Environmental bisimulations for probabilistic higher-order languages

Environmental Bisimulations for Probabilistic Higher-Order Languages

Davide Sangiorgi Valeria Vignudelli
University of Bologna & INRIA
email?

Abstract

Environmental bisimulations for probabilistic higher-order languages are studied. In contrast with applicative bisimulations, environmental bisimulations are known to be more robust and do not require sophisticated techniques such as Howe's in the proofs of congruence.

As representative calculi, call-by-name and call-by-value λ-calculus, and a (call-by-value) λ-calculus extended with references (i.e., a store) are considered. In each case full abstraction results are derived for probabilistic environmental similarity and bisimilarity with respect to contextual preorder and contextual equivalence, respectively. Some possible enhancements of the (bi)simulations, as 'up-to techniques', are also presented.

Probabilities force a number of modifications to the definition of environmental bisimulations in non-probabilistic languages. Some of these modifications are specific to probabilities, others may be seen as general refinements of environmental bisimulations, applicable also to non-probabilistic languages. Several examples are presented, to illustrate the modifications and the differences.

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 Environmental Bisimulation; Probabilistic Lambda Calculus; Contextual Equivalence; Imperative Languages

1. Introduction

The general topic of the paper are techniques for proving behavioural equivalence 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; ideally it should coincide with context equivalence. Bisimulation has been studied in depth in deterministic λ-calculi, e.g., [2, 12, 22, 35, 39]. In contrast, so far, it has been little explored in probabilistic λ-calculi, mainly in the form of *applicative bisimilarity* [7, 9] for the pure call-by-name and call-by-value languages.

Applicative bisimulations are known however to have some significant limitations. First, they do not scale very well to languages richer than pure λ-calculus. For instance, they are unsound under the presence of references, or even just generative names, or data abstraction; see [21] for an enlightening discussion. Secondly, congruence proofs of applicative bisimulations are notoriously hard. Such proofs usually rely on Howe's method [14], which is however fragile outside the pure λ-calculus. Related to the problems with congruence are also the difficulties of applicative bisimulations with "up-to context" techniques (the usefulness of these techniques in higher-order languages and its problems with applicative bisimulations have been studied by Lassen [22]; see also [19, 36, 39]).

To relieve this burden, *environmental bisimulations* have been proposed [42], refining earlier proposals in [1, 5, 16, 19, 45]. The distinguishing feature of these bisimulations is that the pairs of tested terms are enriched with environments, that intuitively collect the observer's knowledge about values computed during the bisimulation game. The elements of the environment can be used to construct terms to be supplied as inputs during the bisimulation game. The notion has been applied to a variety of languages, including pure λ-calculi [42, 46], extensions of λ-calculi [3, 19, 20, 45], or languages for concurrency or distribution [32, 33, 43]. In environmental bisimulations the proof of congruence goes by induction over contexts, as in proofs for first-order languages. For this, the proofs in the literature rely on a 'small-step' reduction relation. This allows a tight control over the syntax of the contexts, which fails with big-step reductions because in a higher-order language contexts may arbitrarily grow during reduction.

With probabilities, the drawbacks of applicative bisimilarity are magnified: full abstraction with respect to contextual equivalence may fail also in a pure λ-calculus, and Howe's technique has to be enriched with non-trivial 'disentangling' properties for sets of real numbers, these properties themselves proved by modeling the problem as a flow network and then applying the Max-flow Min-cut Theorem.

In this paper, our goal is understanding environmental bisimulations in probabilistic higher-order languages. As representative

calculi we consider call-by-name and call-by-value λ-calculus, and a (call-by-value) λ-calculus extended with higher-order references. The computation is made probabilistic by endowing these calculi with a primitive for binary, fair, probabilistic choice. In each case we derive full abstraction results for probabilistic environmental similarity and bisimilarity with respect to contextual preorder and contextual equivalence, respectively. Below, we discuss the main differences of our proposals in comparison with ordinary (i.e., non probabilistic) environmental (bi)simulations.

Static and dynamic environments In ordinary environmental bisimulation the values produced during the bisimulation game are placed into the environment, so that the observer can later play them at will during the bisimulation game. This schema is irrespective of the evaluation strategy (call-by-name or call-by-value), and is *the* distinguishing feature of environmental bisimulations over the applicative ones. 'Playing a term' means copying it. However, in the λ-calculus the copying possibilities for call-by-name and call-by-value are quite different. In call-by-name, evaluation only occurs in functional position and therefore the term resulting from the evaluation may not be copied. In call-by-value, in contrast, a term may be evaluated also in argument position, and then given as input to a function; thus copying is possible also after evaluation. The different copying behaviour is well visible, for instance, in linear logic interpretations of call-by-name and call-by-value [26].

Now, the semantics of probabilistic languages is sensitive to the copying operation; for instance the probability of success of an experiment, if non-trivial, may be lowered by playing the experiment several times. This has a strong impact on behavioural equivalences for call-by-name and call-by-value in probabilistic λ-calculi. As an example, using Ω for a purely divergent term,

$$A \stackrel{\mathtt{def}}{=} \lambda x.(x \oplus \Omega) \quad \text{and} \quad B \stackrel{\mathtt{def}}{=} (\lambda x.x) \oplus (\lambda x.\Omega) \qquad (1)$$

are contextually equivalent in call-by-name: if evaluated alone they always terminate; if evaluated with an argument, they return the argument with the same probability. More generally, in call-by-name abstraction distributes over probabilistic choice. In contrast, distributivity fails in call-by-value, exploiting the possibility of copying evaluated terms; e.g., the probabilities of termination for A and B are different in the context $(\lambda x.x\,(x\,\lambda y.y))[\cdot]$.

To be able to express such behavioural differences, in our environmental bisimulations the values produced during the bisimulation game are placed into the environment *only* in call-by-value. We call such a value environment a *dynamic environment* because it may grow during the bisimulation game. It is precisely the use of the dynamic environment that allows us to separate the two terms A and B above. In probabilistic call-by-name, dynamic environments would break full abstraction for contextual equivalence. The only environment for call-by-name is *static*. The static environment for two compared objects F, G is a pair of λ-terms M, N, which are, intuitively, the initial λ-terms from which, using evaluation and interaction according to the bisimulation game, the objects F, G have been derived. This (small) static environment is sufficient to ensure that the congruence proof of the bisimilarity remains in the style of ordinary environmental bisimulation (i.e., it does not require sophisticated techniques such as Howe's). In short, the static environment reflects the copying possibility for terms before evaluation, whereas the dynamic environment reflects the copying possibility for values resulting from evaluation.

Formal sums In our probabilistic relations the objects compared are not plain λ-terms but *formal sums*, that are the objects produced by the semantics of a term. These are, intuitively, syntactic representations of probability distributions. As a consequence, environments are not just tuples of values, but formal sums of tuples of values. To see why related objects must be formal sums, consider again the terms A and B in (1): our environmental bisimulation for call-by-name equates A and B by relating A and the formal sum resulting from the evaluation of B. None of the components of the formal sum, $\lambda x.x$ and $\lambda x.\Omega$, could separately be related with A. (A form of bisimulation on formal sums, namely a probabilistic version of *logical bisimulation*, is already defined in [9] for call-by-name; its drawbacks are discussed in Section 6.)

In pure call-by-value λ-calculus, full abstraction for contextual equivalence would also hold without formal sums (i.e., relating plain λ-terms), for the same reason why, in the same language, applicative bisimilarity on plain terms is fully abstract [7]. We do not pursue this simplification of environmental bisimulations because it would be unsound in extensions of the calculus. For instance, consider the following terms of a probabilistic λ-calculus with store (again, an instance of distributivity):

$$H \stackrel{\mathtt{def}}{=} (\boldsymbol{\nu}\, x := 0)(\lambda.(M \oplus N)) \quad K \stackrel{\mathtt{def}}{=} (\boldsymbol{\nu}\, x := 0)(\lambda.M \oplus \lambda.N)$$

where $(\boldsymbol{\nu}\, x := 0)$ indicates the creation of a new reference, initialised with 0, $\lambda.L$ is a thunk (i.e., $\lambda z.L$ for z not free in L), and where, using $L_1 \,\underline{\mathtt{seq}}\, L_2$ for the sequential evaluation of L_1 followed by L_2,

$$M \stackrel{\mathtt{def}}{=} \mathtt{if}\ !x = 0\ \mathtt{then}\ (x := 1 \,\underline{\mathtt{seq}}\, \mathtt{true})\ \mathtt{else}\ \Omega$$
$$N \stackrel{\mathtt{def}}{=} \mathtt{if}\ !x = 0\ \mathtt{then}\ (x := 1 \,\underline{\mathtt{seq}}\, \mathtt{false})\ \mathtt{else}\ \Omega\,.$$

The terms M and N only differ at their first evaluation, when the value of x is set to 1 and M produces `true` whereas N produces `false`; thereafter x is 1 and both terms diverge. As a consequence, H and K are contextually equivalent: at their first evaluation they always terminate, at later evaluations they always diverge.

To place H and K in a bisimulation, H has to be related with the formal sum obtained from the evaluation of K; again, the single components alone would be distinguished. Once more, this is a copying issue, due to the possibility of copying terms but not stores.

Big-step reduction, term closure, and congruence proof To achieve full abstraction, in the probabilistic case the bisimulation clauses have to use a big-step, rather than a small-step, reduction relation. Precisely, we give finitary big-step approximants from which the semantics of a term is obtained via a least-fixed point construction (a similar semantics is [8]). The reason, intuitively, is that while in pure λ-calculi a term converges (i.e., it terminates its computation) in a finite number of reductions, in the probabilistic calculi there may be several terminating computation paths, in which the total number of reductions need not be finitary. An example is given by the terms

$$P \stackrel{\mathtt{def}}{=} RR \quad \text{and} \quad Q \stackrel{\mathtt{def}}{=} \lambda.\Omega\,, \quad \text{for} \quad R \stackrel{\mathtt{def}}{=} \lambda x.(xx) \oplus Q \qquad (2)$$

where $\oplus$ denotes a probabilistic choice. The terms P and Q are contextually equal, intuitively because they both have probability 1 of becoming term Q: after some reductions, P may become Q or may become P again, with equal probability. Only by exploring the whole computation tree produced by P does one find out that the infinite number of leaves in the tree make a probability 1 of obtaining Q. None of the finite approximants of the infinite tree gives the same information (a formal sum made of a finite subset of the leaves would not be equivalent to Q).

When the reduction relation is small-step, as in ordinary environmental bisimulations, the related terms need not be values, because a normalising term need not produce a value in a single step and bisimulations must be closed under the reduction adopted. In contrast, as our environmental bisimulations are big-steps, the bisimulation game may be confined to values.

A more significant consequence of the adoption of big-step reductions is that the induction over contexts in the proofs of congruence for ordinary environmental bisimulations is replaced by an

induction on the number of small-step reductions with which a big-step approximant is derived (possibly coupled with an induction on the size of a context), combined with two levels of continuity arguments. One level stems from the least fixed point construction employed in the definition of the infinitary big-step semantic on terms. The second level stems from a characterisation of bisimilarity as the kernel of the similarity preorder and, in turn, as the kernel of a finitary similarity in which (on the challenger side) the big-step reduction relation employed is finite. The proof of the characterisation with the finitary similarity makes use of least fixed-points via a saturation construction on formal sums where, intuitively, a formal sum is better than another formal sum if the former conveys more accurate probabilistic information than the latter.

Up-to techniques Our proofs and examples rely on a few enhancements of the bisimulation proof method ('bisimulations up-to'), some of which are extensions of common (bi)simulation enhancements, others are specific to probabilistic calculi. An example of the latter is 'simulation up-to lifting', whereby it is sufficient, in the coinductive game, that two derivative formal sums are in the probabilistic lifting of the candidate relation, rather than in the candidate relation itself.

While the bisimulations act on formal sums and use infinitary big-step reductions to values, we also explore coinductive games played on plain λ-terms and on finitary multi-step reductions to terms (not necessarily values) as sound proof techniques. In particular, we combine these with up-to context, so to be able to compare terms in the middle of their evaluation when a common context can be isolated and removed.

Structure of the paper Section 2 introduces some general definitions and notations for the paper. Section 3 presents environmental bisimulations for pure call-by-name, establishes basic properties including full abstraction for bisimilarity and similarity, and develops various up-to techniques. Section 4 considers pure call-by-value, and Section 5 an extension with ML-like references. Section 6 discusses additional related work, and Section 7 concludes, also mentioning possible future work.

2. Preliminaries

We introduce general notations and terminologies for the paper. Familiarity with standard terminologies (such as free/bound variables, and α-conversion) is assumed.

