Effectful applicative bisimilarity

Dal Lago, Ugo; Gavazzo, Francesco ; Levy, Paul Blain

DOI:
10.1109/LICS.2017.8005117

Effectful Applicative Bisimilarity: Monads, Relators, and Howe's Method (Long Version)

Ugo Dal Lago Francesco Gavazzo Paul Blain Levy

Abstract

We study Abramsky's applicative bisimilarity abstractly, in the context of call-by-value λ-calculi with algebraic effects. We first of all endow a computational λ-calculus with a monadic operational semantics. We then show how the theory of relators provides precisely what is needed to generalise applicative bisimilarity to such a calculus, and to single out those monads and relators for which applicative bisimilarity is a congruence, thus a sound methodology for program equivalence. This is done by studying Howe's method in the abstract.

1 Introduction

Program equivalence is one of the central notions in the theory of programming languages, and giving satisfactory definitions and methodologies for it is a challenging problem, for example when dealing with higher-order languages. The problem has been approached, since the birth of the discipline, in many different ways. One can define program equivalence through denotational semantics, thus relying on a model and stipulating two programs to be equivalent if and only if they are interpreted by the same denotation. If the calculus at hand is equipped with a notion of *observation*, typically given through some forms of operational semantics, one could proceed following the route traced by Morris, and define programs to be *contextual* equivalent when they *behave* the same *in every* context.

Both these approaches have their drawbacks, the first one relying on the existence of a (not too coarse) denotational model, the latter quantifying over all contexts, and thus making concrete proofs of equivalence hard. Handier methodologies for proving programs equivalent have been introduced along the years based on logical relations and applicative bisimilarity. Logical relations were originally devised for typed, normalising languages, but later generalised to more expressive formalisms, e.g., through step-indexing [3] and biorthogonality [6]. Starting from Abramsky's pioneering work on applicative bisimilarity [1], coinduction has also been proved to be a useful methodology for program equivalence, and has been applied to a variety of calculi and language features.

The scenario just described also holds when the underlying calculus is not pure, but effectful. There have been many attempts to study effectful λ-calculi [36, 32] by way of denotational semantics [21, 14, 12], logical relations [7], and applicative bisimilarity [27, 10, 9]. But while the denotational and logical relation semantics of effectful calculi have been studied in the abstract [18, 20], the same cannot be said about applicative bisimilarity and related coinductive techniques. There is a growing body of literature on applicative bisimilarity for calculi with, e.g., nondeterministic [27], and probabilistic effects [10], but each notion of an effect has been studied independently, often getting different results. Distinct proofs of congruence for applicative bisimilarity, even if done through a common methodology, namely the so-called Howe's method [19], do not at all have the same difficulty in each of the cases cited above. As an example, the proof of the so-called Key Lemma relies on duality results from linear programming [40] when done for probabilistic effects, contrarily to the apparently similar case of nondeterministic effects, whose

$$
\begin{aligned}
W &\to V \oplus COMP(V, W) \\
Z &\to T\,\underline{1} \\
T\,\underline{n} &\to (R\,\underline{n}) \oplus (T\,\underline{n+1}) \\
R\,\underline{0} &\to \lambda x.x \\
R\,\underline{n+1} &\to COMP(R\,\underline{n}, V)
\end{aligned}
$$

Figure 1: Two Probabilistic Programs.

logical complexity is comparable to that for the plain, deterministic λ-calculus [34, 27]. Finally, as the third author observed in his work with Koutavas and Sumii [24], applicative bisimilarity is fragile to the presence of certain effects, like local states or dynamically created exceptions: in these cases, a sort of information hiding is possible which makes applicative bisimilarity simply too weak, and thus unsound for contextual equivalence.

The observations above naturally lead to some questions. Is there any way to factor out the common part of the congruence proof for applicative bisimilarity in the cases above? Where do the limits on the correctness of applicative bisimilarity lie, in presence of effects? The authors strongly believe that the field of coinductive techniques for higher-order program equivalence should be better understood *in the abstract*, this way providing some answers to the questions above, given that generic accounts for effectful λ-calculi abound in the literature [32, 36].

This paper represents a first step towards answering the questions above. We first of all introduce a computational λ-calculus in which general algebraic effects can be represented, and give a monadic operational semantics for it, showing how the latter coincides with the expected one in many distinct concrete examples. We then show how applicative bisimilarity can be defined for any instance of such a monadic λ-calculus, based on the notion of a relator, which allows to account for the possible ways a relation on a set X can be turned into one for TX, where T is a monad. We then single out a set of axioms for monads and relators which allow us to follow Howe's proof of congruence for applicative bisimilarity *in the abstract*. Noticeably, these axioms are satisfied in all the example algebraic effects we consider. The proof of it allows us to understand the deep reasons *why*, say, different instances of Howe's method in the literature seem to have different complexities.

2 On Coinduction and Effectful λ-Calculi

In this section, we illustrate how coinduction can be useful when proving the equivalence of programs written in higher-order effectful calculi.

Let us start with a simple example of two supposedly equivalent probabilistic functional programs, W and Z, given in Figure 1. (The expression $COMP(M, N)$ stands for the term $\lambda y.M(Ny)$, and $\oplus$ is a binary operation for fair probabilistic choice.) Both W and Z behave like the n-th composition of a function V with itself with probability $\frac{1}{2^n}$, for every n. But how could we even *define* the equivalence of such effectful programs? A natural answer consists in following Morris [33], and stipulate that two programs are contextually equivalent if they behave the same when put in any context, where the observable behaviour of a term can be taken, e.g., as its probability of convergence. Proving two terms to be contextually equivalent can be quite hard, given the universal quantification over all contexts on which contextual equivalence is based.

Applicative bisimilarity is an alternative definition of program equivalence, in which λ-terms are seen as computational objects interacting with their environment by exposing their behaviour, and by taking arguments as input. Applicative bisimilarity has been generalised to effectful λ-calculi of various kinds, and in particular to untyped probabilistic λ-calculi [10], and it is known to

$$
\begin{aligned}
W^{\mathsf{raise}} &\to (V \oplus \mathsf{raise}_e) \oplus \mathit{COMP}(V, W^{\mathsf{raise}})\\
Z^{\mathsf{raise}} &\to T\,\underline{1}\\
T\,\underline{n} &\to ((R\,\underline{n}) \oplus \mathsf{raise}_e) \oplus (T\,\underline{n+1})\\
R\,\underline{0} &\to \lambda x.x\\
R\,\underline{n+1} &\to \mathit{COMP}(R\,\underline{n}, V)
\end{aligned}
$$

Figure 2: Two Probabilistic Programs Throwing Exceptions.

be not only a congruence (thus *sound* for contextual equivalence) but also *fully abstract*, at least for call-by-value evaluation [9]. Indeed, applicative bisimilarity can be applied to the example terms in Figure 1, which can this way be proved contextual equivalent.

The proof of soundness of applicative bisimilarity in presence of probabilistic effects is significantly more complicated than the original one, although both can be done by following the so-called Howe's method [19]. More specifically, the proof that the Howe extension of similarity is a simulation relies on duality from linear programming (through the Max Flow Min Cut Theorem) when done in presence of probabilistic effects, something that is not required in the plain, deterministic setting, nor in presence of *nondeterministic* choice.

Modern functional programming languages, however, can be "effectful" in quite complex ways. As an example, programs might be allowed not only to evolve probabilistically, but also to have an internal state, to throw exceptions, or to perform some input-output operations. Consider, as another simple example, the programs in Figure 2, a variation on the programs from Figure 1 where we allow programs to additionally raise an exception e by way of the raise_e command. Intuitively, W^{raise} and Z^{raise} behave like W and Z, respectively, but they both raise an exception with a certain probability.

While applicative similarity in presence of catchable exceptions is well-known to be unsound [24], the mere presence of the raise_e command does not seem to cause any significant problem. The literature, however, does not offer any result about whether *combining* two or more notions of computational effect for which bisimilarity is known to work well, should be problematic or not. An *abstract* theory accounting for how congruence proofs can be carried out in effectful calculi is simply lacking.

Even if staying within the scope of Howe's method, it seems that each effect between those analysed in the literature is handled by way of some *ad-hoc* notion of bisimulation. As an example, nondeterministic extensions of the λ-calculus can be dealt with by looking at terms as a labelled transition system, while probabilistic extensions of the λ-calculus require a different definition akin to Larsen and Skou's probabilistic bisimulation [10]. What kind of transition do we need when, e.g., dealing with the example from Figure 2? In other words, an abstract theory of effectful applicative bisimilarity would be beneficial from a purely definitional viewpoint, too.

What could come to the rescue here is the analysis of effects and bisimulation which has been carried out in the field of coalgebra [39]. In particular, we here exploit the theory of relators, also known as lax extensions [5, 41].

3 Domains and Monads: Some Preliminaries

In this section, we recall some basic definitions and results on complete partial orders, categories, and monads. All will be central in the rest of this paper. Due to space constraints, there is no hope to be comprehensive. We refer to the many introductory textbooks on partial order theory [13] or category theory [31] for more details.

3.1 Domains and Continuous Σ-algebras

Here we recall some basic notions and results on domains that we will extensively use in this work. The main purpose of this section is to introduce the notation and terminology we will use in the rest of this paper. We address the reader to e.g. [2] for a deeper treatment of the subject.

Recall that a poset is a set equipped with a reflexive, transitive and antisymmetric relation.

Definition 1. *Given a poset $\mathbb{D} = (D, \sqsubseteq_D)$, an ω-chain in D is an infinite sequence $(x_n)_{n<\omega}$ of elements of D such that $x_n \sqsubseteq_D x_{n+1}$, for any $n \geq 0$.*

Definition 2. *A poset $\mathbb{D} = (D, \sqsubseteq_D)$ is an ω-*complete partial order*, ω**CPO** for short, if any ω-chain $(x_n)_{n<\omega}$ in D has least upper bound (lub) in D. A poset $\mathbb{D} = (D, \sqsubseteq)$ is an ω-*complete pointed partial order*, ω**CPPO** for short, if it is an ω**CPO** with a least element $\perp_D$.*

For an ω**CPPO** $\mathbb{D} = (D, \sqsubseteq_D, \perp_D)$ we will often omit subscripts, thus writing $\sqsubseteq$ and $\perp$ for $\sqsubseteq_D$ and $\perp_D$, respectively. Given an ω-chain $(x_n)_{n<\omega}$ we will denote its least upper bound by $\bigsqcup_D \{x_n \mid n < \omega\}$. Oftentimes, following the above convention, we will shorten the latter notation to $\bigsqcup_{n<\omega} x_n$.

