical bisimilarity, as our current proofs rely on the appearance of formal sums only in redex position, a constraint that would probably have to be lifted for call-by-value.

7. A Comparison with Nondeterminism

Syntactically, $\Lambda_{\oplus}$ is identical to an eponymous language introduced by de'Liguoro and Piperno [13]. The semantics we present here, however, is quantitative, and this has of course a great impact on context equivalence. While in a nondeterministic setting what one observes is the *possibility* of converging (or of diverging, or both), terms with different convergence probabilities are considered different in an essential way here. Actually, nondeterministic context equivalence and probabilistic context equivalence are incomparable. As an example of terms that are context equivalent in the *must* sense but not probabilistically, we can take $I \oplus (I \oplus \Omega)$ and $I \oplus \Omega$. Conversely, I is probabilistically equivalent to any term M that reduces to $I \oplus M$ (which can be defined using fixed-point combinators), while I and M are not equivalent in the *must* sense, since the latter can diverge (the divergence is irrelevant probabilistically because it has probability zero). *May* context equivalence, in contrast, is coarser than probabilistic context equivalence.

Despite the differences, the two semantics have similarities. Analogously to what happens in nondeterministic λ-calculi, applicative bisimulation and context equivalence do *not* coincide in the probabilistic setting, at least if call-by-name is considered. The counterexamples to full abstraction are much more complicated in call-by-value λ-calculi [27], and cannot be easily adapted to the probabilistic setting.

8. Conclusions

This is the first paper in which bisimulation techniques for program equivalence are shown to be applicable to probabilistic λ-calculi.

On the one hand, Abramsky's idea of seeing interaction as application is shown to be amenable to a probabilistic treatment, giving rise to a congruence relation that is sound for context equivalence. Completeness, however, fails: the way probabilistic applicative bisimulation is defined allows one to distinguish terms that are context equivalent, but which behave differently as for *when* choices and interactions are performed. On the other, a notion of coupled logical bisimulation is introduced and proved to precisely characterise context equivalence for $\Lambda_{\oplus}$. Along the way, applicative bisimilarity is proved to coincide with context equivalence on pure λ-terms, yielding the Levy-Longo tree equality.

The crucial difference between the two main bisimulations studied in the paper is not the style (applicative *vis-à-vis* logical), but rather the fact that while applicative bisimulation insists on relating only individual terms, coupled logical bisimulation is more flexible and allows us to relate formal sums (which we may think as distributions). This also explains why we need distinct reduction rules for the two bisimulations. See examples 3.12 and 5.5. While not complete, applicative bisimulation, as it stands, is simpler to use than coupled logical bisimulation. Moreover it is a natural form of bisimulation, and it should be interesting trying to transport the techniques for handling it onto variants or extensions of the language.

Topics for future work abound — some have already been hinted at in earlier sections. Among the most interesting ones, one can mention the transport of applicative bisimulation onto the language $\Lambda_{\oplus}^{\mathrm{FS}}$. We conjecture that the resulting relation would coincide with coupled logical bisimilarity and context equivalence, but going through Howe's technique seems more difficult than for $\Lambda_{\oplus}$, given the infinitary nature of formal sums and their confinement to redex positions.

Also interesting would be a more *effective* notion of equivalence: even if the two introduced notions of bisimulation avoid universal quantifications over all possible contexts, they refer to an essentially infinitary operational semantics in which the meaning of a term is obtained as the least upper bound of all its finite approximations. Would it be possible to define bisimulation in terms of approximations without getting too fine grained?

Bisimulations in the style of logical bisimulation (or environmental bisimulation) are known to require up-to techniques in order to avoid tedious equality proofs on concrete terms. In the paper we have introduced some up-to techniques for coupled logical bisimilarity, but additional techniques would be useful. Up-to techniques could also be developed for applicative bisimilarity.

More in the long-run, we would like to develop sound operational techniques for so-called *computational indistinguishability*, a key notion in modern cryptography. Computational indistinguishability is defined similarly to context equivalence; the context is however required to work within appropriate resource bounds, while the two terms can have different observable behaviors (although with negligible probability). We see this work as a very first step in this direction: complexity bounds are not yet there, but probabilistic behaviour, an essential ingredient, is correctly taken into account.

Acknowledgments

The authors would like to thank the anonymous referees for the many interesting comments. This work is partially supported by the ANR project 12IS02001 ‘PACE’, and the project MIUR PRIN 2010LHT4KM ‘CINA’.

References

[1] S. Abramsky. The Lazy λ-Calculus. In D. Turner, editor, *Research Topics in Functional Programming*, pages 65–117. Addison Wesley, 1990.

[2] S. Abramsky and C.-H. L. Ong. Full abstraction in the lazy lambda calculus. *Inf. Comput.*, 105(2):159–267, 1993.

[3] E. Astesiano and G. Costa. Distributive semantics for nondeterministic typed lambda-calculi. *Theor. Comput. Sci.*, 32:121–156, 1984.

[4] H. P. Barendregt. *The Lambda Calculus – Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.

[5] M. Bernardo, R. De Nicola, and M. Loreti. A uniform framework for modeling nondeterministic, probabilistic, stochastic, or mixed processes and their behavioral equivalences. *Inf. Comput.*, 225:29–82, 2013.

[6] G. Boudol. Lambda-calculi for (strict) parallel functions. *Inf. Comput.*, 108(1):51–127, 1994.