We use meta-variables $M, N, P, Q, \ldots$ for terms, and $V, W, \ldots$ for values. We identify α-convertible terms. We write $M\{^N/_x\}$ for the capture-avoiding substitution of N for x in M. A term is *closed* if it contains no free variables. The set of free variables of a term M is $\mathsf{fv}(M)$. A *context* C is an expression obtained from a term by replacing some subterms with *holes* of the form $[\cdot]_i$. We write $C[M_1, \ldots, M_n]$ for the term obtained by replacing each occurrence of $[\cdot]_i$ in C with M_i.

We use a tilde to denote a tuple; for instance $\widetilde{M}$ is a tuple of terms, and $(\widetilde{M})_i$ is its i-th element. Sometimes we write tuples as $\{M_i\}_i$ when we want to emphasize the indexing set. All notations are extended to tuples componentwise.

By default, the results and definitions in the paper are (implicitly) stated for closed terms. They can be generalized to open terms in a standard way for bisimulations in λ-calculi [19, 42, 45], essentially deriving properties between open terms M and N from the corresponding properties between the closed terms $\lambda\widetilde{x}.M$ and $\lambda\widetilde{x}.N$, for $\{\widetilde{x}\} \supseteq \mathsf{fv}(M) \cup \mathsf{fv}(N)$.

The pure λ-calculi will be untyped, whereas we will find types convenient to treat the extension with store.

3. Probabilistic call-by-name λ-calculus

The terms of the probabilistic λ-calculus are generated by the following grammar:

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

We write $\Lambda^{\bullet}_{\oplus}$ for the subset of closed terms. The *values* are the terms of the form $\lambda x.M$ (the abstractions). In call-by-name the *evaluation contexts* (which, in contrast with standard contexts, may have only one occurrence of a single hole $[\cdot]$) are:

$$C := CM \mid [\cdot]$$

In probabilistic languages, the semantics of a term is usually a *distribution*, that is, a function that specifies the probabilities of all possible outcomes for that term [8]. We prefer, in contrast, syntactic representations of distributions, as *formal sums*, because they allow us a tighter control on the manipulations of the operational semantics, which is important in various places of our conductive definitions and proofs. Formal sums have the form

$$\textstyle\sum_{i\in I} p_i; M_i$$

where $0 < p_i \leq 1$, for each i, $\sum_{i\in I} p_i \leq 1$, and I is a (possibly infinite) indexing set. In a summand $p_i; M_i$ of a formal sum, p_i is its *probability value* (or *weight*), and M_i is its *term*. The terms of different summands of a formal sum need not be different. The weight $\mathtt{weight}(\sum_{i\in I} p_i; M_i)$ of a formal sum is $\sum_{i\in I} p_i$. We let F, G range over formal sums, and write $F = G$ if F and G are syntactically equal modulo a permutation of the summands. We use '$+$' for binary sums, in the usual infix form, and sometimes apply it also to formal sums, as in $F + G$. We write the empty formal sum as $\emptyset$. *Value formal sums*, ranged over by Y, Z are formal sums in which the term of each summand is a value.

There is an obvious mapping from formal sums to distributions, whereby a formal sum F yields the distribution in which the probability of a term M is the sum of the weights with which M appears in summands of F. The mapping is not injective: in general, infinitely many formal sums yield the same distribution (because of possible duplicates in the terms of the summands of a formal sum).

We sometimes decompose formal sums using a lifting construction. If $F_i = \sum_{j\in J_i} p_{i,j}; M_{i,j}$, for $i \in I$, then

$$\textstyle\sum_i p_i{\cdot}F_i \stackrel{\mathtt{def}}{=} \sum_{i\in I, j\in J_i} p_i \cdot p_{i,j}; M_{i,j}\ .$$

The semantics of a term M, written $[\![M]\!]$, is a value formal sum produced as the supremum of the value formal sums obtained by finite computations starting from M, using a preorder $\leq_{\mathtt{apx}}$ on formal sums in which $F_1 \leq_{\mathtt{apx}} F_2$ if F_1 is an approximant of F_2 (in other words F_2 conveys more information than F_1); formally, $F_2 = F_1 + G$ for some G. The semantics is obtained in various steps, whose rules are presented in Figure 1:

1. a single-step reduction relation $\longrightarrow$ from terms to formal sums;
2. a multi-step reduction relation $\Longrightarrow$ from terms and formal sums to formal sums, from which a relation $\Longmapsto$ to value formal sums is extracted by retaining only the summands whose term is a value via the function $\mathtt{val}$:
$$\mathtt{val}(\textstyle\sum_i p_i; M_i) \stackrel{\mathtt{def}}{=} \sum_{\{i \mid M_i \text{is a value}\}} p_i; M_i\ ;$$
3. the semantics $[\![\ ]\!]$, mapping terms and formal sums to value formal sums via the supremum construction.

If $M \Longrightarrow \sum_{i\in I} p_i; M_i$ then I is finite, and each i represents a 'possible world' of the probabilistic run of M, with probability p_i and outcome M_i. The subset of possible worlds in which M_i is a value makes for an approximant of M, and from such approximants the semantics of M is obtained.

Since value formal sums form an ω-complete partial order with respect to the $\leq_{\mathtt{apx}}$ preorder, and for every M the set of those value formal sums Y such that $M \Longmapsto Y$ is a countable directed set, the semantics $[\![M]\!]$ of a term M exists and is unique.

Relations $\Longrightarrow$ and $\Longmapsto$ are finitary in the sense that a derivation proof where one of such relations appears in the conclusion only contains a finite number of 'small steps' (relation $\longrightarrow$). When reasoning by induction, sometimes we will need to make such number explicit, therefore writing $\Longrightarrow_n$ and $\Longmapsto_n$, respectively.

Rule MULT, in contrast with MULFS, does not need a finitary condition on the indexing set because a formal sum obtained in a small step from a term may have at most two summands.

3.1 Environmental bisimulation in call-by-name

In call-by-name, a *probabilistic environmental relation* is a set of elements each of which is of the form (M, N) or $((M, N), Y, Z)$, where M, N, Y, Z are all closed, M, N are $\Lambda^{\bullet}_{\oplus}$-terms and Y, Z value formal sums. Intuitively, in the former elements M and N are terms that we wish to prove equal, and in the latter elements Y and Z are value formal sums obtained from M and N via evaluations and interactions with the environment. We use $\mathcal{R}, \mathcal{S}$ to range over probabilistic environmental relations. In a triple $((M, N), Y, Z)$ the pair component (M, N) is the *static environment*, and Y, Z are the *tested formal sums*. We write $\mathcal{R}_{(M,N)}$ for the relation $\{(Y, Z) \mid ((M, N), Y, Z) \in \mathcal{R}\}$; we accordingly use the infix notation $Y\ \mathcal{R}_{(M,N)}\ Z$, and similarly for $M\ \mathcal{R}\ N$. In the remainder of the paper, when discussing probabilistic environmental relations, bisimulations, simulations, or similar, we abbreviate 'probabilistic environmental' as 'PE', or even omit it when non-ambiguous. Static environments (that is, pairs of $\Lambda^{\bullet}_{\oplus}$-terms) are ranged over by $\mathcal{E}$. If $\mathcal{E} = (M, N)$ then its *context closure*, written $\mathcal{E}^{\star}$, is the set of all pairs of the form $(C[M], C[N])$. We use a similar notation for the context closure of relations on λ-terms.

Remark 3.1 (Static environment)**.** The results in the paper would also hold admitting arbitrary sets of pairs of $\Lambda^{\bullet}_{\oplus}$-terms as static environments, rather then single pairs. We have chosen single pairs so to bring up the minimal requirement on static environments for our proofs to hold (notably the congruence for bisimilarity). □

Definition 3.2 (Environmental bisimulation, call-by-name)**.** *A PE relation $\mathcal{R}$ is a* (PE) bisimulation *if*

1. *$M\ \mathcal{R}\ N$ implies $[\![M]\!]\ \mathcal{R}_{(M,N)}\ [\![N]\!]$;*
2. *$\sum_i p_i; \lambda x.M_i\ \mathcal{R}_{\mathcal{E}}\ \sum_j q_j; \lambda x.N_j$ implies:*
   (a) *$\sum_i p_i = \sum_j q_j$;*
   (b) *for all $P, Q \in \mathcal{E}^{\star}$,*
   $\sum_i p_i \cdot [\![M_i\{^P/_x\}]\!]\ \mathcal{R}_{\mathcal{E}}\ \sum_j q_j \cdot [\![N_j\{^Q/_x\}]\!]$.

We write $\approx$ for (PE) bisimilarity*, the union of all PE bisimulations.*

While $\approx$ is a PE relation, we are ultimately interested in comparing λ-terms ($M \approx N$ if $M\ \mathcal{R}\ N$ for some bisimulation $\mathcal{R}$).

Example 3.3. We have

$$M \stackrel{\mathtt{def}}{=} (\lambda.\lambda.\Omega) \oplus (\lambda.\Omega) \approx \lambda.(\lambda.\Omega \oplus \Omega) \stackrel{\mathtt{def}}{=} N\,.$$

This is proved noting that $[\![M]\!] = \frac{1}{2}; \lambda.\lambda.\Omega + \frac{1}{2}; \lambda.\Omega$ and $[\![N]\!] = 1; N$, using the bisimulation $\mathcal{R}$ in which $M\ \mathcal{R}\ N$, $[\![M]\!]\ \mathcal{R}_{(M,N)}\ [\![N]\!]$, $\frac{1}{2}; \lambda.\Omega\ \mathcal{R}_{(M,N)}\ \frac{1}{2}; \lambda.\Omega$, and $\emptyset\ \mathcal{R}_{(M,N)}\ \emptyset$. Terms M, N could not be equated by a bisimulation that acted only on terms (ignoring formal sums), as neither $\lambda.\lambda.\Omega$ nor $\lambda.\Omega$ can be equated to N. □

Definition 3.4 (Simulation)**.** *In Definition 3.2, and in the remainder of the paper for other definitions of probabilistic bisimulation, the corresponding* simulation *is obtained by replacing the equality '=' on the weights with '$\leq$'; thus in Definition 3.2, clause (2.a) becomes $\sum_i p_i \leq \sum_j q_j$.*

The union of all simulations, similarity*, is written $\lesssim$.*

Theorem 3.5. *1. $\approx$ and $\lesssim$ are the largest bisimulation and simulation, respectively.*
2. $\lesssim$ is a preorder, and $\approx$ an equivalence.
3. $\approx\ =\ \lesssim \cap \lesssim^{-1}$

The bisimilarity, or similarity, game uses the semantics of terms, which is a least-fixed point on top of big-step approximants. When proving properties about bisimilarity and similarity, therefore, we need to reason about such approximants. For this, we introduce a finite-step simulation in which the challenge reductions of the simulation game employs the big-step approximants (the relation $\Longmapsto$ of Figure 1). We cannot have characterisations of bisimilarity in terms of a finite-step *bi*-similarity because in general the weights of the approximants of two bisimilar terms are different, as shown in Example 3.6. Hence to reason about bisimilarity we go through its characterisation via similarity (Theorem 3.5), and then the characterisation of similarity via the finite-step similarity (Corollary 3.10).

Example 3.6. Let P and Q be the terms discussed in (2) in the Introduction. A bisimulation relating P and Q is

$$\{(P, Q), ((P, Q), \textstyle\sum_{n\geq 1} \frac{1}{2^n}; Q, 1; Q), ((P, Q), \emptyset, \emptyset)\}\,.$$

We could not prove the equality using finite-step approximants for bisimulation, since those for P are of the form $\sum_{1\leq n\leq m} \frac{1}{2^n}; Q$, for some m, and thus have a smaller total weight than the formal sum $1; Q$ immediately produced by Q. □

Definition 3.7. *A PE relation $\mathcal{R}$ is a* finite-step simulation *if*

1. *$M\ \mathcal{R}\ N$ and $M \Longmapsto Y$ imply $Y\ \mathcal{R}_{(M,N)}\ [\![N]\!]$;*
2. *$\sum_i p_i; \lambda x.M_i\ \mathcal{R}_{\mathcal{E}}\ \sum_j q_j; \lambda x.N_j$ implies:*
   (a) *$\sum_i p_i \leq \sum_j q_j$;*
   (b) *for all $P, Q \in \mathcal{E}^{\star}$, if $\sum_i p_i; M_i\{^P/_x\} \Longmapsto Y$ then $Y\ \mathcal{R}_{\mathcal{E}}\ \sum_j q_j \cdot [\![N_j\{^Q/_x\}]\!]$.*

We write $\lesssim_{\mathrm{fin}}$ for *finite-step similarity*. In finite-step simulations, the challenges are expressed by finitary reductions. Moreover, any $\lesssim_{\mathrm{fin}}$ result on $\Lambda^{\bullet}_{\oplus}$-terms can be established using a finite-step simulation with finite formal sums on the challenger side, though this constraint is not required in the definition.