Notice that an ω-chain being a sequence, its elements need not be distinct. In particular, we say that the chain is *stationary* if there exists $N < \omega$ such that $x_{N+n} = x_N$, for any $n < \omega$.

We will often use the following basics result, stating that if we discard any finite number of elements at the beginning of a chain, we do not affect its set of upper bounds (and its lub).

Lemma 1. *For any ω-chain $(x_n)_{n<\omega}$ and $N < \omega$ the following equality holds:*

$$\bigsqcup_{n<\omega} x_n = \bigsqcup_{n<\omega} x_{N+n}.$$

Definition 3. *Let $\mathbb{D} = (D, \sqsubseteq_D, \perp_D), \mathbb{E} = (E, \sqsubseteq_E, \perp_E)$ be ω**CPPO**s. We say that a function $f : D \to E$ is* monotone *if for all elements x, y in D, $x \sqsubseteq_D y$ implies $f(x) \sqsubseteq_E f(y)$. We say it is* continuous *if it is monotone and preserves lubs. That is, for any ω-chain $(x_n)_{n<\omega}$ in D we have:*

$$f(\bigsqcup_D \{x_n \mid n < \omega\}) = \bigsqcup_E \{f(x_n) \mid n < \omega\}.$$

Finally, we say it is strict *if $f(\perp_D) = \perp_E$.*

Remark 1. *(see [2]) It can be easily shown that if a function is continuous then it is also monotone. However, it should be noticed that to prove that a function $f : D \to E$ is continuous, it is necessary to prove that $(f(x_n))_{n<\omega}$ forms an ω-chain in E, for any ω-chain $(x_n)_{n<\omega}$ in D. That is equivalent to prove monotonicity of f.*

We denote the set of continuous functions from D to E by $D \overset{c}{\to} E$ and write $f : D \overset{c}{\to} E$ for $f \in D \overset{c}{\to} E$.

We will implicitly use the fact that continuous endofunctions on ω**CPPO**s are guaranteed to have least fixed points: given a continuous endofunction $f : D \to D$ on an ω**CPPO** D, there exists an element $\mu f \in D$ such that $f(\mu f) = \mu f$, and for any $x \in D$, if $f(x) = x$ then $\mu f \sqsubseteq x$. Throughout this work, we will use the notation μf and νf to denote least and greatest fixed point of a function f, respectively.

ω**CPPO**s and continuous functions form a category, $\omega\mathbb{CPPO}$, which has a cartesian closed structure. In particular, the cartesian product (of the underlying sets) of ω**CPPO**s is an ω**CPPO** when endowed with the pointwise order (with lubs and bottom element computed pointwise). Similarly, the set of continuous functions spaces between ω**CPPO**s is an ω**CPPO** when endowed with the pointwise order (again, with lubs and bottom element computed pointwise)[1]. Notice that the function space $D \overset{c}{\to} E$ between ω**CPPO**s D and E is an ω**CPPO** even if D does not have a least element (i.e. if D is an ω**CPO**). As a consequence, since we can regard any set X

[1] Let $\mathbb{D}, \mathbb{E}$ be ω**CPPO**s define:

as an ω**CPO** ordered by the identity relation $=_X$ on X^2, the set $X \xrightarrow{c} D = X \to D$ is always an ω**CPPO**, for any ω**CPPO** D. The following well- known result will be useful in several examples.

Lemma 2. *Let $\mathbb{C}, \mathbb{D}, \mathbb{E}$ be ω**CPPO**s. A function $f : C \times D \to E$ is:*
1. *Monotone, if it is monotone in each arguments separately.*
2. *Continuous, if it is continuous in each arguments separately.*

Following [36, 37], we consider operations (like $\oplus$ or raise_e in the examples from Section 2) form a given signature as sources of effects. Semantically, dealing with operation symbols requires the introduction of appropriate algebraic structures interpreting such operation symbols as suitable functions. Combining the algebraic and the order theoretic structures just described, leads to consider algebras carrying a domain structure (ω**CPPO**, in this paper), such that *all* function symbols are interpreted as continuous functions. The formal notion capturing all these desiderata is the one of a continuous Σ-algebra [15].

Recall that a signature $\Sigma = (\mathcal{F}, \alpha)$ consists of a set $\mathcal{F}$ of operation symbols and a map $\alpha : \mathcal{F} \to \mathbb{N}$, assigning to each operation symbol a (finite) arity. A Σ-algebra $(A, (\cdot)^A)$ is given by a carrier set A and an interpretation $(\cdot)^A$ of the operation symbols, in the sense that for $\sigma \in \mathcal{F}$, σ^A is a map from $A^{\alpha(\sigma)}$ to A. We will write $\sigma \in \Sigma$ for $\Sigma = (\mathcal{F}, \alpha)$ and $\sigma \in \mathcal{F}$.

Definition 4. *Given a signature Σ, a continuous Σ-algebra is an ω**CPPO** $\mathbb{D} = (D, \sqsubseteq, \bot)$ such that for any function symbol σ in Σ there is an associated continuous function $\sigma^D : D^{\alpha(\sigma)} \to D$.*

Remark 2. *Observe that for a function symbol $\sigma \in \Sigma$, we do not require σ^D to be strict.*

Before looking at monads, we now give various examples of concrete algebras which can be given the structure of a continuous Σ-algebra for certain signatures. This testifies the applicability of our theory to a relatively wide range of effects.

Example 1. *Let X be a set: the following are examples of ω**CPPO**.*
- *The flat lifting $X_\bot$ of X, defined as $X + \{\bot\}$, ordered as follows: $x \sqsubseteq y$ iff $x = \bot$ or $x = y$.*
- *The set $(X + E)_\bot$ (think to E as a set of exceptions), ordered as in the previous example. We can consider the signature $\Sigma = \{\mathsf{raise}_e \mid e \in E\}$, where each operation symbol raise_e is interpreted as the constant $\mathsf{in}_l(\mathsf{in}_r(e))$.*
- *The powerset $\mathcal{P}X$, ordered by inclusion. The least upper bound of a chain of sets is their union, whereas the bottom is the empty set. We can consider the signature $\Sigma = \{\oplus\}$ containing a binary operation symbol for nondeterministic choice. The latter can be interpreted as (binary) union, which is indeed continuous.*
- *The set of subdistributions $\mathcal{D}X = \{\mu : X \to [0,1] \mid \mathsf{supp}(\mu)$ countable, $\sum_{x \in X} \mu(x) \leq 1\}$ over X, ordered pointwise: $\mu \sqsubseteq \nu$ iff $\forall x \in X.\ \mu(x) \leq \nu(x)$. Note that requiring the support of μ to be countable is equivalent to requiring the existence of $\sum_{x \in X} \mu(x)$. The ω**CPPO** structure is pointwise induced by the one of $[0,1]$ with the natural ordering. The least element is the always zero distribution $x \mapsto 0$ (note that the latter is a subdistribution, and not a distribution). We can consider the signature $\Sigma = \{\oplus_p \mid p \in [0,1]\}$ with a family of probabilistic choice*

- The ω**CPPO** structure on $D \times E$is given by:

$$
\begin{array}{rcl}
(x, y) \sqsubseteq_{D \times E} (x', y') & \Leftrightarrow & x \sqsubseteq_D x' \wedge y \sqsubseteq_E y'; \\
\bot_{D \times E} & = & (\bot_D, \bot_E); \\
\bigsqcup_{D \times E} \{(x_n, y_n) \mid n < \omega\} & = & (\bigsqcup_D \{x_n \mid n < \omega\}, \bigsqcup_E \{y_n \mid n < \omega\}).
\end{array}
$$

- The ω**CPPO** structure on $D \xrightarrow{c} E$ is given by:

$$
\begin{array}{rcl}
f \sqsubseteq_{D \xrightarrow{c} E} g & \Leftrightarrow & \forall x \in D.\ f(x) \sqsubseteq_E g(x); \\
\bot_{D \xrightarrow{c} E} & = & x \mapsto \bot_E; \\
\bigsqcup_{D \xrightarrow{c} E} \{f_n \mid n < \omega\} & = & x \mapsto \bigsqcup_E \{f_n(x) \mid n < \omega\}.
\end{array}
$$

[2]We call such ω**CPO**s *discrete*

operations indexed by real numbers in $[0,1]$*. We can interpret* $\oplus_p$ *as the binary operation* $(x,y) \mapsto p \cdot x + (1-p) \cdot y$*, which is indeed continuous.*

- *The set* $(S \times X)^S_\perp$*, or equivalently* $S \rightharpoonup (X \times S)$ *(the set of partial states over* X*) with extension order:* $f \sqsubseteq g$ *iff* $\forall x \in X.\ f(x) \neq \perp \Rightarrow f(x) = g(x)$*, for a fixed set* S *(of states). The bottom element is the totally undefined function* $x \mapsto \perp$*, whereas the least upper bound of a chain* $(f_n)_{n<\omega}$ *is computed pointwise. Depending on the choice of* S*, we can define several continuous operations on* $(S \times X)^S_\perp$*. For instance, taking* $S = \{\mathtt{true}, \mathtt{false}\}$*, the set of booleans, we can consider the signature* $\Sigma = \{\mathsf{read}, \mathsf{write}_b \mid b \in S\}$ *to be interpreted as the continuous operations read and write*$_b$ *defined by*

$$
\begin{aligned}
\mathit{write}_b(f) &= x \mapsto f(b); \\
\mathit{read}(f,g) &= x \mapsto \mathit{if}\ x = \mathsf{true}\ \mathit{then}\ f(x)\ \mathit{else}\ g(x).
\end{aligned}
$$