[7] G. Boudol and C. Laneve. The discriminating power of the λ-calculus with multiplicities. *Inf. Comput.*, 126(1):83–102, 1996.

[8] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence,*, 25(5): 564–577, 2003.

[9] U. Dal Lago and M. Zorzi. Probabilistic operational semantics for the lambda calculus. *RAIRO - Theor. Inf. and Applic.*, 46(3):413–450, 2012.

[10] U. Dal Lago, D. Sangiorgi, and M. Alberti. On coinductive equivalences for probabilistic higher-order functional programs (long version). Available at `http://arxiv.org/abs/1311.1722`, 2013.

[11] V. Danos and R. Harmer. Probabilistic game semantics. *ACM Trans. Comput. Log.*, 3(3):359–382, 2002.

[12] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theor. Comput. Sci.*, 34:83–133, 1984.

[13] U. de’Liguoro and A. Piperno. Non deterministic extensions of untyped lambda-calculus. *Inf. Comput.*, 122(2):149–177, 1995.

[14] M. Dezani-Ciancaglini and E. Giovannetti. From bohm’s theorem to observational equivalences: an informal account. *Electr. Notes Theor. Comput. Sci.*, 50(2):83–116, 2001.

[15] M. Dezani-Ciancaglini, J. Tiuryn, and P. Urzyczyn. Discrimination by parallel observers: The algorithm. *Inf. Comput.*, 150(2):153–186, 1999.

[16] T. Ehrhard, M. Pagani, and C. Tasson. The computational meaning of probabilistic coherence spaces. In *LICS*, pages 87–96, 2011.

[17] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[18] N. D. Goodman. The principles and practice of probabilistic programming. In *POPL*, pages 399–402, 2013.

[19] A. D. Gordon. Bisimilarity as a theory of functional programming. *Electr. Notes Theor. Comput. Sci.*, 1:232–252, 1995.

[20] A. D. Gordon, M. Aizatulin, J. Borgström, G. Claret, T. Graepel, A. V. Nori, S. K. Rajamani, and C. V. Russo. A model-learner pattern for bayesian reasoning. In *POPL*, pages 403–416, 2013.

[21] M. Hennessy. Exploring probabilistic bisimulations, part I. *Formal Asp. Comput.*, 24(4-6):749–768, 2012.

[22] D. J. Howe. Proving congruence of bisimulation in functional programming languages. *Inf. Comput.*, 124(2):103–112, 1996.

[23] R. Jagadeesan and P. Panangaden. A domain-theoretic model for a higher-order process calculus. In *ICALP*, pages 181–194, 1990.

[24] C. Jones and G. D. Plotkin. A probabilistic powerdomain of evaluations. In *LICS*, pages 186–195, 1989.

[25] V. Koutavas, P. B. Levy, and E. Sumii. From applicative to environmental bisimulation. *Electr. Notes Theor. Comput. Sci.*, 276: 215–235, 2011.

[26] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991.

[27] S. B. Lassen. *Relational Reasoning about Functions and Nondeterminism.* PhD thesis, University of Aarhus, 1998.

[28] S. Lenglet, A. Schmitt, and J.-B. Stefani. Howe’s method for calculi with passivation. In *CONCUR*, pages 448–462, 2009.

[29] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

[30] C.-H. L. Ong. Non-determinism in a functional setting. In *LICS*, pages 275–286, 1993.

[31] P. Panangaden. *Labelled Markov Processes*. Imperial College Press, 2009.

[32] S. Park, F. Pfenning, and S. Thrun. A probabilistic language based on sampling functions. *ACM Trans. Program. Lang. Syst.*, 31(1), 2008.

[33] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[34] A. Pfeffer. IBAL: A probabilistic rational programming language. In *IJCAI*, pages 733–740. Morgan Kaufmann, 2001.

[35] A. M. Pitts. Operationally-based theories of program equivalence. In *Semantics and Logics of Computation*, pages 241–298. Cambridge University Press, 1997.

[36] A. M. Pitts. Howe’s method for higher-order languages. In D. Sangiorgi and J. Rutten, editors, *Advanced Topics in Bisimulation and Coinduction*, pages 197–232. Cambridge University Press, 2011.

[37] N. Ramsey and A. Pfeffer. Stochastic lambda calculus and monads of probability distributions. In *POPL*, pages 154–165, 2002.

[38] N. Saheb-Djahromi. Probabilistic LCF. In *MFCS*, volume 64 of *LNCS*, pages 442–451, 1978.

[39] D. Sands. From SOS rules to proof principles: An operational metatheory for functional languages. In *POPL*, pages 428–441, 1997.

[40] D. Sangiorgi and D. Walker. *The pi-Calculus – a theory of mobile processes.* Cambridge University Press, 2001.

[41] D. Sangiorgi, N. Kobayashi, and E. Sumii. Logical bisimulations and functional languages. In *FSEN*, volume 4767 of *LNCS*, pages 364–379, 2007.

[42] D. Sangiorgi, N. Kobayashi, and E. Sumii. Environmental bisimulations for higher-order languages. *ACM Trans. Program. Lang. Syst.*, 33(1):5, 2011.

[43] K. Sieber. Call-by-value and nondeterminism. In *TLCA*, volume 664 of *LNCS*, pages 376–390, 1993.

[44] S. Thrun. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, pages 1–35, 2002.