Remark 3.8. Clause (2b) of Definition 3.7 cannot be written thus:
for all $P, Q \in \mathcal{E}^{\star}$, if $M_i\{^P/_x\} \Longmapsto Y_i$ for every i
then $\sum_i p_i \cdot Y_i\ \mathcal{R}_{\mathcal{E}}\ \sum_j q_j \cdot [\![N_j\{^Q/_x\}]\!]$
because, as the index set I can be infinite, the challenge in the bisimulation game might not be finitary. By contrast, reduction $\Longmapsto$ on formal sums (from Figure 1) is finitary. This allows proofs by induction on the number of single-steps in a reduction. □

We denote by $\mathtt{Pairs}(\mathcal{R})$ the set of pairs of terms in a PE relation $\mathcal{R}$. We use two saturation constructions to turn a simulation into a finite-step simulation and conversely. Given a PE relation $\mathcal{R}$, its *saturation by approximants* is

$$\mathtt{Pairs}(\mathcal{R}) \cup \{(\mathcal{E}, Y, Z) \mid \text{there is } Y' \text{ with } Y'\ \mathcal{R}_{\mathcal{E}}\ Z \text{ and } Y \leq_{\mathtt{apx}} Y'\}$$

and its *saturation by suprema* is $\bigcup_n \mathcal{R}^n$, where

$$\mathcal{R}^0 \stackrel{\mathtt{def}}{=} \mathcal{R}$$
$$\mathcal{R}^{n+1} \stackrel{\mathtt{def}}{=} \mathcal{R}^n \cup \{(\mathcal{E}, Y, Z) \mid \text{there are } \{Y_i\}_i \text{ with } Y_i\ \mathcal{R}^n_{\mathcal{E}}\ Z,\ Y_i \leq_{\mathtt{apx}} Y_{i+1}, \text{ and } Y = \sup\{Y_i\}_i\}.$$

Lemma 3.9. *1. The saturation by approximants of a simulation is a finite-step simulation.*
2. The saturation by suprema of a finite-step simulation is a simulation.

single-step reduction relation from terms to formal sums

$$\textsc{Beta}\ \frac{}{(\lambda x.M)N \longrightarrow 1; M\{^N/_x\}} \qquad \textsc{Sum}\ \frac{}{M_1 \oplus M_2 \longrightarrow \frac{1}{2}; M_1 + \frac{1}{2}; M_2} \qquad \textsc{Eval}\ \frac{M \longrightarrow \sum_i p_i; M_i \quad C \text{ is an evaluation context}}{C[M] \longrightarrow \sum_i p_i; C[M_i]}$$

multi-step reduction relation from terms to formal sums

$$\textsc{Mul0}\ \frac{}{M \Longrightarrow 1; M} \qquad \textsc{MulT}\ \frac{M \longrightarrow \sum_i p_i; M_i \qquad M_i \Longrightarrow F_i}{M \Longrightarrow \sum_i p_i; F_i}$$

multi-step reduction relation from formal sums to formal sums:

$$\textsc{MulFS}\ \frac{M_i \Longrightarrow F_i}{\sum_{i\in I} p_i; M_i + G \Longrightarrow \sum_{i\in I} p_i; F_i + G}\ I \text{ finite}$$

multi-steps reduction relation terms and formal sums to value formal sums

$$\textsc{MulVT}\ \frac{M \Longrightarrow F \qquad \mathtt{val}(F) = Y}{M \Longmapsto Y} \qquad \textsc{MulVFS}\ \frac{F \Longrightarrow F' \qquad \mathtt{val}(F') = Y}{F \Longmapsto Y}$$

the semantic mapping, from terms and formal sums to value formal sums

$$[\![M]\!] \stackrel{\mathtt{def}}{=} \sup\{Y | M \Longmapsto Y\} \qquad [\![F]\!] \stackrel{\mathtt{def}}{=} \sup\{Y | F \Longmapsto Y\}$$

Figure 1. Operational semantics for call-by-name

Proof. The proof of (1) follows from the definition of $[\![M]\!]$ as the supremum of the set $\{Y | M \Longmapsto Y\}$. For (2), the crux is proving by induction on n that if $\sum_i p_i; \lambda x.M_i \ \mathcal{R}^n_{\mathcal{E}}\ \sum_j q_j; \lambda x.N_j$ then:

1. $\sum_i p_i \leq \sum_j q_j$;
2. $\sum_i p_i; M_i\{^P/_x\} \Longmapsto Y$ implies $Y\ \mathcal{R}^n_{\mathcal{E}}\ \sum_j q_j \cdot [\![N_j\{^Q/_x\}]\!]$, for all $P, Q \in \mathcal{E}^\star$.

□

Corollary 3.10. *The similarity and finite-step similarity preorders, $\lesssim$ and $\lesssim_{\mathrm{fin}}$, coincide.*

The following example highlights the differences between simulations and finite-step simulations, by proving the equality in Example 3.6 using finite-step simulations.

Example 3.11. Terms P and Q of Example 3.6 can be proved equivalent by exhibiting the following finite-step simulations, where $Y_0 \stackrel{\mathtt{def}}{=} \emptyset$ and $Y_m \stackrel{\mathtt{def}}{=} \sum_{1\leq n\leq m} \frac{1}{2^n}; Q$ for $m \geq 1$:

$\mathcal{R} \stackrel{\mathtt{def}}{=} \{(P,Q), ((P,Q), Y_m, 1; Q), ((P,Q), \emptyset, \emptyset)\}$

$\mathcal{S} \stackrel{\mathtt{def}}{=} \{(Q,P), ((Q,P), 1; Q, \sum_{n\geq 1} \frac{1}{2^n}; Q), ((Q,P), \emptyset, \emptyset)\}$.

□

To derive the substitutivity properties of the similarity, and hence of the bisimilarity, we also need an up-to technique for the finite-step similarity. Specifically, we need an up-to lifting technique whereby, in the simulation game, two derivative formal sums can be decomposed into 'smaller' formal sums and it is then sufficient that these are pairwise related. We write $\mathtt{lift}(\mathcal{S})$ for the probabilistic lifting of a relation $\mathcal{S}$ on formal sums:

$$\mathtt{lift}(\mathcal{S}) \stackrel{\mathtt{def}}{=} \{(F,G) \mid \text{there are } I, p_i, F_i, G_i, \text{ for } i \in I, \text{ with } F_i\ \mathcal{S}\ G_i \text{ and } F = \textstyle\sum_i p_i \cdot F_i \text{ and } G = \sum_i p_i \cdot G_i\}$$

Definition 3.12. *A PE relation $\mathcal{R}$ is a* finite-step simulation up-to lifting *if*

1. *$M\,\mathcal{R}\,N$ and $M \Longmapsto Y$ imply $Y\ \mathtt{lift}(\mathcal{R}_{(M,N)})\ [\![N]\!]$;*
2. *$\sum_i p_i; \lambda x.M_i\ \mathcal{R}_{\mathcal{E}}\ \sum_j q_j; \lambda x.N_j$ implies:*
   (a) $\sum_i p_i \leq \sum_j q_j$;
   (b) for all $P, Q \in \mathcal{E}^\star$, if $\sum_i p_i; M_i\{^P/_x\} \Longmapsto Y$ then $Y\ \mathtt{lift}(\mathcal{R}_{\mathcal{E}})\ \sum_j q_j \cdot [\![N_j\{^Q/_x\}]\!]$.

Lemma 3.13. *If $\mathcal{R}$ is a finite-step simulation up-to lifting then $\mathcal{R} \subseteq \lesssim_{\mathrm{fin}}$.*

Example 3.14. Let $P \stackrel{\mathtt{def}}{=} \lambda.\Omega$, $Q \stackrel{\mathtt{def}}{=} \lambda.\lambda.\Omega$, and

$$M \stackrel{\mathtt{def}}{=} (P \oplus P) \oplus (Q \oplus Q)\,, \quad N \stackrel{\mathtt{def}}{=} (P \oplus Q) \oplus (P \oplus Q)\,.$$

The following finite-step simulation up-to lifting shows $M \lesssim_{\mathrm{fin}} N$:

$$\mathcal{R} \stackrel{\mathtt{def}}{=} \{(M,N), ((M,N), 1; P, 1; P), ((M,N), 1; Q, 1; Q), ((M,N), \emptyset, 1; P), ((M,N), \emptyset, 1; Q)\}\,.$$

The 'up-to lifting' allows us to have a relation with only Dirac formal sums (i.e., a single summand with probability 1). □

Lemma 3.15. *If $M \lesssim_{\mathrm{fin}} N$ then $C[M] \lesssim_{\mathrm{fin}} C[N]$, for any context C.*

Proof. Given a a finite-step simulation $\mathcal{R}$ saturated by approximations, we prove that the PE relation

$$\begin{array}{l}\{(C[M], C[N]) \mid M\,\mathcal{R}\,N\}\\ \cup\{((C[M], C[N]), 1; \lambda x.C'[M], 1; \lambda x.C'[N]) \mid M\,\mathcal{R}\,N\}\\ \cup\{((C[M], C[N]), Y, Z) \mid Y\,\mathcal{R}_{(M,N)}\,Z\}\\ \cup\{((M,N), \emptyset, Z) \mid \text{for some } M, N, Z\}\end{array}$$

is a finite-step simulation up-to lifting. □

Corollary 3.16. *On $\Lambda^\bullet_\oplus$-terms, $\approx$ is a congruence, and $\lesssim$ a pre-congruence.*

3.2 Contextual equivalence

We set $M \Downarrow \stackrel{\mathtt{def}}{=} \mathtt{weight}([\![M]\!])$ (the probability of termination).

Definition 3.17. *M and N are in the* contextual preorder*, written $M \leq_{\mathtt{ctx}} N$, (resp. in* contextual equivalence*, written $M =_{\mathtt{ctx}} N$), if $C[M] \Downarrow \leq C[N] \Downarrow$ (resp. $C[M] \Downarrow = C[N] \Downarrow$), for every context C.*

Lemma 3.18 (Completeness)**.** *On $\Lambda^\bullet_\oplus$-terms, $\leq_{\mathtt{ctx}} \subseteq \lesssim$.*

Proof. We prove that the following is a simulation:

$$(\leq_{\mathtt{ctx}}) \cup \{((M,N), [\![C[M]]\!], [\![C[N]]\!]) \mid M \leq_{\mathtt{ctx}} N\}.$$ □

Corollary 3.19 (Full abstraction)**.** *On $\Lambda^{\bullet}_{\oplus}$-terms:*

1. *relations $\leq_{\mathtt{ctx}}$ and $\lesssim$ coincide.*
2. *relations $=_{\mathtt{ctx}}$ and $\approx$ coincide.*

3.3 Up-to techniques

We have pointed out (Example 3.3 and 3.6) that our simulations (and bisimulations) have to be based on formal sums and cannot employ finitary reductions, as in ordinary environmental bisimulations, in order to faithfully represent contextual equivalence. However each of these features is sound and can therefore be used in proof techniques. In this section we show examples of such techniques. These techniques are very limited and we leave for future work the development of more conclusive techniques.

Finitary reductions — the possibility of stopping the evaluation of a term after a few β-reductions — are interesting in enhancements with up-to context (the ability of isolating and removing common contexts in derivative terms) because sometimes such common contexts appear in the middle of a reduction. For applicability, up-to context is usually combined with further up-to techniques that allow us to bring up the common contexts. In the first up-to technique, where the coinduction game still uses formal sums, we combine up-to context with up-to lifting, so to be able to decompose related formal sums into pieces with different common contexts. In the technique, the context closure of the up-to context is only applied onto λ-terms. The closure could probably be made more powerful by applying it also on formal sums, at the price of a more complex proof, but its usefulness is unclear.

In clause (2b) below, and in the remainder of the paper, we use the function $\mathtt{dirac}$ that takes a set of pairs of λ-term (M, N) and returns the pairs of their (Dirac) formal sums $(1; M, 1; N)$.

Definition 3.20. *A PE relation $\mathcal{R}$ is a* finite-step simulation up-to lifting and context *if:*

1. *$M\,\mathcal{R}\,N$ and $M \Longmapsto Y$ imply $Y\,\mathtt{lift}(\,\mathcal{R}_{\,(M,N)})\,[\![N]\!]$;*
2. *$\sum_i p_i; \lambda x.M_i\ \mathcal{R}_{\mathcal{E}}\ \sum_j q_j; \lambda x.N_j$ implies:*
   (a) $\sum_i p_i \leq \sum_j q_j$;
   (b) for all $P, Q \in \mathcal{E}^{\star}$, if $\sum_i p_i; M_i\{^P/_x\} \Longmapsto Y$ then
   $Y\ \mathtt{lift}(\,\mathtt{dirac}(\mathcal{E}^{\star}) \cup \mathcal{R}_{\mathcal{E}}\,)\ \sum_j q_j{\cdot}[\![N_j\{^Q/_x\}]\!]$.