- *The set* $U^\infty \times X_\perp$ *(modelling computations with output streams) with the product order, for a fixed set* U *(think of* U *as a set of characters). The set* U^∞ *of streams over* U *is the set of all finite and infinite strings (or words) over* U*. Formally, a stream* $u \in U^\infty$ *is a function* $u : \mathbb{N} \to U_\perp$ *(i.e. a partial function from* $\mathbb{N}$ *to* U*) such that* $u(n) = \perp$ *implies* $u(n+c) = \perp$*, for any* $c \geq 0$*. A finite stream is a function* u *such that there exists an* n *for which* $u(n) = \perp$*. We can endow* U^∞ *with the so-called approximation order, i.e. the extension order on* $\mathbb{N} \to U_\perp$*. A finite approximation of length* n *of a stream* u *is a stream* w *of length* n *such that* $w \sqsubseteq u$ *holds. Clearly, the set of finite approximants of a stream* u *forms an* ω*-chain, and for any* $u \in U^\infty$ *we have* $u = \bigsqcup_{n<\omega} u^{(n)}$*, where* $u^{(n)} = (u(0), \ldots, u(n-1), \perp)$ *denotes the* n*-th approximant of* u*. We can define the concatenation* $u :: w$ *of a finite stream* $u = (u(0), \ldots, u(n-1), \perp)$ *and a stream* $w \in U^\infty$ *by*

$$
(u :: w)(k) = \begin{cases} u(k) & \text{if } k \leq n-1; \\ w(c) & \text{if } k = n + c, \text{ for } c \geq 0. \end{cases}
$$

  We can extend concatenation to infinite streams defining $u :: w = u$*, for* u *infinite stream. It is easy to prove that concatenation is continuous in its second argument, although even monotonicity fails for its first argument. For, consider the streams* c, cc *(which are shorthand for* $(c, \perp), (c, c, \perp)$*, respectively). We clearly have* $c \sqsubseteq cc$*, but* $c :: b = cb \not\sqsubseteq ccb = cc :: b$*.*
  Finally, we can consider the signature $\Sigma = \{\mathsf{print}_c \mid c \in U\}$ *interpreted as the family of operations* print_c *defined by* $\mathit{print}_c(u,x) = (c :: u, x)$*. It is easy to see that since concatenation is continuous in its second argument, then so does* print_c*.*

3.2 Monads

The notion of monad is given via the equivalent notion of Kleisli Triple (see [31]). Let $\mathbb{C}$ be a category.

Definition 5. *A Kleisli Triple* $\langle T, \eta, (\cdot)^\dagger \rangle$ *consists of an endomap* T *over objects of* $\mathbb{C}$*, a family of arrows* η_X*, for any object* X*, and an operation (called Kleisli extension or Kleisli star)* $(\cdot)^\dagger : \mathsf{Hom}_{\mathbb{C}}(X, TY) \to \mathsf{Hom}_{\mathbb{C}}(TX, TY)$*, (for all objects* X, Y*) satisfying the equations*

$$
\begin{aligned}
f^\dagger \circ \eta &= f; \\
\eta^\dagger &= id; \\
(g^\dagger \circ f)^\dagger &= g^\dagger \circ f^\dagger;
\end{aligned}
$$

where f *and* g *have the appropriate types.*

Given the equivalence between the notions of monad and Kleisli Triple, we will be terminologically sloppy, using the terms 'monads' and 'Kleisli Triples' interchangeably. In particular, for a monad/Kleisli Triple $\langle T, \eta, (\cdot)^\dagger \rangle$ we will implicitly assume functoriality of the endomap T. Finally, we will often denote a Kleisli Triple $\langle T, \eta, (\cdot)^\dagger \rangle$ simply as T.

To any Kleisli Triple $\langle T, \eta, (\cdot)^\dagger \rangle$ on a category $\mathbb{C}$ we can associate the so-called Kleisli category $\mathcal{K}\ell(T)$ over $\mathbb{C}$.

Definition 6. *Given a Kleisli triple as above, we define the Kleisli category $\mathcal{K}\ell(T)$ (over $\mathbb{C}$) as follows:*

- *Objects of $\mathcal{K}\ell(T)$ are those of $\mathbb{C}$.*
- *To any arrow $f : X \to TY$ in $\mathbb{C}$ we associate an arrow $\bar{f} : X \to Y$ in $\mathcal{K}\ell(T)$.*
- *The identity arrow $id_X : X \to X$ in $\mathcal{K}\ell(T)$ is η_X.*
- *Given arrows $\bar{f} : X \to Y, \bar{g} : Y \to Z$ (which correspond to arrows $f : X \to TY$ and $g : Y \to TZ$ in $\mathbb{C}$), define their composition to be $g^\dagger \circ f$.*

From now on we fix the base category $\mathbb{C}$ to be the category $\mathbb{SET}$ of sets and functions.

Remark 3. *Since we work in $\mathbb{SET}$, we will extensively use the so called bind operator $\gg\!=$ in place of Kleisli extensions. Such operator takes as arguments an element u of TX, together with a function $f : X \to TY$ and returns an element $u \gg\!= f$ in TY. Concretely, we can define $u \gg\!= f$ as $f^\dagger(u)$. Vice versa, we can define the Kleisli extension $f^\dagger$ of f as $x \mapsto (x \gg\!= f)$.*

Example 2. *All the constructions introduced in Example 1 carry the structure of a monad.*

- *The functor $TX = X_\perp$ is (part of) a monad, with left injection as unit and bind operator defined by*
$$u \gg\!= f = \begin{cases} f(x) & \text{if } u = \mathsf{in}_l(x), \text{ for some } x \in X; \\ \mathsf{in}_r(\perp) & \text{otherwise.} \end{cases}$$
- *The powerset functor $\mathcal{P}$ is a monad with unit $x \mapsto \{x\}$ and bind operator defined by $u \gg\!= f = \bigcup_{x \in u} f(x)$.*
- *The subdistribution functor $\mathcal{D}$ is a monad with unit given via the Dirac distribution δ and bind operator defined by*
$$\mu \gg\!= f = y \mapsto \sum_{x \in X} \mu(x) \cdot f(x)(y).$$
- *The partiality and exception functor $TX = (X + E)_\perp$ for a given set E of exceptions is a monad with the function $x \mapsto \mathsf{in}_l(\mathsf{in}_l(x))$ as unit. The bind operator is defined by*
$$u \gg\!= f = \begin{cases} u & \text{if } u = \mathsf{in}_r(\perp) \text{ or } u = \mathsf{in}_l(\mathsf{in}_r(e)); \\ f(x) & \text{if } u = \mathsf{in}_l(\mathsf{in}_l(x)). \end{cases}$$
- *The partiality and global state functor $TX = S \to (X \times S)_\perp$ for a given set S of states, is a monad with unit $x \mapsto (s \mapsto (x, s))$ and the bind operator defined by*
$$(\sigma \gg\!= f)(s) = \begin{cases} \mathsf{in}_r(\perp) & \text{if } \sigma(s) = \mathsf{in}_r(\perp); \\ f(y)(t) & \text{if } \sigma(s) = \mathsf{in}_r(y, t). \end{cases}$$
- *The output functor $TX = U^\infty \times X_\perp$ is a monad with unit $x \mapsto (\varepsilon, \mathsf{in}_l(x))$ where, to avoid confusion, we use denote the empty stream by ε, and bind operator defined by*
$$\begin{aligned} (u, \mathsf{in}_r(\perp)) \gg\!= f &= (u, \mathsf{in}_r(\perp)); \\ (u, \mathsf{in}_l(x)) \gg\!= f &= (u :: w, y) \\ &\qquad \text{where } (w, y) = f(x). \end{aligned}$$

For a given signature Σ, we are interested in monads on $\mathbb{SET}$ that carry a continuous Σ-algebra structure.

Definition 7. *An ω**CPPO** order $\sqsubseteq$ on a monad T is a map that assigns to each set X a relation $\sqsubseteq_X \subseteq TX \times TX$ and an element $\perp_X \in TX$ such that*

- *The structure $(TX, \sqsubseteq_X, \perp_X)$ is an ω**CPPO**.*

- *The bind operator is continuous in both arguments. That is,*

$$(\bigsqcup_{n<\omega} u_n) \gg= f = \bigsqcup_{n<\omega} (u_n \gg= f);$$
$$u \gg= (\bigsqcup_{n<\omega} f_n) = \bigsqcup_{n<\omega} (u \gg= f_n).$$

*We say that $\sqsubseteq$ is strict in its first argument if we additionally have $\bot \gg= f = \bot$ (and similarly for its second argument). We say that T carries a continuous Σ-algebra structure if T has an ω***CPPO** *order such that TX is a continuous Σ-algebra with respect to the order $\sqsubseteq_X$, for any set X.*

Most of the time we will work with a fixed set X. As a consequence, we will omit subscripts, just writing $\sqsubseteq$ in place of $\sqsubseteq_X$. Similarly, for an operation σ in Σ, we will write σ^T in place of σ^{TX} (the interpretation of σ as an operation on TX).

Remark 4. *The last definition is essentially regarding the bind operator as a* continuous *function (in both arguments) from $TX \times (X \to TY)$ to TY. This makes sense since $TX \times (X \to TY)$ is an ω***CPPO***: regarding the set X as the discrete ω***CPO***, we have $X \to TY = X \xrightarrow{c} TY$, so that $TX \times (X \to TY)$ is an ω***CPPO***, being the product of two ω***CPPO***s. Because $\gg=$ is continuous in both its arguments, we have $(\bigsqcup_n u_n) \gg= (\bigsqcup_n f_n) = \bigsqcup_n (u_n \gg= f_n)$.*

The bind operation will be useful when giving an operational semantics to the sequential (monadic) composition of programs. As a consequence, although we did not explicitly require the bind operator to be strict (especially in its first argument), such condition will be often desired (especially when giving semantics to call-by-value languages).

Example 3. *Example 1 shows that all monads in Example 2 have an ω***CPPO** *order. It is easy to check that all bind operations, with the exception of the one for the output monad, are strict in their first argument. In fact, even monotonicity of the bind operator for output monad fails, due to the failure of monotonicity for concatenation (see 1). The reason why this property does not hold for the output monad relies on non-monotonicity of the concatenation operator on its first argument. Nonetheless, we can endow $U^\infty \times X_\bot$ with a different order, obtaining the desired result:*

$$(u, x) \sqsubseteq (w, y) \quad \textit{iff} \quad (x = \mathsf{in}_r(\bot) \wedge u \sqsubseteq w) \vee (x \neq \mathsf{in}_r(\bot) \wedge x = y \wedge u = w).$$

*It is not hard to see that we obtain an ω***CPPO** *with continuous bind operator.*

Definition 7 requires the bind operator to be continuous. This condition is a special case of the more general notion of order-enrichment [23] for a monad.