Lemma 3.21. *If $\mathcal{R}$ is a finite-step simulation up-to lifting and context then $\mathcal{R} \subseteq \lesssim_{\mathrm{fin}}$.*

Proof. Let $\mathcal{R}$ be a finite-step simulation up-to lifting and context. We prove that

$$\begin{array}{l}\mathcal{R}' \stackrel{\mathtt{def}}{=} \mathtt{Pairs}(\mathcal{R})\\ \cup\,\{((M,N), Y, Z)\ \mid\ Y'\,\mathcal{R}_{\,(M,N)}\,Z \text{ and } Y \leq_{\mathtt{apx}} Y', \text{for some } Y'\}\\ \cup\,\{((M,N), 1; \lambda x.C[M], 1; \lambda x.C[N])\ \mid\ M\ \mathcal{R}\ N\}\\ \cup\,\{((M,N), \emptyset, Z)\ \mid\ M, N \in \Lambda, Z \in \mathsf{FS}\}\end{array}$$

is a finite-step simulation up-to lifting, from which the result follows by $\mathcal{R} \subseteq \mathcal{R}'$. □

Example 3.22. The up-to lifting and context technique allows us to prove that terms A, B defined in (1) in the Introduction are bisimilar, using the PE relation

$$\{(A, B), ((A, B), [\![A]\!], [\![B]\!])\}\ .$$

Indeed, $[\![A]\!] = 1; A$ and $[\![B]\!] = \frac{1}{2}; \lambda x.x + \frac{1}{2}; \lambda x.\Omega$ and, for any pair of arguments of the form $C[A], C[B]$ used to test the formal sums, we obtain the pair $\frac{1}{2}; C[A], \frac{1}{2}; C[B]$, which is in $\mathtt{dirac}((A, B)^{\star})$. □

In the second up-to technique, reductions are still finitary but now the game is entirely played on terms, without appeal to formal sums. We present the technique in combination with forms of up-to context, up-to lift, up-to distribution, and up-to reduction.

A *term relation* is a relation $\mathcal{T}_{(M,N)}$ on values of $\Lambda^{\bullet}_{\oplus}$ and the index (M, N) is a pair of $\Lambda^{\bullet}_{\oplus}$-terms. The index corresponds, intuitively, to a static environment of an environmental bisimulation. We use the notation $\mathcal{T}^{\star-}_{(M,N)}$ for $\mathcal{T}_{(M,N)} \cup \{(M, N)\}^{\star}$.

A term M deterministically reduces to G (notation: $\stackrel{\mathrm{d}}{\Longrightarrow}$) if $M \Longrightarrow G$ and only the last reduction in the sequence may be derived using rule SUM. We write $M \rhd M'$ if M and M' deterministically reduce to the same formal sum, but M' takes fewer steps. That is, there are G, m, m' with $m \geq m'$ and with $M \stackrel{\mathrm{d}}{\Longrightarrow}_m G$, and $M' \stackrel{\mathrm{d}}{\Longrightarrow}_{m'} G$.

Thus, in Definition 3.23, $\rhd^{\star}\,\mathcal{T}^{\star-}_{(M,N)}$ is the set

$$\{(P, Q)\ \mid\ P \rhd^{\star} P' \text{ for } P' \text{ with } P'(\mathcal{T}_{(M,N)} \cup \{(M, N)\}^{\star})Q\}\ .$$

We then write $F =_{\mathtt{dis}} F'$ if F and F' represent the same probabilistic distribution. In the up-to technique below, $\rhd$ gives us the 'up-to reduction', and $=_{\mathtt{dis}}$ the 'up-to distribution'. We use up-to distribution to manipulate formal sums, which are purely syntactic objects. Finally, $\mathtt{liftd}$ is the lifting on sets of pairs of terms seen as their Dirac formal sums (i.e., $\mathtt{liftd} = \mathtt{lift}(\mathtt{dirac})$), and $\Longrightarrow{=_{\mathtt{dis}}}$ is the composition of the two relations, i.e., $M \stackrel{\mathrm{d}}{\Longrightarrow}{=_{\mathtt{dis}}} F$ if there is F' with $M \stackrel{\mathrm{d}}{\Longrightarrow} F'$ and $F' =_{\mathtt{dis}} F''$.

Definition 3.23. *A term relation $\mathcal{T}_{(M,N)}$ is a* bisimulation up-to context closure, distribution, reduction, and lifting *if*

1. *$[\![M]\!] =_{\mathtt{dis}} \mathtt{dirac}(\mathcal{T}_{(M,N)}) =_{\mathtt{dis}} [\![N]\!]$;*
2. *if $\lambda x.M'\ \mathcal{T}_{(M,N)}\ \lambda x.N'$ then for all P $(M, N)^{\star}$ Q,*

$$M'\{^P/_x\} \stackrel{\mathrm{d}}{\Longrightarrow}{=_{\mathtt{dis}}}\ \mathtt{liftd}(\rhd^{\star}\,\mathcal{T}^{\star-}_{(M,N)})\ {=_{\mathtt{dis}}}{\stackrel{\mathrm{d}}{\Longleftarrow}}\ N'\{^Q/_x\}\ .$$

Lemma 3.24. *If $\mathcal{T}_{(M,N)}$ is a bisimulation up-to context closure, distribution, reduction, and lifting then $(M, N) \in \approx$.*

Example 3.25. The first clause of the the up-to technique in Definition 3.23 is not sound if $\mathcal{T}_{(M,N)}$ is substituted by $\mathcal{T}^{\star-}_{(M,N)}$: in this case, for any pair of values V, W, relation $\mathcal{T}_{(V,W)} \stackrel{\mathtt{def}}{=} \emptyset$ would satisfy the definition, since $[\![V]\!] = 1; V$, $[\![W]\!] = 1; W$ and $1; V\,\mathtt{dirac}((V, W)^{\star})1; W$. □

3.4 Fixed-point combinator example

In the reductions of this example, we write a trivial formal sum $1; M$ as M, so to have reductions between λ-terms. We exploit the up-to technique of Definition 3.23 to prove the equivalence between two fixed-point combinators. One of the combinators is Υ:

$$\begin{array}{c}\Upsilon \stackrel{\mathtt{def}}{=} \lambda y.y(Dy(Dy))\\ \text{where } D \stackrel{\mathtt{def}}{=} \lambda y.\lambda x.y(xx)\ .\end{array}$$

For any term L we have

$$\begin{array}{rcl}\Upsilon L & \longrightarrow & L(DL(DL))\\ \text{and then}\quad DL(DL) & \longrightarrow\!\!\longrightarrow & L(DL(DL))\ .\end{array} \tag{3}$$

The other combinator at any cycle can probabilistically decide whether to behave differently (i.e., as Turing's fixed-point combinator) or to turn for good into into the previous Υ combinator:

$$\begin{array}{rl}&\Theta \stackrel{\mathtt{def}}{=} \Delta\Delta\\ \text{where}\quad &\Delta \stackrel{\mathtt{def}}{=} \lambda x.\lambda y.((y(Dy(Dy))) \oplus (y(xxy)))\ .\end{array}$$

Thus the computation of ΘL will unveil, for a while, some L's while computing as Turing's combinator, and then will continue unveiling L's by computing as Υ. Indeed, for

$$\Theta_1 \stackrel{\mathtt{def}}{=} \lambda y.((y(Dy(Dy))) \oplus (y(\Delta\Delta y)))\,,$$

we have

$$\begin{aligned}\Theta L &\longrightarrow \Theta_1 L \longrightarrow (L(DL(DL))) \oplus (L(\Delta\Delta L)) \\ &\longrightarrow \tfrac{1}{2}; L(DL(DL)) + \tfrac{1}{2}; L(\Delta\Delta L)\ .\end{aligned} \tag{4}$$

We can establish $\Upsilon \approx \Theta$ using the term relation

$$\mathcal{T}_{(\Upsilon,\Theta)} \stackrel{\texttt{def}}{=} \{(\Upsilon, \Theta_1)\}\ .$$

The interesting case is the bisimulation clause for (Υ, Θ_1). Take any $M\ \{(\Upsilon,\Theta)\}^\star\ N$. By (3), we have $\Upsilon M \longrightarrow M(DM(DM))$, whereas by (4), $\Theta N \stackrel{\mathrm{d}}{\Longrightarrow} \frac{1}{2}; N(DN(DN)) + \frac{1}{2}; N(\Delta\Delta N)$. Now we could conclude, up-to context closure, distribution, reduction, and lifting if we can show that the pairs

$$\begin{aligned}&(M(DM(DM)), N(DN(DN))) \\ \text{and}\ &(M(DM(DM)), N(\Theta N))\end{aligned}$$

are in $\rhd^\star\ \mathcal{T}^{\star -}_{(M,N)}$ This holds because: the first pair is in $\{(\Upsilon,\Theta)\}^\star$; for the second pair, by (3) we deduce $DM(DM) \rhd \Upsilon M$, and then we have $M(DM(DM)) \rhd^\star M(\Upsilon M)\ \{(\Upsilon,\Theta)\}^\star\ N(\Theta N)$.

The example also shows the usefulness of static environments (whose terms need not be values) for context closures in 'up-to context' techniques.

4. Probabilistic call-by-value λ-calculus

In call-by-value, the static environments are not anymore sufficient. As in ordinary environmental bisimulations, we need a dynamic environment to record the values produced during the bisimulation game. In ordinary environmental bisimulations such environments are tuples of values. In the probabilistic case formal sums come into the picture. *Environment formal sums* are terms of the form

$$\textstyle\sum_i p_i; \widetilde{V}_i$$

(i.e., sums of weighted tuples) in which all tuples $\widetilde{V}_i$ have the same length and, as for ordinary formal sums, $0 < p_i \leq 1$ for each i and $\sum_i p_i \leq 1$. We call the length of the tuples $\widetilde{V}_i$'s the *length* of the environment formal sum. The tuples $\widetilde{V}_i$ represent the dynamic environment: the knowledge that an observer has accumulated during the bisimulation game. There may be several such elements $\widetilde{V}_i$, reflecting the possible worlds produced by the probabilistic evaluation. During the bisimulation game, the environment formal sum is updated. Viewing the environment formal sum as a matrix, in which $\widetilde{V}_i$ represents the i-row and the elements $(\widetilde{V}_1)_r, (\widetilde{V}_2)_r, \ldots$ (the r-th element of each row) represent the r-th column. A column is a set of values that the various possible worlds have produced at the same step of the bisimulation game. (This explains why the tuples $\widetilde{V}_i$'s of the sum have the same length.)

More precisely, in the bisimulation game at each possible world i a term M_i (constructed from the $\widetilde{V}_i$'s using a context closure discussed below) is evaluated. The evaluation of M_i yields, probabilistically, a multiset of value (as a formal sum). This multiset is empty when all evaluations from M_i diverge; in this case the whole row i disappears, meaning that in the i-th possible world the observer never receives an answer. When the multiset is non-empty, the row i is split into as many possible worlds as the values in the multiset. For instance if the evaluation of M_i produces V with probability $\frac{1}{2}$ and V' with probability $\frac{1}{3}$ then the row $p_i; V_i$ is split into the two rows $\frac{1}{2} \cdot p_i; V_i, V$ and $\frac{1}{3} \cdot p_i; V_i, V'$.

This splitting operation is captured in the following multiplication of an environment formal sum $\sum_{i\in I} p_i; \widetilde{V}_i$ and a tuple of formal sums $F_i = \sum_{j\in J_i} p_{i,j}; V_{i,j}$:

$$\textstyle\sum_{i\in I} p_i; \widetilde{V}_i \cdot F_i \stackrel{\texttt{def}}{=} \sum_{i\in I, j\in J_i} p_i \cdot p_{i,j}; \widetilde{V}_i, V_{i,j}\ .$$

$$\mathbf{Y} = \sum \begin{array}{c} \overbrace{\phantom{|V_{1,1}|V_{1,2}|V_{1,3}|V_{1,4}|}}^{\mathbf{Y}\lfloor_1\ \mathbf{Y}\lfloor_2\ \mathbf{Y}\lfloor_3\ \mathbf{Y}\lfloor_4} \\ \begin{array}{ll} p_1; & |V_{1,1}|V_{1,2}|V_{1,3}|V_{1,4}| \\ p_2; & |V_{2,1}|V_{2,2}|V_{2,3}|V_{2,4}| \\ p_3; & |V_{3,1}|V_{3,2}|V_{3,3}|V_{3,4}| \end{array} \end{array} \begin{array}{l} \}\ \widetilde{V}_1 \\ \}\ \widetilde{V}_2 \\ \}\ \widetilde{V}_3 \end{array}$$

$$\sum \begin{array}{ll} p_1; & |V_{1,1}|V_{1,2}| \\ p_2; & |V_{2,1}|V_{2,2}| \end{array} \cdot [\![I \oplus \Omega]\!] = \sum \begin{array}{ll} \frac{p_1}{2}; & |V_{1,1}|V_{1,2}|I| \\ \frac{p_2}{2}; & |V_{2,1}|V_{2,2}|I| \\ \frac{p_2}{2}; & |V_{2,1}|V_{2,2}|\lambda.\Omega| \end{array}$$

Figure 2. Formal sums as matrices

The view of environment formal sums as matrices is illustrated in Figure 2, for an environment formal sum $\mathbf{Y} \stackrel{\texttt{def}}{=} \sum_{1\leq i\leq 3} p_i; \widetilde{V}_i$ of length 4. The figure also illustrates the extraction of the column r of the formal sum, written $\mathbf{Y}\lfloor_r$, that yields the tuple of values along the same column, and the multiplication of an environment formal sum with formal sums resulting from the semantics of terms, one per row.

We use $\mathbf{Y}, \mathbf{Z}$ to range over environment formal sums, and write $|\mathbf{Y}|$ for the length of $\mathbf{Y}$. We sometimes treat a formal sum as a special case of environment formal sum in which all tuples have length one.

The *dynamic environment* of two formal sums $\sum_{i\in I} p_i; \widetilde{V}_i$ and $\sum_{j\in J} q_j; \widetilde{W}_j$ is the pair of tuples (of tuples) $(\{\widetilde{V}_i\}_{i\in I}, \{\widetilde{W}_j\}_{j\in J})$. In environmental bisimulations, the input for two higher-order functions is constructed as the context closure of their environments. In call-by-value, the environments have both a static and a dynamic component and the inputs are constructed accordingly. Given a static environment (M, N) and a dynamic environment $(\{\widetilde{V}_i\}_i, \{\widetilde{W}_j\}_j)$, their context closure, written

$$(\{M, \widetilde{V}_i\}_i, \{N, \widetilde{W}_j\}_j)^{\hat{\star}}$$

is the set of all pairs of tuples $(\{T_i\}_i, \{U_j\}_j)$ for which there is a context C such that for every i we have $T_i = C[M, \widetilde{V}_i]$, and for every j we have $U_j = C[N, \widetilde{W}_j]$. Thus every T_i is obtained from the same context C by filling its holes with the first element M of the static environment and the dynamic environment $\widetilde{V}_i$. Similarly for U_j, using N, the tuple $\widetilde{W}_j$ and the same context C. Moreover, as we are in call-by-value, C should be a value context, that is, either an abstraction or a hole $[\cdot]_r$ for some $r > 1$. A value context guarantees that each term T_i, U_j is a value.

The operational semantics of call-by-value is defined as in call-by-name, provided that the rule for β-reduction and the evaluation contexts are redefined thus:

$$\textsc{BetaV}\quad \frac{}{(\lambda x.M)V \longrightarrow 1 \cdot M\{V/x\}}$$

$$\text{Evaluation contexts}\qquad C = [\cdot] \ \Big|\ CM \ \Big|\ VC$$

In call-by-value, a probabilistic environmental relation (that we still abbreviate as PE relation) is like for call-by-name, except that formal sums are replaced by environment formal sums. That is, each element of the relation is either of the form (M, N) (a pair of $\Lambda^\bullet_\oplus$-terms) or $\mathbf{Y}\ \mathcal{R}_\mathcal{E}\ \mathbf{Z}$ (two environment formal sums, collecting the dynamic environment, with a static environment).
If $\mathcal{E} = (M, N)$ is a static environment, then $\mathcal{E}_1$ and $\mathcal{E}_2$ denote the projections, i.e., the terms M and N, respectively.

In a PE relation, related environment formal sums are *compatible*, meaning that they have the same length. In the remainder, compatibility of environment formal sums is tacitly assumed.

Definition 4.1 (Environmental bisimulation, call-by-value)**.** *A PE relation is a* (PE) bisimulation *if*

1. $M \mathcal{R} N$ *implies* $[\![M]\!] \mathcal{R}_{(M,N)} [\![N]\!]$;
2. $\sum_i p_i; \widetilde{V}_i \mathcal{R}_{\mathcal{E}} \sum_j q_j; \widetilde{W}_j$ *implies:*
   (a) $\sum_i p_i = \sum_j q_j$;
   (b) *for all* r, *if* $(\widetilde{V}_i)_r = \lambda x.M_i$ *and* $(\widetilde{W}_j)_r = \lambda x.N_j$ *then for all* $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V}_i\}_i, \{\mathcal{E}_2, \widetilde{W}_j\}_j)^{\hat{\star}}$ *we have*
   $$\sum_i p_i; \widetilde{V}_i \cdot [\![M_i\{^{T_i}/_x\}]\!] \mathcal{R}_{\mathcal{E}} \sum_j q_j; \widetilde{W}_j \cdot [\![N_j\{^{U_j}/_x\}]\!] ;$$
   (c) $\sum_i p_i; \widetilde{V}_i \cdot [\![\mathcal{E}_1]\!] \mathcal{R}_{\mathcal{E}} \sum_j q_j; \widetilde{W}_j \cdot [\![\mathcal{E}_2]\!]$.

(PE) bisimilarity, $\approx$ is the union of all PE bisimulations, and the corresponding *similarity* is $\lesssim$.

The structure of the above definition is similar to that of ordinary environmental bisimulations. There are three main differences: first, the appearance of formal sums and of probability measures (notably in clause (2a)); second, the use of an (infinitary) big-step semantics, rather than a small-step, which shows up in the function $[\![\ ]\!]$ in clauses (1) and (2b); thirdly the appearance of a static environment, that is used in the context closure and in clauses (1) and (2c). In clause (2b), the related environment formal sums, viewed as matrices, grow by the addition of a new column resulting, on left-hand side, from the multiplication of each row $p_i; \widetilde{V}_i$ with the formal sum $[\![M_i\{^{T_i}/_x\}]\!]$, and similarly on the right-hand side. Thus the compatibility between related environment formal sums is maintained. Clause (2c) allows to re-evaluate the static environment at any time. This clause and other features are necessary in order to achieve full abstraction in the imperative case (see Section 5.1) but they could be removed in pure call-by-value, following [7].

Theorem 4.2. *1.* $\approx$ *is an equivalence relation;*
2. $\lesssim$ *is a preorder;*
3. $\approx = \lesssim \cap \lesssim^{-1}$.

Example 4.3. Terms M and N in Example 3.3 are equivalent in call-by-name, but not in call-by-value. A bisimulation relating these terms should contain the formal sums $[\![M]\!] = \frac{1}{2}; \lambda.\lambda.\Omega + \frac{1}{2}; \lambda.\Omega$ and $[\![N]\!] = 1; N$, with static environment $\mathcal{E} = (M, N)$ and thus the triple $(\mathcal{E}, \frac{1}{2}; \lambda.\lambda.\Omega, \lambda.\Omega, \frac{1}{2}; N, \lambda.\Omega)$ would be in the relation as well. However, the values in the first column of the dynamic environment can be tested again, by clause (2b) of bisimulation, leading to the triple

$$(\mathcal{E}, \tfrac{1}{2}; \lambda.\lambda.\Omega, \lambda.\Omega, \lambda.\Omega, \tfrac{1}{4}; N, \lambda.\Omega, \lambda.\Omega) ,$$

which does not satisfy the clause on the weights of related formal sums. □

The main results for environmental bisimilarity and similarity in call-by-value (congruence and full abstraction with respect to contextual preorder and equivalence) are as for call-by-name, and the structure of the proofs is similar. The details are however different due to the presence of dynamic environments. As for call-by-name, so in call-by-value to reason about bisimilarity and similarity we need a finite-step simulation, with challenges produced by the finitary big-step approximants. To make sure that the challenges are finite-step, we define *extended environment formal sums*, i.e., terms

$$\sum_i p_i; \widetilde{V}_i; M_i$$

in which the environment formal sum $\sum_i p_i; \widetilde{V}_i$ is extended with an additional column of arbitrary $\Lambda^{\bullet}_{\oplus}$-term (non necessarily values). Intuitively, an element $\widetilde{V}_i; M_i$ indicates that the λ-term M_i has to be run with an observer whose knowledge is $\widetilde{V}_i$. Extended environment formal sums are ranged over by $\mathbf{F}, \mathbf{G}$ and $\mathtt{val}(\mathbf{F})$ is defined analogously to formal sums:

$$\mathtt{val}(\textstyle\sum_i p_i; \widetilde{V}_i; M_i) \stackrel{\mathtt{def}}{=} \sum_{\{i \mid M_i \text{is a value}\}} p_i; \widetilde{V}_i, M_i .$$

Extended environment formal sums allow us to define the multi-step reduction relation from extended environment formal sums to environment formal sums: for $\mathbf{F} = \sum_{i \in I} p_i; \widetilde{V}_i; M_i + \mathbf{G}$, where I is a finite set, we set

$$\frac{M_i \Longmapsto Y_i \quad \text{for every } i}{\mathbf{F} \Longmapsto \sum_{i \in I} p_i; \widetilde{V}_i \cdot Y_i + \mathtt{val}(\mathbf{G})}$$

This intuitively corresponds to the multi-step reduction relation from formal sums to value formal sums. In clause (1) below we see formal sums as special cases of environment formal sums.

Definition 4.4. *A PE relation is a* finite-step simulation *if*

1. $M \mathcal{R} N$ *and* $M \Longmapsto Y$ *imply* $Y \mathcal{R}_{(M,N)} [\![N]\!]$;
2. $\sum_i p_i; \widetilde{V}_i \mathcal{R}_{\mathcal{E}} \sum_j q_j; \widetilde{W}_j$ *implies:*
   (a) $\sum_i p_i \leq \sum_j q_j$;
   (b) *for all* r, *if* $(\widetilde{V}_i)_r = \lambda x.M_i$ *and* $(\widetilde{W}_j)_r = \lambda x.N_j$ *then for all* $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V}_i\}_i, \{\mathcal{E}_2, \widetilde{W}_j\}_j)^{\hat{\star}}$ *we have* $\sum_i p_i; \widetilde{V}_i; M_i\{^{T_i}/_x\} \Longmapsto \mathbf{Y}$ *and* $\mathbf{Y} \mathcal{R}_{\mathcal{E}} \sum_j q_j; \widetilde{W}_j \cdot [\![N_j\{^{U_i}/_x\}]\!]$.

We write $\lesssim_{\mathrm{fin}}$ for the union of all finite-step simulations. Analogously to call-by-name, we use a saturation by approximants and a saturation by suprema to move from a simulation to a finite-step simulation and conversely, and exploit this to prove that similarity and finite-step similarity coincide.

Lemma 4.5. *Relations* $\lesssim$ *and* $\lesssim_{\mathrm{fin}}$ *coincide.*

As in call-by-name, we derive congruence for bisimilarity and similarity by first proving the property for finite-step similarity. We also exploit a combination of two up-to techniques for finite-step simulation, namely up-to lifting and up-to environment. Up-to lifting is defined analogously to call-by-name, using the lifting operation on environment formal sums. Up-to environment intuitively allows us to exchange columns of environment formal sums (when these are viewed as matrices as in Figure 2), and to add new columns. Adding columns is safe because it means enlarging the dynamic environment, thus allowing more tests.

Having these up-to techniques, we derive the result by showing that the contextual closure (which, differently from call-by-name, is now applied both to terms and to the environments of formal sums) of a finite-step simulation saturated by approximants is a finite-step simulation up-to lifting and environment.

Lemma 4.6. *If* $M \lesssim_{\mathrm{fin}} N$ *then for every context* C *we have that* $C[M] \lesssim_{\mathrm{fin}} C[N]$.

The definitions of the contextual preorder and equivalence, $\leq_{\mathtt{ctx}}$ and $=_{\mathtt{ctx}}$, are as for call-by-name.

Theorem 4.7 (Full abstraction)**.** *On* $\Lambda^{\bullet}_{\oplus}$*-terms:*

1. *relations* $\leq_{\mathtt{ctx}}$ *and* $\lesssim$ *coincide;*
2. *relations* $=_{\mathtt{ctx}}$ *and* $\approx$ *coincide.*

We have checked that, in pure call-by-value, environmental bisimulations would be fully abstract also if defined on terms, and with plain tuples of values as dynamic environment (rather than environment formal sums). Indeed also applicative bisimilarity on terms is fully abstract on the same languages [7]. The proof of full abstraction would need 'disentangling' argument for sets of real

numbers in the case of soundness, and separation results from [48] in the case of completeness.