Definition 8. *We say that a category $\mathbb{C}$ is ω***CPPO***-enriched if*
- *Each hom-set $\mathsf{Hom}_{\mathbb{C}}(X, Y)$ carries a partial order $\sqsubseteq$ with an ω***CPPO** *structure.*
- *Composition is continuous. That is, the following equations hold:*

$$g \circ (\bigsqcup_{n<\omega} f_n) = \bigsqcup_{n<\omega} (g \circ f_n);$$
$$(\bigsqcup_{n<\omega} f_n) \circ g = \bigsqcup_{n<\omega} (f_n \circ g).$$

Definition 9. *A monad T on $\mathbb{C}$ is ω***CPPO***-enriched if $\mathcal{K}\ell(T)$ is ω***CPPO***-enriched. That is, for every pair of objects X, Y, the set $\mathsf{Hom}_{\mathbb{C}}(X, TY)$ carries an ω***CPPO***-structure such that composition is continuous and Kleisli star is locally continuous. Concretely, that means that the following equations hold (cf. [16]):*

$$(\bigsqcup_{n<\omega} f_n) \circ h = \bigsqcup_{n<\omega} (f_n \circ h);$$
$$u^\dagger \circ \bigsqcup_{n<\omega} f_n = \bigsqcup_{n<\omega} (u^\dagger \circ f_n);$$
$$(\bigsqcup_{n<\omega} f_n)^\dagger = \bigsqcup_{n<\omega} f_n^\dagger.$$

Our notion of ω**CPPO** order on a monad T on $\mathbb{SET}$ is nothing but a special case of ω**CPPO**-enrichment. Since we are in $\mathbb{SET}$, and we have the terminal object 1 (say $1 = \{*\}$), any element u of TX directly corresponds to the arrow $\bar{u} : 1 \to TX$, defined by $\bar{u}(*) = u$. In particular, we have $TX \cong 1 \to TX = 1 \xrightarrow{c} TX$ (since 1 is discrete). For a function $f : X \to Y$ and an element $u \in X$ we can simulate function application $f(u)$ as $\bar{u} \circ f$ (meaning that $\bar{u} \circ f = \overline{f(u)}$). As a consequence, we have that $u \gg\!= f$ corresponds to $f^\dagger \circ \bar{u}$. Finally, observe that the equation

$$\overline{\bigsqcup_{TX}\{u_n \mid n < \omega\}} = \bigsqcup_{1 \to TX} \{\overline{u}_n \mid n < \omega\}$$

holds. We show that if T is ω**CPPO**-enriched, then the bind operator is continuous in both arguments. In fact, $(\bigsqcup_{n<\omega} u_n) \gg\!= f$ corresponds to the function

$$f^\dagger \circ \overline{\bigsqcup_{n<\omega} u_n} = f^\dagger \circ \bigsqcup_{n<\omega} \bar{u}_n = \bigsqcup_{n<\omega} (f^\dagger \circ \bar{u}_n)$$

which itself corresponds to $\bigsqcup_{n<\omega}(u_n \gg\!= f)$. Similarly, $u \gg\!= \bigsqcup_{n<\omega} f_n$ corresponds to

$$(\bigsqcup_{n<\omega} f_n)^\dagger \circ \bar{u} = \bigsqcup_{n<\omega} f_n^\dagger \circ \bar{u} = \bigsqcup_{n<\omega} (f_n^\dagger \circ \bar{u})$$

which corresponds to $\bigsqcup_{n<\omega}(u \gg\!= f_n)$.

Most of the monads commonly used e.g. in functional programming to model side-effects are not order enriched. This follows from the requirement of having a bottom element. The reason behind that condition relies on the fact that our operational semantics will be model non-termination explicitly. That is, a (purely) divergent program M will be evaluated in the bottom element of the monad. For instance let us consider pure λ-calculus. Standard operational semantics employs inductively defined judgments of the form $M \Downarrow V$, meaning that $\Downarrow \subseteq \Lambda \times \mathcal{V}$ (which, in general, can be viewed as $\Downarrow \subseteq \Lambda \times T\mathcal{V}$, for T the identity monad). Such a semantics does not capture divergence explicitly: for instance, we just have that there exists no value V such that $\Omega \Downarrow V$. The operational semantics we will define in the next chapter associates to each program a subset of the finite approximations it is evaluated to, and then consider the lub of such approximations. As a consequence, we need the monad to have bottom element $\perp$, so that we will have that the semantics of Ω to be indeed $\perp$. Nevertheless, we recall that any set can always be lifted to an ω**CPPO** by adding a bottom element to it and considering the flat ordering. As a consequence, although most of the monads commonly used in functional programming are not order-enriched, their flat version is.

4 A Computational Calculus and Its Operational Semantics

In this section we define a computational λ-calculus. Following [32, 27, 30], we syntactically distinguish between values and computations. We fix a signature Σ of operation symbols (the sources of side-effects), and a monad T carrying a continuous Σ-algebra structure (which describes the nature of the wanted effectful computations generated by the operations in Σ).

Definition 10. *Given a signature Σ, the sets Λ_Σ and $\mathcal{V}_\Sigma$ of terms and values are defined by the following grammars:*

$$\begin{aligned} M, N &::= \mathsf{return}\ V \mid VW \mid M\ \mathsf{to}\ x.N \mid \sigma(M, \dots, M); \\ V, W &::= x \mid \lambda x.M. \end{aligned}$$

where x ranges over a fixed countably infinite set $\mathtt{X}$ *of variables and σ ranges over Σ.*

The term $(M\ \mathsf{to}\ x.N)$ captures monadic binding (which is usually expressed using a "let-in" notation). A calculus with an explicit separation between terms and values has the advantage to make proofs simpler, without sacrificing expressiveness. For instance, we can encode terms' application MN as $(M\ \mathsf{to}\ x.(N\ \mathsf{to}\ y.xy))$ and vice versa $(M\ \mathsf{to}\ x.N)$ as $(\lambda x.N)M$.

Example 4. *We can model several calculi combining the signatures from Example 1.*

- *For a given set E of exceptions, we can define a probabilistic λ-calculus with exceptions as Λ_Σ, for a signature $\Sigma = \{\oplus_p, \mathsf{raise}_e \mid p \in [0,1], e \in E\}$. In particular, we will have terms of the form $M \oplus_p N$ and raise_e. Replacing the probabilistic choice operator $\oplus_p$ with its nondeterministic counterpart $\oplus$ we obtain a nondeterministic calculus with exceptions.*
- *We can define a nondeterministic calculus with global (boolean) states as Λ_Σ, for a signature $\Sigma = \{\oplus, \mathsf{write}_b, \mathsf{read} \mid b \in \{true, false\}\}$. In particular, we will have terms of the form $M \oplus N$, $\mathsf{write}_b.M$, and $\mathsf{read}(M,N)$. The intuitive meaning of $\mathsf{write}_b.M$ is to store b and then continue as M, whereas the intuitive meaning of $\mathsf{read}(M,N)$ is to read the value in the store: if such value is the boolen true then continue as M, otherwise as N. A formal semantics for these two functions is given in Example 1.*
- *We can define a nondeterministic calculus with output using the signature $\Sigma = \{\oplus, \mathsf{print}_c \mid c \in U\}$, where U is a given alphabet. The intuitive meaning of $\mathsf{print}\ c.M$ is to output c and then continue as M. A formal semantics for this function is given in Example 1.*

In what follows, we work with a fixed arbitrary signature Σ. As a consequence, we often denote the sets of terms and values as Λ and $\mathcal{V}$, respectively, thus omitting subscripts. Moreover, we consider terms and values modulo α-equivalence and assume Barendregt Convention [4]. We let $FV(M)$ denote the set of free variables of the term M. A term M is closed if $FV(M) = \emptyset$. We denote finite sets of variables, terms and values using "bar notation": for instance, we write $\bar{x}$ and $\bar{V}$ for a finite set of variables and values, respectively. For a finite set $\bar{x}$ of variables define

$$\Lambda(\bar{x}) = \{M \mid FV(M) \subseteq \bar{x}\};$$
$$\mathcal{V}(\bar{x}) = \{V \mid FV(V) \subseteq \bar{x}\};$$

to be the sets of terms and values with free variables in $\bar{x}$, respectively. The set of closed terms and values are then defined as $\Lambda(\emptyset)$ and $\mathcal{V}(\emptyset)$, and denoted by Λ_0 and $\mathcal{V}_0$, respectively.

Definition 11. *Define for all values V, W and any term M, the value $V[W/y]$ obtained by (simultaneous) substitution of W for y in V, and the term $M[y := V]$ obtained by (simultaneous) substitution of V for y in M as follows (recall we are assuming Barendregt's convention):*

$$\begin{aligned}
x[W/x] &= W \\
x[W/y] &= x \\
(\lambda x.M)[W/y] &= \lambda x.M[y := W] \\
\\
(\mathsf{return}\ V)[y := W] &= \mathsf{return}\ V[W/y] \\
(VV')[y := W] &= V[W/y]V'[W/y] \\
(M\ \mathsf{to}\ x.N)[y := W] &= M[y := W]\ \mathsf{to}\ x.N[y := W]
\end{aligned}$$

Big-step semantics associates to each closed term M an element $[\![M]\!]$ in $T\mathcal{V}_0$. Such a semantics is defined by means of an approximation relation $\Downarrow_n$, indexed by a natural number n, whose definition is given in Figure 3. Judgments are of the form $M \Downarrow_n X$, where $M \in \Lambda_0$, $X \in T\mathcal{V}_0$ and $n \geq 0$. Intuitively, a judgment $M \Downarrow_n X$ states that X is the n-th approximation of the computation obtained by call-by-value evaluating M. (By the way, all the results in this paper would remain valid also if evaluating terms in call-by-*name* order, which is however less natural in presence of effects.)

The system in Figure 3 is 'syntax directed', meaning that given a judgment $M \Downarrow_n X$, the solely syntactic form of M and the number n uniquely determine the last rule used to derive $M \Downarrow_n X$. As a consequence, each judgment has a unique derivation.