5. Probabilistic imperative λ-calculus

In this section we add imperative features, namely higher-order references (locations), to the call-by-value calculus, along the lines of the languages in [19, 42]. The syntax of terms and values is:

$$
\begin{array}{rll}
M ::= & x & \text{variables} \\
\mid & c & \text{constants} \\
\mid & \lambda x.M & \text{functions} \\
\mid & M_1 M_2 & \text{applications} \\
\mid & l & \text{locations} \\
\mid & (\boldsymbol{\nu}\, x := M_1) M_2 & \text{new location} \\
\mid & !M & \text{dereferencing} \\
\mid & M_1 := M_2 & \text{assignments} \\
\mid & \mathtt{op}(M_1 ... M_n) & \text{primitive operations} \\
\mid & \mathtt{if}\ M_1\ \mathtt{then}\ M_2\ \mathtt{else}\ M_3 & \text{if-then-else} \\
\mid & \#_i(M) & \text{projection} \\
\mid & (M_1 ... M_n) & \text{tuples} \\
\mid & M_1 \oplus M_2 & \text{probabilistic choice}
\end{array}
$$

$$V ::= c \mid \lambda x.M \mid l \mid (V_1 ... V_n)$$

We use s, t to range over stores, i.e., mappings from locations to closed values and l, k over locations. Then $s[l \to V]$ is the update of s (possibly an extension of s if l is not in the domain of s). The locations that occur in a term M are $\mathtt{Loc}(M)$. We assume that the set of primitive operations contains the equality function on constants, and write $\star$ for the unit value (i.e., the nullary tuple).

The language is typed — a simply-typed system with recursive types — to make sure that the values in the summands of a formal sum have the same structure (e.g., they are all abstractions). We allow recursive types to maintain the peculiar possibility of probabilistic languages of having infinite but meaningful computation trees. For readability and lack of space we omit the expected typing rules and, whenever possible, we omit any mention of the types. For instance, in any store update $s[l \to V]$ it is intended that V has the type appropriate for l; in this case we say that the type of V is *consistent* with that of l. In examples, $M_1\ \mathtt{seq}\ M_2$ denotes term $(\lambda.M_2)M_1$, i.e., the execution of M_1 and M_2 in sequence.

Reduction is defined on terms with a store, i.e., configurations of the form $\langle s; M\rangle$; hence such configurations appear also in formal sums. The small-step reduction and the evaluation contexts are defined in Figure 3, where we assume that the semantics of primitive operations is already given by the function $\mathtt{Prim}$. The rules for the semantic mapping, $[\![\ ]\!]$, and the multistep reductions relations, $\Longrightarrow$ and $\Longmapsto$, remain those of Figure 1, with the addition of a store. In all semantic rules, any configuration $\langle s; M\rangle$; is *well-formed*, in that M is closed and all the locations in M and s are in the domain of s. As in the previous calculi, it is easy to check that the semantics of a terms exists and is unique.

Notations and terminology for (environment) formal sums are adapted to the extended syntax in the expected manner. We only recall the multiplication of an environment formal sum $\mathbf{Y} \stackrel{\mathtt{def}}{=} \sum_i p_i; s_i; \widetilde{V}_i$ and formal sums $Y_i \stackrel{\mathtt{def}}{=} \sum_{j\in J_i} p_{i,j}; s_{i,j}; V_{i,j}$:

$$\textstyle\sum_i p_i; \widetilde{V}_i \cdot [\![Y_i]\!] \stackrel{\mathtt{def}}{=} \sum_{i,j\in J_i} p_i \cdot p_{i,j}; s_{i,j}; \widetilde{V}_i, V_{i,j}\ .$$

The context closure of an environment, $(\{M, \widetilde{V}_i\}_i, \{N, \widetilde{W}_j\}_j)^{\widehat{\star}}$, is defined as in the previous section, but now contexts may not contain locations. This constraint, standard in environmental bisimulations for imperative languages, ensures well-formedness of the terms and is not really a limitation because locations may occur in terms of the environments and may thus end up in terms of the context closure.

5.1 Environmental bisimilarity for the probabilistic imperative λ-calculus

The notion of environmental relation is modified to accommodate stores, which are needed to run terms. The elements of an environmental relation are now well-formed pairs of configurations $(\langle s; M\rangle, \langle t; N\rangle)$ or well-formed triples

$$(\mathcal{E}, \textstyle\sum_i p_i; s_i; \widetilde{V}_i, \sum_j q_j; t_j; \widetilde{W}_j)\ .$$

Well-formedness on triples ensures that the store s_i of the possible world i defines all locations that appear in $\widetilde{V}_i$, s_i, and $\mathcal{E}_1$, and similarly for t_j, W_j and $\mathcal{E}_2$. Further, the triples must be *compatible*: the related environment formal sums should have the same length, and should respect the types, that is, corresponding columns of the environment formal sums should contain terms that have the same type.

Definition 5.1 (Environmental bisimulation, imperative)**.** *A PE relation is a* (PE) bisimulation *if*

1. $\langle s; M\rangle\ \mathcal{R}\ \langle t; N\rangle$ *implies* $[\![\langle s; M\rangle]\!]\ \mathtt{lift}(\mathcal{R}_{(M,N)})\ [\![\langle t; N\rangle]\!]$;
2. $\sum_i p_i; s_i; \widetilde{V}_i\ \mathcal{R}_{\mathcal{E}}\ \sum_j q_j; t_j; \widetilde{W}_j$ *implies:*
   (a) $\sum_i p_i = \sum_j q_j$;
   (b) *for all* r, *if* $(\widetilde{V}_i)_r = \lambda x.M_i$ *and* $(\widetilde{W}_j)_r = \lambda x.N_j$ *then for all* $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V}_i\}_i, \{\mathcal{E}_2, \widetilde{W}_j\}_j)^{\widehat{\star}}$ *we have*
   $$\textstyle\sum_i p_i; \widetilde{V}_i \cdot [\![\langle s_i; M_i\{^{T_i}/_x\}\rangle]\!] \quad \mathtt{lift}(\mathcal{R}_{\mathcal{E}})$$
   $$\textstyle\sum_j q_j; \widetilde{W}_j \cdot [\![\langle t_j; N_j\{^{U_j}/_x\}\rangle]\!]\ ;$$
   (c) *for all* r, *if* $(\widetilde{V}_i)_r = l_i$ *and* $(\widetilde{W}_j)_r = k_j$ *then*
   - $\sum_i p_i; s_i; \widetilde{V}_i, s_i(l_i)\ \mathtt{lift}(\mathcal{R}_{\mathcal{E}})\ \sum_j q_j; t_j; \widetilde{W}_j, t_j(k_j)$,
   - *for all* $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V}_i\}_i, \{\mathcal{E}_2, \widetilde{W}_j\}_j)^{\widehat{\star}}$ *we have* $\sum_i p_i; s_i[l_i \to T_i]; \widetilde{V}_i\ \mathcal{R}_{\mathcal{E}}\ \sum_j q_j; t_j[k_j \to U_j]; \widetilde{W}_j$;

   (d) *for any pair* l, k *of fresh locations, and all* $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V}_i\}_i, \{\mathcal{E}_2, \widetilde{W}_j\}_j)^{\widehat{\star}}$ *we have*
   $$\textstyle\sum_i p_i; s_i[l \to T_i]; \widetilde{V}_i, l\ \mathcal{R}_{\mathcal{E}}\ \sum_j q_j; t_j[k \to U_j]; \widetilde{W}_j, k\ ;$$
   (e) *for all* r, *if* $(\widetilde{V}_i)_r = c_i$ *and* $(\widetilde{W}_j)_r = c_j$ *then all constants in the two columns are the same (i.e., there is* c_a *with* $c_i = c_j = c_a$ *for all* i, j*);*
   (f) *for all* r,
   if $(\widetilde{V}_i)_r = (V_{i,1}, ..., V_{i,n})$ *and* $(\widetilde{W}_j)_r = (W_{j,1}, ..., W_{j,n})$
   then $\sum_i p_i; s_i; \widetilde{V}_i, V_{i,1}, ..., V_{i,n} \quad \mathtt{lift}(\mathcal{R}_{\mathcal{E}})$
   $\sum_j q_j; t_j; \widetilde{W}_j, W_{j,1}, ..., W_{j,n}$;
   (g) $\sum_i p_i; \widetilde{V}_i \cdot [\![\langle s_i; \mathcal{E}_1\rangle]\!]\ \mathtt{lift}(\mathcal{R}_{\mathcal{E}})\ \sum_j q_j; \widetilde{W}_j \cdot [\![\langle t_j; \mathcal{E}_2\rangle]\!]$.

As usual, $\approx$ denotes the largest bisimulation, and $\lesssim$ the corresponding largest simulation.

With respect to the definition for pure call-by-value, the definition above has the additional ingredient of the store, and of clauses (2c) and (2d) to deal with the case in which the values are locations: (2c) gives an observer the possibility of reading and writing the store, and (2d) the possibility of extending the store. Clause (2f) adds all elements of a tuple to the dynamic environment. These aspects are similar to those in ordinary environmental bisimulations for imperative languages [21, 42].

Two further aspects, however, are new. First, by clause (2e), related environment formal sums should be *first-order consistent*,

$$\textsc{Beta}\ \frac{}{\langle s;(\lambda x.M)V\rangle \longrightarrow 1;\langle s;M\rangle\{^V/_x\}} \qquad \textsc{Sum}\ \frac{}{\langle s;M_1\oplus M_2\rangle \longrightarrow \tfrac{1}{2};\langle s;M_1\rangle + \tfrac{1}{2};\langle s;M_2\rangle}$$

$$\textsc{Assign}\ \frac{}{\langle s;l:=V\rangle \longrightarrow 1;\langle s[l\to V];\star\rangle} \qquad \textsc{Deref}\ \frac{s(l)=V}{\langle s;!l\rangle \longrightarrow 1;\langle s;V\rangle} \qquad \textsc{New}\ \frac{l \text{ not in the domain of } s}{\langle s;(\boldsymbol{\nu}\, x:=V)M\rangle \longrightarrow 1;\langle s[l\to V];M\{^l/_x\}\rangle}$$

$$\textsc{IfTrue}\ \frac{}{\langle s;\ \texttt{if true then}\ M_1\ \texttt{else}\ M_2\rangle \longrightarrow 1;\langle s;M_1\rangle} \qquad \textsc{IfFalse}\ \frac{}{\langle s;\ \texttt{if false then}\ M_1\ \texttt{else}\ M_2\rangle \longrightarrow 1;\langle s;M_2\rangle}$$

$$\textsc{Proj}\ \frac{}{\langle s;\#_i(\widetilde{V})\rangle \longrightarrow 1;\langle s;(\widetilde{V})_i\rangle} \qquad \textsc{PrimOp}\ \frac{\texttt{Prim}(\texttt{op},\widetilde{c})=c'}{\langle s;\texttt{op}(\widetilde{c})\rangle \longrightarrow 1;\langle s;c'\rangle} \qquad \textsc{Eval}\ \frac{M\longrightarrow \sum_i p_i;M_i \quad C \text{ is an evaluation context}}{C[M]\longrightarrow \sum_i p_i;C[M_i]}$$

$$\text{Evaluation contexts}\quad C:=\ [\cdot]\ \big|\ CM\ \big|\ VC\ \big|\ \texttt{op}(\widetilde{c},C,\widetilde{M})\ \big|\ (\widetilde{V},C,\widetilde{M})\ \big|\ \#_i C$$
$$\big|\ (\boldsymbol{\nu}\, x:=C)M_2\ \big|\ \texttt{if}\ C\ \texttt{then}\ M_1\ \texttt{else}\ M_2\ \big|\ !C\ \big|\ C:=M\ \big|\ l:=C$$

Figure 3. Single-step reduction relation for imperative probabilistic λ-calculus

meaning that corresponding columns of constants should contain exactly one constant. This constraint is a consequence of the equality test on constants in the language. To ensure that first-order consistency is maintained in the bisimulation game, most of the clauses use a lifting construction. Thus, when the evaluation of first-order terms may probabilistically yield different constants, lifting allows us to separate the final possible worlds according to the specific constants obtained. This constraint is further discussed in Example 5.5. A second new aspect is that, since the effect of the evaluation of the terms in the static environment may change depending on the current store, clause (2g) allows us to derive a congruence result for arbitrary terms (not necessarily values), as illustrated in the example below.