Lemma 3 (Determinacy)**.** *For any term M, if $M \Downarrow_n X$ and $M \Downarrow_n Y$, then $X = Y$.*

Proof. By induction on n. If $n = 0$, then both $M \Downarrow_n X$ and $M \Downarrow_n Y$ must be the conclusion of an instance of rule (bot) (all other rules requires n to be positive). As a consequence, we have

$$\frac{}{M \Downarrow_0 \bot}\ (\mathsf{bot}) \qquad \frac{}{\mathsf{return}\ V \Downarrow_{n+1} \eta(V)}\ (\mathsf{ret})$$

$$\frac{M \Downarrow_n X \qquad N[x := V] \Downarrow_n Y_V}{M\ \mathsf{to}\ x.N \Downarrow_{n+1} X \gg\!= (V \mapsto Y_V)}\ (\mathsf{seq})$$

$$\frac{M[x := V] \Downarrow_n X}{(\lambda x.M)V \Downarrow_{n+1} X}\ (\mathsf{app})$$

$$\frac{M_1 \Downarrow_n X_1 \qquad \dots \qquad M_k \Downarrow_n X_k}{\sigma(M_1, \dots, M_k) \Downarrow_{n+1} \sigma^T(X_1, \dots, X_k)}\ (\mathsf{op})$$

Figure 3: Big-step Semantics.

$X = \bot = Y$. Suppose now $n = m + 1$, for some $m \geq 0$. We proceed by case analysis on the last rule used to derive $M \Downarrow_n X$.

Case (bot). This case is not possible, since $n > 0$.

Case (ret). Then M is of the form return V, for some value V, X is $\eta(V)$, and $M \Downarrow_{m+1} Y$ is return $V \Downarrow_{m+1} Y$. The latter judgment must follow from an instance of rule (ret) as well, and thus $Y = \eta(V)$.

Case (app). Then M is of the form $(\lambda x.N)V$ and we have $N[x := V] \Downarrow_m X$, for some term N. Therefore, the judgment $M \Downarrow_{m+1} Y$ is of the form $(\lambda x.N)V \Downarrow_{m+1} Y$ implying it can only be the conclusion of an instance of the rule (app). Therefore, $N[x := V] \Downarrow_m Y$ holds as well. We can apply the induction hypothesis on the latter and $N[x := V] \Downarrow_m X$ thus inferring $X = Y$.

Case (seq). Then M is of the form N to $x.N'$, X is of the form $X' \gg\!= (V \mapsto X'_V)$, and both $N \Downarrow_m X'$ and $N'[x := V] \Downarrow_m X'_V$ hold, for some terms N, N' and elements X', X'_V in $T\mathcal{V}_0$. As a consequence, the judgment $M \Downarrow_{m+1} Y$ has the form N to $x.N' \Downarrow_{m+1} Y$, implying it must be the conclusion of an instance of the rule (seq) as well. Therefore, we have $N \Downarrow_m Y'$ and $N'[x := V] \Downarrow_m Y'_V$, and $Y = Y' \gg\!= (V \mapsto Y'_V)$, for some elements Y', Y'_V. We can then apply the induction hypothesis on $N \Downarrow_m X'$, $N \Downarrow_m Y'$ and $N'[x := V] \Downarrow_m X'_V$, $N'[x := V] \Downarrow_m Y'_V$, obtaining $X' = Y'$, $X'_V = Y'_V$ and thus $X' \gg\!= (V \mapsto X'_V) = Y' \gg\!= (V \mapsto Y'_V)$.

Case (op). Then M is of the form $\sigma(M_1, \dots, M_k)$, X is of the form $\sigma^T(X_1, \dots, X_k)$, and the judgment $M_1 \Downarrow_m X_1, \dots, M_k \Downarrow_m X_k$ hold, for some terms $M_1, \dots, M_k$ and elements $X_1, \dots, X_k$ in $T\mathcal{V}_0$. As a consequence, the judgment $M \Downarrow_{m+1} Y$ has the form $\sigma^T(M_1, \dots, M_k) \Downarrow_{m+1} Y$. The latter must be the conclusion of an instance of the rule (op), meaning that we have judgments $M_1 \Downarrow_m Y_1, \dots, M_k \Downarrow_m Y_m$ and $Y = \sigma^T(Y_1, \dots, Y_m)$. We can apply the induction hypothesis on the pair of judgments $M_i \Downarrow_m X_i, M_i \Downarrow_m Y_i$, for $i \in \{1, \dots, k\}$, inferring $X_i = Y_i$. We conclude $\sigma^T(X_1, \dots, X_k) = \sigma^T(Y_1, \dots, Y_k)$.

□

Lemma 4. *For any term M if $M \Downarrow_n X$ and $M \Downarrow_{n+N} Y$, then $X \sqsubseteq Y$.*

Proof. The proof follows the same pattern of the previous one, where in the inductive case we use monotonicity of both the bind operator and the operations σ^T. □

Corollary 1. *Let M be a a term and X_n be the (unique) element in $T\mathcal{V}_0$ such that $M \Downarrow_n X$. Then, the sequence $(X_n)_{n<\omega}$ forms an ω-chain in $T\mathcal{V}_0$.*

A direct consequence of the above corollary is that we can define the evaluation $[\![M]\!]$ of a term M as

$$[\![M]\!] = \bigsqcup_{M \Downarrow_n X} X.$$

This allows us to explicitly capture non-termination (which is usually defined coinductively). For instance, it is easy to show that for the purely (i.e. having no side-effects) divergent program

Ω, defined as $(\lambda x.xx)(\lambda x.xx)$, we have $[\![\Omega]\!] = \bot$. This style of operational semantics [11, 10] is precisely the reason we require the monad T to carry an ω**CPPO** structure. Modelling divergence in this way turned out to be fundamental in e.g. probabilistic calculi [11].

Definition 12. *Let M be a term. Define the n-th approximation $M^{(n)} \in T\mathcal{V}_0$ of M as follows:*

$$\begin{aligned}
M^{(0)} &= \bot \\
(\mathsf{return}\ V)^{(n+1)} &= \eta(V) \\
((\lambda x.M)V)^{(n+1)} &= (M[x := V])^{(n)} \\
(M\ \mathsf{to}\ x.N)^{(n+1)} &= M^{(n)} \mathbin{>\!\!>\!=} (V \mapsto (N[x := V])^{(n)}) \\
(\sigma(M_1, \dots, M_k))^{(n+1)} &= \sigma^T(M_1^{(n)}, \dots, M_k^{(n)})
\end{aligned}$$

Lemma 5. *For any term M we have $M \Downarrow_n M^{(n)}$.*

Proof. The proof is by induction on n. If $n = 0$, then we trivially have $M \Downarrow_0 \bot$. If $n = m + 1$, for some $m \geq 0$, we proceed by case analysis on the last rule used to derive the judgment $M \Downarrow_{m+1} M^{(m+1)}$. As a paradigmatic example, we show the case for rule (seq). Suppose $M \Downarrow_{m+1} M^{(m+1)}$ is of the form $(N\ \mathsf{to}\ x.N') \Downarrow_{m+1} Y \mathbin{>\!\!>\!=} (V \mapsto Y'_V)$ and the judgments $N \Downarrow_m Y$, $N'[x := V] \Downarrow_m Y'_V$ hold, for some terms N, N' and elements Y, Y'_V in $T\mathcal{V}_0$. We can apply the induction hypothesis on m, obtaining $N \Downarrow_m N^{(m)}$ and $N'[x := V] \Downarrow_m (N'[x := V])^{(m)}$. By Lemma 3 we thus have $N^{(m)} = Y$ and $(N'[x := V])^{(m)} = Y'_V$. We can conclude

$$Y \mathbin{>\!\!>\!=} (V \mapsto Y'_V) = N^{(m)} \mathbin{>\!\!>\!=} (V \mapsto (N'[x := V])^{(m)}) = (N\ \mathsf{to}\ x.N')^{(m+1)}.$$

$\square$

Corollary 1 and Lemma 5 together imply that for any term M we have the ω-chain $(M^{(n)})_{n<\omega}$ of finite approximations of M. That means, in particular, that $[\![M]\!]$ is equal to $\bigsqcup_{n<\omega} M^{(n)}$. For instance, by previous lemma we have $\Omega^{(0)} = \bot$ and $\Omega^{(n+1)} = ((xx)[x := \lambda x.xx])^{(n)} = \Omega^{(n)}$. As a consequence, we have for any n, $\Omega^{(n)} = \bot$, and thus $[\![\Omega]\!] = \bot$.

Since both $\mathbin{>\!\!>\!=}$ and σ^T are continuous, we can characterise operational semantics equationally.

Lemma 6. *The following equations hold:*

$$\begin{aligned}
[\![\mathsf{return}\ V]\!] &= \eta(V); \\
[\![(\lambda x.M)V]\!] &= [\![M[x := V]]\!]; \\
[\![M\ \mathsf{to}\ x.N]\!] &= [\![M]\!] \mathbin{>\!\!>\!=} (V \mapsto [\![N[x := V]]\!]); \\
[\![\sigma(M_1, \dots, M_n)]\!] &= \sigma^T([\![M_1]\!], \dots, [\![M_n]\!]).
\end{aligned}$$

Proof. By Lemma 1 we have $[\![M]\!] = \bigsqcup_{n<\omega} M^{(n+1)}$, meaning that we can freely ignore $M^{(0)}$ (which is $\bot$). We prove each equation separately.
Case 1. We have:

$$[\![\mathsf{return}\ V]\!] = \bigsqcup_{n<\omega} (\mathsf{return}\ V)^{(n+1)} = \bigsqcup_{n<\omega} \eta(V) = \eta(V).$$

Case 2. We have:

$$[\![(\lambda x.M)V]\!] = \bigsqcup_{n<\omega} ((\lambda x.M)V)^{(n+1)} = \bigsqcup_{n<\omega} (M[x := V])^{(n)} = [\![M[x := V]]\!].$$

Case 3. We have:

$$
\begin{aligned}
[\![M \text{ to } x.N]\!] &= \bigsqcup_{n<\omega} (M \text{ to } x.N)^{(n+1)} \\
&= \bigsqcup_{n<\omega} (M^{(n)} \gg\!\!= (V \mapsto (N[x := V])^{(n)})) \\
&= \bigsqcup_{n<\omega} M^{(n)} \gg\!\!= \bigsqcup_{n<\omega} (V \mapsto (N[x := V])^{(n)}) && \text{(Continuouity of } \gg\!\!= \text{)} \\
&= \bigsqcup_{n<\omega} M^{(n)} \gg\!\!= (V \mapsto \bigsqcup_{n<\omega} (N[x := V])^{(n)}) && \text{(Lub of functions)} \\
&= [\![M]\!] \gg\!\!= (V \mapsto [\![N[x := V]]\!]).
\end{aligned}
$$

Case 4. We have:

$$
\begin{aligned}
[\![\sigma(M_1, \ldots, M_k)]\!] &= \bigsqcup_{n<\omega} (\sigma(M_1, \ldots, M_k))^{(n+1)} \\
&= \bigsqcup_{n<\omega} \sigma^T(M_1^{(n)}, \ldots, M_k^{(n)}) \\
&= \sigma^T(\bigsqcup_{n<\omega} M_1^{(n)}, \ldots, \bigsqcup_{n<\omega} M_k^{(n)}) && \text{(Continuouity of } \sigma^T\text{)} \\
&= \sigma^T([\![M_1]\!], \ldots, [\![M_k]\!])
\end{aligned}
$$

□

It is actually not hard to see that the function $[\![\cdot]\!]$ is the least solution to the equations in Lemma 6.

5 On Relational Reasoning

In this section we introduce the main machinery behind our soundness results. The aim is to generalise notions and results from e.g. [27, 17, 28] to take into account generic effects. We will use results from the theory of coalgebras [39] to come up with a general notion of applicative (bi)similarity parametric over a notion of observation, given through the concept of relator.

5.1 Relators

The concept of relator [41, 29] is an abstraction meant to capture the possible ways a relation on a set X can be turned into a relation on TX. Recall that for an endofunctor $F : \mathbb{C} \to \mathbb{C}$, an F-coalgebra [39] consists of an object X of $\mathbb{C}$ together with a morphism $\gamma_X : X \to FX$. As usual, we are just concerned with the case in which $\mathbb{C}$ is $\mathbb{SET}$.

Definition 13. *Let F be an endofunctor on $\mathbb{SET}$, and X, Y be sets. A relator Γ for F is a map that associates to each relation $\mathcal{R} \subseteq X \times Y$ a relation $\Gamma\mathcal{R} \subseteq FX \times FY$ such that*

$$=_{FX} \subseteq \Gamma(=_X) \tag{Rel-1}$$

$$\Gamma\mathcal{S} \circ \Gamma\mathcal{R} \subseteq \Gamma(\mathcal{S} \circ \mathcal{R}) \tag{Rel-2}$$

$$\Gamma((f \times g)^{-1}\mathcal{R}) = (Ff \times Fg)^{-1}\Gamma\mathcal{R} \tag{Rel-3}$$

$$\mathcal{R} \subseteq \mathcal{S} \implies \Gamma\mathcal{R} \subseteq \Gamma\mathcal{S} \tag{Rel-4}$$

where for $f : Z \to X$, $g : W \to Y$ we have $(f \times g)^{-1}\mathcal{R} = \{(z, w) \mid f(z)\ \mathcal{R}\ g(w)\}$, and $=_X$ denotes the identity relation on X. A relator Γ is conversive *if $\Gamma(\mathcal{R}^c) = (\Gamma\mathcal{R})^c$, where $\mathcal{R}^c$ denotes the converse of $\mathcal{R}$.*

Example 5. *For each of the monads introduced in previous sections, we give some examples of relators. Most of these relators coincide with the relation lifting of their associated functor. It is in fact well known that for any weak-pullback preserving functor, its relation lifting is a relator [25]. We use the notation Γ for a relator aimed to capture the structure of a simulation relation, and Δ for a relator aimed to capture the structure of a bisimulation relation. This distinction is not formal, and only makes sense in the context of concrete examples: its purpose is to stress that from formal view point, both concrete notions of similarity and bisimilarity are modeled as forms of Γ-similarity (for a suitable relator Γ). Let $\mathcal{R} \subseteq X \times Y$:*

- *For the partiality monad $TX = X_\perp$ define the relators $\Gamma_\perp, \Delta_\perp$ by*

$$\begin{aligned} u\ \Gamma_\perp\mathcal{R}\ v \textit{ iff } & u = \mathsf{in}_l(x) \implies v = \mathsf{in}_l(y) \wedge x\ \mathcal{R}\ y; \\ u\ \Delta_\perp\mathcal{R}\ v \textit{ iff } & u = \mathsf{in}_l(x) \implies v = \mathsf{in}_l(y) \wedge x\ \mathcal{R}\ y, \\ & v = \mathsf{in}_l(y) \implies u = \mathsf{in}_l(x) \wedge x\ \mathcal{R}\ y. \end{aligned}$$

  Note that $u = \mathsf{in}_l(x)$ means, in particular, $u \neq \mathsf{in}_r(\perp)$. Thus, for instance, u and v are $\Gamma_\perp\mathcal{R}$ related if whenever u converges, so does v and the values to which u, v converge are $\mathcal{R}$-related. The relator $\Delta_\perp$ is conversive.

- *For the nondeterministic powerset monad $\mathcal{P}$ define relators $\Gamma_\mathcal{P}$ and $\Delta_\mathcal{P}$ by*

$$\begin{aligned} u\ \Gamma_\mathcal{P}\mathcal{R}\ v \textit{ iff } & \forall x \in u.\ \exists y \in v.\ x\ \mathcal{R}\ y; \\ u\ \Delta_\mathcal{P}\mathcal{R}\ v \textit{ iff } & \forall x \in u.\ \exists y \in v.\ x\ \mathcal{R}\ y, \\ & \forall y \in v.\ \exists x \in u.\ x\ \mathcal{R}\ y. \end{aligned}$$

  The relator $\Delta_\mathcal{P}$ is conversive.

- *For the probabilistic subdistributions monad $\mathcal{D}$ define relators $\Gamma_\mathcal{D}$ and $\Delta_\mathcal{D}$ by*

$$\begin{aligned} \mu\ \Gamma_\mathcal{D}\mathcal{R}\ \nu & \textit{ iff } \forall U \subseteq X.\ \mu(U) \leq \nu(\mathcal{R}(U)); \\ \mu\ \Delta_\mathcal{D}\mathcal{R}\ \nu & \textit{ iff } \mu\ \Gamma_\mathcal{D}\mathcal{R}\ \nu\ \wedge\ \nu\ \Gamma_\mathcal{D}\mathcal{R}^c\ \mu; \end{aligned}$$

  where $\mathcal{R}(U) = \{y \in Y \mid \exists x \in U.\ x\ \mathcal{R}\ y\}$ and $\mu(U) = \sum_{x \in U} \mu(x)$. The relator $\Delta_\mathcal{D}$ is conversive.

- *For the exception monad $TX = X + E$ define the relators $\Gamma_\mathcal{E}$ and $\Delta_\mathcal{E}$ by (letters e, e' range over E)*

$$\begin{aligned} u\ \Gamma_\mathcal{E}\mathcal{R}\ v \textit{ iff } & u = \mathsf{in}_r(e) \implies v = \mathsf{in}_r(e') \wedge e = e', \\ & u = \mathsf{in}_l(x) \implies v = \mathsf{in}_l(y) \wedge x\ \mathcal{R}\ y; \\ u\ \Delta_\mathcal{E}\mathcal{R}\ v \textit{ iff } & u\ \Gamma_\mathcal{E}\mathcal{R}\ v, \\ & v = \mathsf{in}_r(e') \implies u = \mathsf{in}_r(e) \wedge e = e', \\ & v = \mathsf{in}_l(y) \implies u = \mathsf{in}_l(x) \wedge x\ \mathcal{R}\ y. \end{aligned}$$

  The relator $\Delta_\mathcal{E}$ is conversive.

- *For the partiality and exception monad (i.e. the exception monad with divergence) $TX = (X+E)_\perp$ we can define relators simply composing relators for the partiality monad with relators for the exceptions monads (see Lemma 7). Notably, define $\Gamma_{\mathcal{E}_\perp}$ as $\Gamma_\perp \circ \Gamma_\mathcal{E}$ and $\Delta_{\mathcal{E}_\perp}$ as $\Delta_\perp \circ \Delta_\mathcal{E}$. The relator $\Delta_{\mathcal{E}_\perp}$ is conversive.*

- *For the state monad $TX = (X \times S)^S$ define the relator $\Delta_\mathcal{S}$ by*

$$\begin{aligned} f\ \Delta_\mathcal{S}\mathcal{R}\ g \textit{ iff } & \forall s \in S.\ s_1 = s_2 \textit{ and } x_1\ \mathcal{R}\ x_2, \\ & \textit{where } (x_1, s_1) = f(s) \textit{ and } (x_2, s_2) = g(s). \end{aligned}$$

  The relator $\Delta_\mathcal{S}$ is conversive.

- *For the output monad $TX = U^\infty \times X_\perp$ we can define relators based on the order defined in Example 3.*

$$\begin{aligned} (u, x)\ \Gamma_\mathcal{U}\mathcal{R}\ (w, y) & \textit{ iff } (x = \mathsf{in}_r(\perp) \wedge u \sqsubseteq w) \vee (x = \mathsf{in}_l(x') \wedge y = \mathsf{in}_l(y') \wedge x'\mathcal{R}\ y'); \\ (u, x)\ \Delta_\mathcal{U}\mathcal{R}\ (w, y) & \textit{ iff } (u, x)\ \Gamma_\mathcal{U}\mathcal{R}\ (w, y) \wedge (w, y)\ \Gamma_\mathcal{U}\mathcal{R}^c\ (u, x). \end{aligned}$$

  The relator $\Delta_\mathcal{U}$ is conversive.

Checking that the above are indeed relators is a tedious but easy exercise. It is useful to know that the collection of relators is closed under certain operations (see [29] for proofs).