Example 5.2. Let $M \stackrel{\texttt{def}}{=} l := 1$ and $N \stackrel{\texttt{def}}{=} \texttt{if}\ !l := 0\ \texttt{then}\ l := 1\ \texttt{else}\ \Omega$. Without clause (2g), $\langle l = 0; M\rangle$ and $\langle l = 0; N\rangle$ are bisimilar, but they are not contextually equivalent: if $C \stackrel{\texttt{def}}{=} [\cdot]\ \underline{\texttt{seq}}\ [\cdot]$, then $\langle l = 0; C[M]\rangle$ terminates whereas $\langle l = 0; C[N]\rangle$ does not. This aspect is determined by the store, probabilities do not really matter. Ordinary environmental bisimulations do not have a static environment, and cannot therefore test repeated runs of given terms that are not values; as a consequence M and N are equated, and bisimulation is not fully substitutive on arbitrary terms (see [42, section 5.2]). □

The following examples are meant to further illustrate and motivate the form and the clauses of our bisimulation. The examples only use boolean and integer locations, and we accordingly assume that all locations in the language are of these types. Higher-order locations would not affect the essence of the examples and would complicate the description of the required bisimulations due to the possibility of extending the store (clause (2d)). (The full abstraction results will not rely on the existence of locations of specific types.) Moreover, since the terms compared always have the same locations, we assume that fresh locations for the extensions of the store (clause (2d)) are the same on both sides.

Example 5.3 shows that in imperative call-by-value, in contrast with pure call-by-value, to achieve full abstraction it is necessary to define bisimulation on formal sums rather than on terms.

Example 5.3. We have explained in Section 1 why the terms

$$H \stackrel{\texttt{def}}{=} (\boldsymbol{\nu}\, x:=0)(\lambda.(M\oplus N)) \qquad K \stackrel{\texttt{def}}{=} (\boldsymbol{\nu}\, x:=0)(\lambda.M\oplus\lambda.N)$$

where $M \stackrel{\texttt{def}}{=} \texttt{if}\ !x=0\ \texttt{then}\ x:=1\ \underline{\texttt{seq}}\ \texttt{true else}\ \Omega$

$N \stackrel{\texttt{def}}{=} \texttt{if}\ !x=0\ \texttt{then}\ x:=1\ \underline{\texttt{seq}}\ \texttt{false else}\ \Omega$

are contextually equivalent, but would be separated by a bisimulation that acted on terms. With our bisimulation, we can prove H and K equal using a relation that contains (H,K) and all triples $((H,K),\mathbf{Y},\mathbf{Z})$ in which the environment formal sums $\mathbf{Y},\mathbf{Z}$ are first-order consistent, have the same total weight and, seeing them as matrices, for every column r of the dynamic environments that is not made of constants one of the following properties holds:

a. there is l such that all terms in $\mathbf{Y}\lfloor_r$ are $\lambda.(M\oplus N)\{^l/_x\}$, whereas all terms in $\mathbf{Z}\lfloor_r$ are either $\lambda.M\{^l/_x\}$ or $\lambda.N\{^l/_x\}$; moreover l does not occur elsewhere in terms of the dynamic environment and its value in the store is 1.
b. there is l such that all terms in $\mathbf{Y}\lfloor_r$ are $\lambda.(M\oplus N)\{^l/_x\}$ whereas $\mathbf{Z}\lfloor_r$ contains both $\lambda.M\{^l/_x\}$ and $\lambda.N\{^l/_x\}$, with equal probability (the sum of the weights of the rows with the former term is the same as the sum with the latter term); moreover l does not occur elsewhere in the dynamic environment and its value in the store is 0.
c. there is l such that all terms in $\mathbf{Y}\lfloor_r$ and $\mathbf{Z}\lfloor_r$ are l; moreover l is set to the same value in all the stores.

In the bisimulation game, the bisimulation clauses (1) and (2g) of Definition 5.1 are handled appealing to item (b). The most interesting case is the bisimulation clause (2b) applied to a column r of functions that satisfy the item (b). The result of the evaluation of such functions (with $\star$ as argument) is that l is set to 1 and then `true` and `false` are returned, with the same probability. Using the lifting construction we can now split the possible worlds in which `true` has been produced and those in which `false` has been produced, yielding two pairs of environment formal sums both of which are in the bisimulation (note that the lifting splits the original column r so that the corresponding column in the two final pairs satisfies item (a) above). □

In the paper we sometimes view environment formal sums as matrices (Figure 2). This however is only for representation convenience: our environments are *tuples of rows* (each row representing a possible world originated by the probabilistic evaluation of terms), rather than *tuples of columns*, that is, tuples of formal sums. The next example shows that if the environments were tuples of formal sums, where formal sums are added to the environment following the evaluation of terms during the bisimulation game, then bisimilarity would not be complete. Intuitively this happens because the histories of different possible worlds would not be anymore separated and could interfere.

Example 5.4. Let

$$A \stackrel{\texttt{def}}{=} (\boldsymbol{\nu}\, y:=0)(L\oplus M) \quad B \stackrel{\texttt{def}}{=} (\boldsymbol{\nu}\, y:=0)(L\oplus N)$$
$$L \stackrel{\texttt{def}}{=} \lambda.!y \quad M \stackrel{\texttt{def}}{=} \lambda.(y:=1\ \underline{\texttt{seq}}\ 2) \quad N \stackrel{\texttt{def}}{=} \lambda.2\,.$$

Terms A and B create a new location and allow the reading capability on it in the subterm L. The writing capability, in contrast, exists only in the subterm M of A. A behaviour from A that could not be mimicked with B is the run of M, where 1 is assigned to the location x, followed by a run of L, where x is read and 1 is emitted (with B, any value produced by L would be 0). This behaviour, however, is impossible, because L and M are in a probabilistic choice and are therefore obtained in two distinct possible worlds, in one of which x can only be read, in the other x can only be written. Moreover, the writing capability alone is irrelevant, because the location is private; hence it can be omitted from M, resulting in the term N that appears in B. Indeed, A and B are contextually equivalent.

However, the 'wrong' behaviour above for A could be reproduced in the bisimulation if the environments were tuples of formal sums (that is, all possible worlds have the same environment, made of formal sums). The formal sum obtained by the evaluation of A, with summand terms L and M, would be stored in the environment and could then be executed several times, with possible interleaving of evaluations of L and M. (The example could be made more complex so to obtain a 'wrong' behaviour from the execution of two different formal sums in the environment, rather than by multiple execution of the same formal sum)

With our bisimulation, we can prove A, B equal using a relation composed by (A, B) and by all triples $((A, B), \mathbf{Y}, \mathbf{Z})$ where $\mathbf{Y} = 1; s; V_1, ..., V_n$ and $\mathbf{Z} = 1; t; W_1, ..., W_n$ are first-order consistent, and where for each column r that does not contain constants one of the following holds:

(a) there is l such that $V_r = L\{^l/_y\} = W_r$; moreover l does not occur elsewhere in the dynamic environment or within a location of the stores, and is set to 0 in both stores;
(b) there is l such that $V_r = M\{^l/_y\}$ and $W_r = N$; and, again, l does not occur elsewhere in the dynamic environment or within a location of the stores; moreover in the store s we have $s(l) \in \{0, 1\}$ whereas in t we have $t(l) = 0$
(c) $V_r = W_r = l$ for some l assigned to the same value in both stores.a

The proof that this relation is a bisimulation crucially exploits the lifting construction. For instance, using (a) and (b) one shows that the semantics of A and B are in the lifting of the relation, and similarly one proceeds when handling clause (2g) of the bisimulation; the lifting also intervenes when the bisimulation clause (2b) is applied to terms for which item (a) above holds. □

The main purpose of the lifting construct in Definition 5.1 of environmental bisimulation is to maintain the first-order consistency of related environment formal sums. One may wonder whether something simpler would suffice, namely avoiding the lifting construct altogether and simply requiring that, whenever two first-order terms are evaluated, the probability of obtaining a given constant is the same on both sides (and thus even avoiding the addition of such values onto the dynamic environments). The example below shows that this would be unsound.

Example 5.5. We compare the terms $A \stackrel{\texttt{def}}{=} (\boldsymbol{\nu}\, x{:=}0)\langle M, N_1\rangle$ and $B \stackrel{\texttt{def}}{=} (\boldsymbol{\nu}\, x{:=}0)\langle M, N_2\rangle$ where

$$M \stackrel{\texttt{def}}{=} \lambda.\ \texttt{if}\ !x = 0\ \texttt{then}\ ((x := 1\ \underline{\texttt{seq}}\ \texttt{true}) \oplus (x := 2\ \underline{\texttt{seq}}\ \texttt{false}))\ \texttt{else}\ \Omega$$
$$N_1 \stackrel{\texttt{def}}{=} \lambda.\ \texttt{if}\ !x = 2\ \texttt{then}\ x := 3\ \underline{\texttt{seq}}\ n\ \texttt{else}\ \Omega$$
$$N_2 \stackrel{\texttt{def}}{=} \lambda.\ \texttt{if}\ (!x = 1 \vee !x = 2)\ \texttt{then}\ x := 3\ \underline{\texttt{seq}}\ (n \oplus \Omega)\ \texttt{else}\ \Omega$$

and n is any integer. The terms A and B produce the values $\langle M, N_1\rangle\{^l/_x\}$ and $\langle M, N_2\rangle\{^l/_x\}$ and l is a location that is accessible only to such values. The definitions of $M\{^l/_x\}$ and $N_i\{^l/_x\}$ (for $i = 1, 2$) use conditionals on the content of l in such a way that the only meaningful manipulations with the values $\langle M\{^l/_x\}, N_i\{^l/_x\}\rangle$ is to evaluate $M\{^l/_x\}$ first, and then, possibly, to evaluate $N_i\{^l/_x\}$. Any other order of evaluation would produce a divergence.

We explain why, intuitively, bisimilarity would equate A and B if, on constants, bisimulation simply checked the probabilities of obtaining each constant (rather than employing the lifting construction). The evaluation of (the body of) $M\{^l/_x\}$ produces `true` or `false`, with the same probability $\frac{1}{2}$ and with l respectively set to 1 and 2. Then the only meaningful observation is the evaluation of the values $N_i\{^l/_x\}$. This means evaluating the formal sums

$$F_1 \stackrel{\texttt{def}}{=} \tfrac{1}{2}; \langle l = 1; N_1\{^l/_x\}\star\rangle + \tfrac{1}{2}; \langle l = 2; N_1\{^l/_x\}\star\rangle$$
$$\text{and } F_2 \stackrel{\texttt{def}}{=} \tfrac{1}{2}; \langle l = 1; N_2\{^l/_x\}\star\rangle + \tfrac{1}{2}; \langle l = 2; N_2\{^l/_x\}\star\rangle\ .$$

The evaluation of F_1 terminates only when $l = 2$, yielding the value formal sum $Y_1 \stackrel{\texttt{def}}{=} \frac{1}{2}; \langle l = 3; n\rangle$. The evaluation of F_2, in contrast, may terminate under both stores, yielding the value formal sum $Y_2 \stackrel{\texttt{def}}{=} \frac{1}{4}; \langle l = 3; n\rangle + \frac{1}{4}; \langle l = 3; n\rangle$. Both in Y_1 and in Y_2 the outcome n has the overall probability $\frac{1}{2}$.

The terms A and B however are not contextually equivalent, because distinguished by a context C that evaluates $M\{^l/_x\}$ and then proceeds with the evaluation $N_i\{^l/_x\}$ *only* when the outcome from $M\{^l/_x\}$ was `true`. Now, $C[A]$ never terminates, whereas $C[B]$ terminates and produces n with probability $\frac{1}{4}$.

Our environmental bisimulation distinguishes A from B because we separately analyze the possible worlds in which the evaluation of $M\{^l/_x\}$ has produced `true` and the possible worlds in which the evaluation has produced `false`, somehow mimicking the effect of the context C above. □

Yet another possibility for avoiding the lifting construct of the Definition 5.1 of bisimulation might have been to drop the requirement of first-order consistency, thus allowing environment formal sums in which a first-order column may contain different constants. Thus constants would be added to the dynamic environment as any other type of value, and one would simply check that, at any time, the weights for the occurrences of a given constant in related columns are the same; formally, replacing clause (2e) with:

(2e′) for every column r and every constant c, $\sum_{\{i \mid (\widetilde{V}_i)_r = c\}} p_i = \sum_{\{j \mid (\widetilde{W}_j)_r = c\}} q_j$.

This possibility is unsound too, which can be shown using a minor variant of the previous example.

The basic properties and definitions for environmental bisimulations in pure call-by-value remain valid, with the due adjustments. In some cases, however, some subtleties arise. We only report the definition of finite-step simulation and the full abstraction result. In the finite-step simulation, clauses (2b) and (2g) are modified so to make sure that only a finite number of reductions are performed on the challenger side.