Lemma 7 (Algebra of Relators)**.** *Let F, G be endofunctors on $\mathbb{SET}$. Then*

1. *Let $(\Gamma_i)_{i\in I}$ be a family of relators for F. The intersection $\bigcap_{i\in I}\Gamma_i$ defined by $(\bigcap_{i\in I}\Gamma_i)\mathcal{R} = \bigcap_{i\in I}\Gamma_i(\mathcal{R})$ is a relator for F.*
2. *The converse Γ^c of Γ defined by $\Gamma^c(\mathcal{R}) = (\Gamma\mathcal{R}^c)^c$ is a relator for F. We have the equality $(\Gamma^c)^c = \Gamma$ and, additionally, $\Gamma^c = \Gamma$ if Γ is conversive.*
3. *Let Γ, Γ' be relators for F, G, respectively. Then $\Gamma' \circ \Gamma$ is a relator for $G \circ F$. Moreover, if both Γ and Γ' are conversive, then so is $\Gamma' \circ \Gamma$.*
4. *Given a relator Γ for F, $\Gamma \cap \Gamma^c$ is the greatest (wrt the pointwise order) conversive relator for F contained in Γ.*

We can now give a general notion of simulation with respect to a given relator.

5.2 Bisimulation, in the Abstract

A relator Γ for a monad T expresses the observable part of the side-effects encoded by T. Its abstract nature allows to give abstract definitions of simulation and bisimulation parametric in the notion of observation given by Γ.

Definition 14. *Let $\gamma_X : X \to FX, \gamma_Y : Y \to FY$ be F-coalgebras:*

1. *A Γ-simulation is a relation $\mathcal{R} \subseteq X \times Y$ such that*

$$x\mathcal{R}y \implies \gamma_X(x)\ \Gamma\mathcal{R}\ \gamma_Y(y).$$

2. *Γ-similarity $\precsim^{\Gamma}_{X,Y}$ is the largest Γ-simulation.*

Example 6. *It is immediate to see that the corresponding notions of Γ-similarity for the (bi)simulation relators of Example 5 coincide with widely used notions of (bi)similarity.*

As usual, the notion of similiarity can be characterised coinductively as the greatest fixed point of a suitable functional.

Definition 15. *Let $\gamma_X : X \to FX$, $\gamma_Y : Y \to FY$ be F-coalgebras. Define the functional $\mathcal{F}^{\Gamma}_{X,Y} : 2^{X\times Y} \to 2^{X\times Y}$ by*

$$\mathcal{F}^{\Gamma}_{X,Y}(\mathcal{R}) = (\gamma_X \times \gamma_Y)^{-1}(\Gamma\mathcal{R}).$$

When clear from the context, we will write $\mathcal{F}_\Gamma$ and $\precsim_\Gamma$ in place of $\mathcal{F}^{\Gamma}_{X,Y}$ and $\precsim^{\Gamma}_{X,Y}$.

Lemma 8. *The following hold:*

1. *The functional $\mathcal{F}_\Gamma$ is monotone, and thus has a greatest fixed point $\nu\mathcal{F}_\Gamma$.*
2. *A relation $\mathcal{R}$ is a Γ-simulation iff it is a post fixed-point of $\mathcal{F}_\Gamma$. Therefore, Γ-similarity coincides with $\nu\mathcal{F}_\Gamma$.*

Proof. Monotonicity of $\mathcal{F}_\Gamma$ directly follows from monotonicity of Γ, and thus it has greatest fixed point by Knaster-Tarski Theorem (recall that the set $2^{X\times Y}$ carries a complete lattice structure under the inclusion order). A straightforward calculation shows that a relation $\mathcal{R}$ is a Γ-simulation iff it is a post fixed-point of $\mathcal{F}_\Gamma$. Together with point 1, the latter implies $\nu\mathcal{F}_\Gamma = \precsim_\Gamma$. □

Proposition 1. *Let $\gamma_X : X \to FX$ be an F-coalgebra.*

1. *Γ-similarity is a preorder.*
2. *If Γ is conversive, then Γ-similarity is an equivalence relation.*

Proof. Let $\gamma_X : X \to FX$ be an F-coalgebra.

1. We prove that $\precsim_\Gamma$ is reflexive by coinduction, showing that the identity relation $=_X$ on X is a Γ-simulation. In fact, from $x =_X x$ we obtain $\gamma_X(x) =_{FX} \gamma_X(x)$, and thus we can conclude $\gamma_X(x)\ \Gamma(=_X)\ \gamma_X(x)$, by (Rel-1).
   We now show that $\precsim_\Gamma$ is transitive. Suppose to have $x \precsim_\Gamma y \precsim_\Gamma z$. By very definition of $\precsim_\Gamma$ there exist Γ-simulations $\mathcal{R}, \mathcal{S}$ such that $x\ \mathcal{R}\ y$ and $y\ \mathcal{S}\ z$, and thus $\gamma_X(x)\ (\Gamma\mathcal{S} \circ \Gamma\mathcal{R})\ \gamma_X(z)$. Thanks to (Rel-4) we can conclude that $\mathcal{S} \circ \mathcal{R}$ is a Γ-simulation as well, meaning, in particular, that $x \precsim_\Gamma z$
2. We simply observe that if $\mathcal{R}$ is a Γ-simulation, then so is $\mathcal{R}^c$, for a conversive relator Γ.

$\square$

Since T is a monad we consider relators that properly interact with the monadic structure of T, which are also known as *lax extensions* for T [5].

Definition 16. *Let T be a monad, X, X', Y, Y' be sets, $f : X \to TX', g : Y \to TY'$ be functions, and $\mathcal{R} \subseteq X \times Y, \mathcal{S} \subseteq X' \times Y'$ be relations. We say that Γ is a relator for T if it is a relator for T regarded as a functor, and*
- $x\ \mathcal{R}\ y \implies \eta_X(x)\ \Gamma\mathcal{R}\ \eta_Y(y)$*;*
- $u\ \Gamma\mathcal{R}\ v \implies (u \gg\!= f)\ \Gamma\mathcal{S}\ (v \gg\!= g)$*, whenever* $x\ \mathcal{R}\ y \implies f(x)\ \Gamma\mathcal{S}\ g(y)$*.*

Remark 5. *Definition 16 can be more compactly expressed using Kleisli star, thus requiring that*

$$\mathcal{R} \subseteq (\eta_X \times \eta_Y)^{-1}(\Gamma\mathcal{S}) \tag{Lax-Unit}$$

$$\mathcal{R} \subseteq (f, g)^{-1}(\Gamma\mathcal{S}) \Longrightarrow \Gamma\mathcal{R} \subseteq (f^\dagger \times g^\dagger)^{-1}(\Gamma\mathcal{S}) \tag{Lax-Bind}$$

or diagramatically

$$\begin{tikzcd} X \arrow[r, "\mathcal{R}", "/" marking] \arrow[d, "\eta_X"'] & Y \arrow[d, "\eta_Y"] \\ TX \arrow[r, "/" marking, "\Gamma\mathcal{R}"'] & TY \end{tikzcd}$$

$$\begin{tikzcd} X \arrow[r, "\mathcal{R}", "/" marking] \arrow[d, "f"'] & Y \arrow[d, "g"] \\ TX' \arrow[r, "/" marking, "\Gamma\mathcal{S}"'] & TY' \end{tikzcd} \implies \begin{tikzcd} TX \arrow[r, "\Gamma\mathcal{R}", "/" marking] \arrow[d, "f^\dagger"'] & TY \arrow[d, "g^\dagger"] \\ TX \arrow[r, "/" marking, "\Gamma\mathcal{S}"'] & TY \end{tikzcd}$$

where we write $\mathcal{R} : X \nrightarrow Y$ for $\mathcal{R} \subseteq X \times Y$.

Example 7. *All relators of the form Γ_T in Example 5 are relators for T. Proving that is quite standard, with the exception of the probabilistic case where the proof essentially relies on the Max Flow Min Cut Theorem [40].*

Definition 17. *Let T come with an ω***CPPO** *order $\sqsubseteq$. We say that $\Gamma\mathcal{R}$ is* inductive *if for any ω-chain $(u_n)_{n<\omega}$ in TX, we have:*

$$\perp\ \Gamma\mathcal{R}\ u \tag{ω-comp 1}$$

$$(\forall n.\ u_n\ \Gamma\mathcal{R}\ v) \implies \bigsqcup_n u_n\ \Gamma\mathcal{R}\ v. \tag{ω-comp 2}$$

We say that Γ respects Σ *if*

$$(\forall k.\ u_k\ \Gamma\mathcal{R}\ v_k) \implies \sigma(u_1, \ldots, u_n)\ \Gamma\mathcal{R}\ \sigma(v_1, \ldots, v_n) \tag{Σ-comp}$$

for any $\sigma \in \Sigma$, where $k \in \{1, \ldots, \alpha(\sigma)\}$.

Remark 6. *For a monad T carrying a continuous Σ-algebra structure and a function $f : X \to TY$, we required $f^\dagger : TX \to TY$ to be continuous, TX being an ω***CPPO***. Since TX is also a Σ-algebra, it seems natural to require $f^\dagger$ to be also a Σ-algebra homomorphism. In fact, such requirement implies condition (Σ-comp) and has the advantage of being more general than the latter, not depending from the specific relator considered. Let T be a monad on $\mathbb{SET}$. Following [38] we say that an n-ary algebraic operation (where n is some set) associates to each set X a function $\sigma_X : (TX)^n \to TX$ in such a way for every function $f : X \to TY$, the Kleisli extension $f^\dagger$ is a homomorphism. Recall that an n-ary generic effect is an element of Tn. As shown in [38], there is a bijection from generic effects to algebraic operations as follows. Every n-ary generic effect p gives rise to an n-ary algebraic operation $\hat{p}$, where $\hat{p}_X$ sends u to $u^\dagger(p)$. Conversely, each n-ary algebraic operation σ is $\hat{p}$ for a unique n-ary generic effect p, viz. $\sigma_n(\eta_n)$.*

We can now generalise our condition on T by requiring it to be equipped with an n-ary algebraic operation for each $\sigma \in \Sigma$ of arity n. This is the equivalent to extending our definitions by requiring the additional axiom that Kleisli extensions are homomorphisms. Moreover, requiring the bind operator to be strict in its first argument means that $\bot$ is an algebraic constant. Let us now prove that this condition implies condition (Σ-comp). For, suppose σ has arity n, and $\forall k.\ u_k\ \Gamma\mathcal{R}\ v_k$ holds, meaning that we have the square

$$\begin{tikzcd} n \arrow[r, "=_n", "/"{marking}] \arrow[d, "u"'] & n \arrow[d, "v"] \\ TX \arrow[r, "/"{marking}, "\Gamma\mathcal{R}"'] & TY \end{tikzcd}$$

As a consequence, we also have the square

$$\begin{tikzcd} Tn \arrow[r, "\Gamma(=_n)", "/"{marking}] \arrow[d, "u^\dagger"'] & Tn \arrow[d, "v^\dagger"] \\ TX \arrow[r, "/"{marking}, "\Gamma\mathcal{R}"'] & TY \end{tikzcd}$$

and therefore

$$\begin{tikzcd} Tn \arrow[r, "=_{Tn}", "/"{marking}] \arrow[d, "u^\dagger"'] & Tn \arrow[d, "v^\dagger"] \\ TX \arrow[r, "/"{marking}, "\Gamma\mathcal{R}"'] & TY \end{tikzcd}$$

Writing the algebraic operation associated with σ as $\hat{p}$, we have $p =_{Tn} p$, and so $u^\dagger(p)\ \Gamma\mathcal{R}\ v^\dagger(p)$, which essentially means

$$\sigma_X(u_1, ..., u_n)\ \Gamma\mathcal{R}\ \sigma_Y(v_1, ..., v_n).$$

To the ends of this paper, condition (Σ-comp) is sufficient and thus we will use that throughout.