Definition 5.6. *A PE relation is a* finite-step simulation *if it satisfies the same clauses (2a), (2c), (2d), (2e) and (2f) of (the simulation version of) Definition 5.1; and, in place of clauses (1), (2b) and (2g) we have:*

1. $\langle s; M\rangle\ \mathcal{R}\ \langle t; N\rangle$ *and* $\langle s; M\rangle \Longmapsto Y$ *imply* $Y\ \texttt{lift}(\mathcal{R}_{(M,N)})\ [\![\langle t; N\rangle]\!]$;
2. $\sum_i p_i; s_i; \widetilde{V}_i\ \mathcal{R}_{\mathcal{E}}\ \sum_j q_j; t_j; \widetilde{W}_j$ *implies:*
   (b) for all r, if $(\widetilde{V}_i)_r = \lambda x.M_i$ and $(\widetilde{W}_j)_r = \lambda x.N_j$ then for all $(\{T_i\}_i, \{U_j\}_j) \in (\{\mathcal{E}_1, \widetilde{V}_i\}_i, \{\mathcal{E}_2, \widetilde{W}_j\}_j)^{\star}$

if $\sum_i p_i; s_i; \widetilde{V}_i; M_i\{^{T_i}/_x\} \Longmapsto Y$ *then*
$Y\ \mathtt{lift}(\mathcal{R}_\mathcal{E})\ \sum_j q_j; \widetilde{W}_j \cdot [\![\langle t_j; N_j\{^{U_j}/_x\}\rangle]\!]$;

(g) *if* $\sum_i p_i; s_i; \widetilde{V}_i; \mathcal{E}_1 \Longmapsto Y$ *then*
$Y\ \mathtt{lift}(\mathcal{R}_\mathcal{E})\ \sum_j q_j; \widetilde{W}_j \cdot [\![\langle t_j; \mathcal{E}_2\rangle]\!]$.

Definition 5.7. *M and N are in the* contextual preorder*, written $M \leq_{\mathtt{ctx}} N$, (resp.* contextually equivalent*, written $M =_{\mathtt{ctx}} N$), if, for any store s and context C such that $\langle s; C[M]\rangle$ and $\langle s; C[N]\rangle$ are well-formed, $\langle s; C[M]\rangle\Downarrow \leq \langle s; C[N]\rangle\Downarrow$ (resp. $\langle s; C[M]\rangle\Downarrow = \langle s; C[N]\rangle\Downarrow$).*

Theorem 5.8 (Full abstraction)**.** *Let $\{\widetilde{l}\}$ be the set of locations that appear in M or N. We have, for any $\widetilde{V}$ whose type is consistent with that of $\widetilde{l}$:*

- *$M \leq_{\mathtt{ctx}} N$ if and only if $\langle \widetilde{l} = \widetilde{V}; (\widetilde{l}, M)\rangle \lesssim \langle \widetilde{l} = \widetilde{V}; (\widetilde{l}, N)\rangle$;*
- *$M =_{\mathtt{ctx}} N$ if and only if $\langle \widetilde{l} = \widetilde{V}; (\widetilde{l}, M)\rangle \approx \langle \widetilde{l} = \widetilde{V}; (\widetilde{l}, N)\rangle$.*

6. Additional related works

Most relevant related work has been discussed in the introduction. We add some more technical remarks here. The semantics of pure probabilistic λ-calculus is presented in [8] as a mapping onto distributions. It is straightforward to check that the formal sums resulting from our semantics yield the same distributions.

The probabilistic λ-calculus, i.e., a λ-calculus with endowed with binary, fair, probabilistic choice, was first presented in [8]. In [9] and [7] probabilistic applicative bisimulations for pure call-by-name and call-by-value λ-calculi are shown to be congruence, using Howe's method coupled with a disentangling technique. Completeness however only holds in call-by-value, while it fails in call-by-name. In call-by-name, completeness is obtained using coupled logical bisimulation, a probabilistic version of the logical bisimilarity for deterministic languages [41]. Drawbacks of all forms of logical bisimilarity are a non-monotone functional and a confinement to pure λ-calculi. Further, up-to techniques may be difficult in logical bisimilarity. For instance, Example 3.22 cannot be proved with the techniques in [9]: the equality fails for applicative bisimilarity, and the up-to context technique provided for logical bisimilarity is not powerful enough (the paper shows a similar example, akin to our Example 3.3, where however the functions employed immediately throw away their input, and this is essential for the proof).

A further difference between applicative bisimulations and environmental bisimulations shows up when one considers the corresponding *simulations*. Even in cases where applicative bisimilarity is fully abstract for contextual equivalence, the corresponding simulation may not be fully abstract for the contextual preorder. Pure call-by-value is such an example [7**?**]. In contrast, in all calculi in the paper, full abstraction for environmental bisimilarity carries over to the corresponding simulation, with a similar proof.

An alternative bisimulation for enriched calculi is *normal form* (or *open*) bisimulation [15, 23, 40, 44]. This is is complete (with respect to contextual equivalence) only in certain extensions of the λ-calculus (e.g., call-by-value with *both* state *and* callcc), and would be incomplete in other languages (such as λ-calculus without state or/and callcc, and languages with constants or types).

Another approach to contextual equivalence in higher-order languages is via logical relations (see, e.g., [28, Chapter 8] and [34]). This technique has been applied to probabilistic typed higher-order languages by Bizjak and Birkedal [4]. They use both step-indexing and biortogonality; this introduces some universal quantification (e.g., on evaluation contexts) which makes it difficult to prove examples such as 5.3. An attempt at combining bisimulations and logical relations in the non-probabilistic case is [**?**].

In denotational semantics, fully abstract models for probabilistic PCF have been studied in [**?**] using domain theory and adding statistical termination testers, and in [**?**] using probabilistic coherence spaces. [**?**] provides a fully abstract game semantics for probabilistic Algol, using a quotienting step.

7. Conclusions

We have investigated environmental bisimulations in sequential higher-order languages, considering pure λ-calculi, both call-by-name and call-by-value, and an extension with references.

While we have tried to respect the general schema of environmental bisimulations, our definitions and results present noticeable technical differences. Some differences, such as the appeal to formal sums, are specific to probabilities. Other differences, however, may be seen as insights into environmental bisimulations that were suggested by the study of probabilities. An example is the distinction between a static and a dynamic environment, which reflects the copying facilities of the language on the terms of the environment. This distinction yields sharper congruence results, which shows up well in the imperative λ-calculus: with ordinary environmental bisimulations, bisimilarity is fully substitutive only for values — for general terms substitutivity holds only for evaluation contexts (see Example 5.2). The example in Section 3.4 shows that static environments can also be useful in context closures of 'up-to context' techniques.

To understand environmental bisimulations for call-by-value calculi, we have found important the study of the imperative extension. Only in the richer language do various aspects of our definitions find a justification: the use of formal sums (Example 5.3); dynamic environments as formal sums of tuples of values, as opposite to, e.g., tuples of formal sums (Example 5.4); the lifting construct to handle first-order values (Example 5.5). The pure call-by-value calculus has allowed us to present the concepts in a simpler setting, as a stepping stone towards the imperative extension, but seems a rather peculiar language, one in which a number of variations of the definitions collapse.

The dynamic environments are used only for the call-by-value calculi. In general, the form of the bisimulation clauses depends on the features of the calculus. It would be interesting to investigate abstract formulation of bisimulation, of which the concrete definitions presented in this paper would be instances. Possible bases for such a framework could be coalgebras [38] or bigraphs [27].

Enhancements of the bisimulation proof method, as up-to techniques, are useful for environmental bisimulations [42]. In the paper we have explored some basic enhancements, mainly in call-by-name, including new forms of up-to specific to probabilities such as 'up-to lifting'. The study of powerful enhancements goes beyond the scopes of the paper. It could be pursued in a number of directions, for instance investigating other forms of up-to and strengthening the 'up-to context' enhancements in the paper.

Another interesting direction for future work is the addition of concurrency. A major consequence of this could be the move to semantics that combine probabilities with non-determinism.

Acknowledgments

We have benefited from discussions with Raphaëlle Crubillé and Ugo Dal Lago, and we thank the anonymous referees. The work has been partially supported by the MIUR-PRIN project 'CINA', and the ANR project 12IS02001 'PACE'.

References

[1] M. Abadi and A. D. Gordon. A bisimulation method for cryptographic protocols. In C. Hankin, editor, *Proc. ESOP'98*, volume 1381 of *Lecture Notes in Computer Science*, pages 12–26. Springer, 1998.

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

[3] D. Biernacki and S. Lenglet. Environmental bisimulations for delimited-control operators. In *APLAS*, volume 8301 of *Lecture Notes in Computer Science*, pages 333–348. Springer, 2013.

[4] A. Bizjak and L. Birkedal. Step-indexed logical relations for probability. In *FoSSaCS 2015*, pages 279–294, 2015.

[5] M. Boreale and D. Sangiorgi. Bisimulation in name-passing calculi without matching. In *Proc. 13th LICS Conf.* IEEE Computer Society Press, 1998.

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

[7] R. Crubillé and U. Dal Lago. On probabilistic applicative bisimulation and call-by-value λ-calculi. In *ESOP 2014*, pages 209–228, 2014.

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

[9] U. Dal Lago, D. Sangiorgi, and M. Alberti. On coinductive equivalences for higher-order probabilistic functional programs. In *POPL '14*, pages 297–308, 2014.

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

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

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

[13] 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.

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

[15] R. Jagadeesan, C. Pitcher, and J. Riely. Open bisimulation for aspects. *T. Aspect-Oriented Software Development*, 5:72–132, 2009.

[16] A. Jeffrey and J. Rathke. Towards a theory of bisimulation for local names. In *LICS'99*, pages 56–66, 1999.

[17] P. Johann, A. Simpson, and J. Voigtländer. A generic operational metatheory for algebraic effects. In *LICS 2010*, pages 209–218, 2010.

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

[19] V. Koutavas and M. Wand. Small bisimulations for reasoning about higher-order imperative programs. In *Proc. POPL'06*, pages 141–152, 2006.

[20] V. Koutavas and M. Wand. Bisimulations for untyped imperative objects. In *ESOP'06*, pages 146–161, 2006.

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

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

[23] S. B. Lassen and P. B. Levy. Typed normal form bisimulation. In *CSL 2007*, volume 4646 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2007.

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

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

[26] J. Maraist, M. Odersky, D. N. Turner, and P. Wadler. Call-by-name, call-by-value, call-by-need and the linear lambda calculus. *Theor. Comput. Sci.*, 228(1-2):175–210, 1999.

[27] R. Milner. Pure bigraphs: Structure and dynamics. *Inf. Comput.*, 204 (1):60–122, 2006.

[28] J. C. Mitchell. *Foundations for Programming Languages*. MIT Press, 1996.

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

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

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

[32] A. Piérard and E. Sumii. Sound bisimulations for higher-order distributed process calculus. In *FOSSACS 2011*, pages 123–137, 2011.

[33] A. Piérard and E. Sumii. A higher-order distributed calculus with name creation. In *LICS 2012*, pages 531–540, 2012.

[34] A. Pitts. Typed operational reasoning. In B. C. Pierce, editor, *Advanced Topics in Types and Programming Languages*, chapter 7, pages 245–289. MIT Press, 2005.

[35] 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.

[36] D. Pous and D. Sangiorgi. Enhancements of the bisimulation proof method. In D. Sangiorgi and J. Rutten, editors, *Advanced Topics in Bisimulation and Coinduction*. Cambridge University Press, 2012.

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

[38] J. Rutten and B. Jacobs. (co)algebras and (co)induction. In D. Sangiorgi and J. Rutten, editors, *Advanced Topics in Bisimulation and Coinduction*. Cambridge University Press, 2012.

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

[40] D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. *Inf. and Comp.*, 111(1):120–153, 1994.

[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] N. Sato and E. Sumii. The higher-order, call-by-value applied pi-calculus. In *APLAS 2009*, pages 311–326, 2009.

[44] K. Støvring and S. B. Lassen. A complete, co-inductive syntactic theory of sequential control and state. In *Proc. POPL'07*, pages 161–172, 2007. To appear.

[45] E. Sumii and B. C. Pierce. A bisimulation for dynamic sealing. *Theor. Comput. Sci.*, 375(1-3):169–192, 2007. A preliminary version in *Proc. POPL'04*, 2004.

[46] E. Sumii and B. C. Pierce. A bisimulation for type abstraction and recursion. *J. ACM*, 54(5), 2007. A preliminary version in Proc. POPL'05, 2005.

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

[48] F. van Breugel, M. W. Mislove, J. Ouaknine, and J. Worrell. Domain theory, testing and simulation for labelled markov processes. *Theor. Comput. Sci.*, 333(1-2):171–197, 2005.