Following Abramsky [1] we introduce Applicative Transition System (ATSs) over a monad (taking into account effectful computations) and define the notion of *applicative simulation*. Let T be a monad.

Definition 18. *An applicative transition system (over T) consists of the following:*
- *A state space made of a pair of sets (X, Y) modelling closed terms and values, respectively.*
- *An evaluation function $\varepsilon : X \to TY$.*
- *An application function $\cdot : Y \to Y \to X$.*

The notion of ATS distinguishes between terms and values. As a consequence, we often deal with pairs of relations $(\mathcal{R}_X, \mathcal{R}_Y)$, where $\mathcal{R}_X, \mathcal{R}_Y$ are relations over X and Y, respectively. We refer to such pairs as XY-relations. XY-relations belongs to $2^{X\times X} \times 2^{Y\times Y}$. The latter, being the product of complete lattices, is itself a complete lattice.

Definition 19. *Let Γ be a relator for T. An applicative Γ-simulation is an XY-relation $\mathcal{R} = (\mathcal{R}_X, \mathcal{R}_Y)$ such that:*

$$x \; \mathcal{R}_X x' \implies \varepsilon(x) \; \Gamma\mathcal{R}_Y \; \varepsilon(x') \tag{Sim-1}$$
$$y \; \mathcal{R}_Y y' \implies \forall w \in Y. \; y \cdot w \; \mathcal{R}_X \; y' \cdot w. \tag{Sim-2}$$

The above definition induces an operator $\mathcal{B}_\Gamma$ on $2^{X \times X} \times 2^{Y \times Y}$ defined for $\mathcal{R} = (\mathcal{R}_X, \mathcal{R}_Y)$ as $(\mathcal{B}_\Gamma(\mathcal{R}_X), \mathcal{B}_\Gamma(\mathcal{R}_Y))$, where

$$\mathcal{B}_\Gamma(\mathcal{R}_X) = \{(x, x') \mid \varepsilon(x) \; \Gamma\mathcal{R}_Y \; \varepsilon(x')\}$$
$$\mathcal{B}_\Gamma(\mathcal{R}_Y) = \{(y, y') \mid \forall w \in Y. \; y \cdot w \; \mathcal{R}_X \; y' \cdot w\}.$$

It is easy to prove that since Γ is monotone, then so is $\mathcal{B}_\Gamma$. As a consequence, we can define applicative Γ-similarity as the greatest fixed point $\nu\mathcal{B}_\Gamma$ of $\mathcal{B}_\Gamma$.

Proposition 2. *The following hold:*
1. *Applicative Γ-similarity $\precsim_\Gamma$ is a preorder.*
2. *If Γ is conversive, then $\precsim_\Gamma$ is an equivalence relation.*

Proof. The proof strictly follows the proof of Proposition **??** (proving the desired properties with respect to clause (Sim-2) is straightforward). As an example, we show by coinduction that $\precsim_\Gamma$ is reflexive by proving that the XY-identity relation $(=_X, =_Y)$ is an applicative Γ-simulation. From $x =_X x$ we infer $\varepsilon(x) =_{TY} \varepsilon(x)$, and thus $\varepsilon(x) \; \Gamma(=_Y) \; \varepsilon(x)$, by (Rel-1). Moreover, we trivially have that $y =_Y y$ implies $y \cdot w =_X y \cdot w$. $\square$

6 Contextual Preorder and Applicative Similarity

In the previous section, the axioms needed to generalise applicative bisimilarity to our setting have been given. What remains to be done is to appropriately instantiate all this to Λ_Σ. We introduce the notions of contextual preorder and applicative similarity (which will be then extended to contextual equivalence and applicative *bi*similarity). From now we assume to have a monad T carrying a continuous Σ-algebra structure. Moreover, we assume any relator for T to be inductive and to respect Σ. It is convenient to work with generalisations of relations on closed terms (resp. values) called λ-term relations.

Definition 20. *An open relation over terms is a set $\mathcal{R}_\Lambda$ of triples $(\bar{x}, M, N)$ where $M, N \in \Lambda(\bar{x})$. Similarly, an open relation over values is a set $\mathcal{R}_\mathcal{V}$ of triples $(\bar{x}, V, W)$ where $V, W \in \mathcal{V}(\bar{x})$. A λ-term relation is a pair $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$ made of an open relation $\mathcal{R}_\Lambda$ over terms and an open relation $\mathcal{R}_\mathcal{V}$ over values. A closed λ-term relation is a pair $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$ where $\mathcal{R}_\Lambda \subseteq \Lambda_0 \times \Lambda_0$ and similarly for $\mathcal{R}_\mathcal{V}$.*

Remark 7. *Formally, we can see an open relation over terms (and similarly over values) as an element of the cartesian product $\prod_{\bar{x}} 2^{\Lambda(\bar{x}) \times \Lambda(\bar{x})}$. That is, an open relation is a function that associates to each finite set $\bar{x}$ of variables a (binary) relation between open terms in $\Lambda(\bar{x})$. Since, $2^{\Lambda(\bar{x}) \times \Lambda(\bar{x})}$ is a complete lattice, for any finite set of variables $\bar{x}$, then so is $\prod_{\bar{x}} 2^{\Lambda(\bar{x}) \times \Lambda(\bar{x})}$. That is, the set of open relations over terms (and over values) forms a complete lattice (the order is given pointwise). As a consequence, the set of λ-term relations is a complete lattice as well. These algebraic properties allow us to define open relations both inductively and coinductively, and, in particular, to extend notions and results developed in the relational calculus of [27, 26, 17, 28].*

We will use infix notation and write $\bar{x} \vdash M \; \mathcal{R}_\Lambda \; N$ to indicate that $(\bar{x}, M, N) \in \mathcal{R}_\Lambda$. The same convention applies to values and open relations over values. For a λ-term relation $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$, we often write $\bar{x} \vdash M \; \mathcal{R} \; N$ (i.e. $(\bar{x}, M, N) \in \mathcal{R}$) for $\bar{x} \vdash M \; \mathcal{R}_\Lambda \; N$ (i.e. $(\bar{x}, M, N) \in \mathcal{R}_\Lambda$). The same convention holds for values and $\mathcal{R}_\mathcal{V}$. Finally, we will use the notations $\emptyset \vdash M \; \mathcal{R} \; N$ and $M \; \mathcal{R} \; N$ interchangeably (and similarly for values).

There is a canonical way to extend a closed relation to an open one.

$$
\begin{array}{lr}
\forall \bar{x}.\ \forall x \in \bar{x}.\ \bar{x} \vdash x\ \mathcal{R}_{\mathcal{V}}\ x & \text{(Comp1)} \\
\forall \bar{x}.\ \forall x \notin \bar{x}.\ \forall M, N.\ \bar{x} \cup \{x\} \vdash M\ \mathcal{R}_{\Lambda}\ N \implies \bar{x} \vdash \lambda x.M\ \mathcal{R}_{\mathcal{V}}\ \lambda x.N & \text{(Comp2)} \\
\forall \bar{x}.\ \forall V, W\ \bar{x} \vdash V\ \mathcal{R}_{\mathcal{V}}\ W \implies \bar{x} \vdash \mathsf{return}\ V\ \mathcal{R}_{\Lambda}\ \mathsf{return}\ W & \text{(Comp3)} \\
\forall \bar{x}.\ \forall V, V', W, W'.\ \bar{x} \vdash V\ \mathcal{R}_{\mathcal{V}}\ V' \wedge \bar{x} \vdash W\ \mathcal{R}_{\mathcal{V}}\ W' \implies \bar{x} \vdash VW\ \mathcal{R}_{\Lambda}\ V'W' & \text{(Comp4)} \\
\forall \bar{x}.\ \forall x \notin \bar{x}.\ \forall M, M', N, N'. & \\
\bar{x} \vdash M\ \mathcal{R}_{\Lambda}\ M' \wedge \bar{x} \cup \{x\} \vdash N\ \mathcal{R}_{\Lambda}\ N' \implies \bar{x} \vdash (M\ \mathsf{to}\ x.N)\ \mathcal{R}_{\Lambda}\ (M'\ \mathsf{to}\ x.N') & \text{(Comp5)} \\
\forall \bar{x}.\ \forall \sigma \in \Sigma.\ \forall M_1, N_1, \ldots, M_n, N_n. & \\
(\forall i \in \{1, \ldots, n\}.\ \bar{x} \vdash M_i\ \mathcal{R}_{\Lambda}\ N_i) \implies \bar{x} \vdash \sigma(M_1, \ldots, M_n)\ \mathcal{R}_{\Lambda}\ \sigma(N_1, \ldots, N_n) & \text{(Comp6)}
\end{array}
$$

Figure 4: Compatibility Clauses.

Definition 21. *Define the open extension operator mapping a closed relation over terms $\mathcal{R}$ to the open relation $\mathcal{R}^\circ$ (over terms) as follows: $(\bar{x}, M, N) \in \mathcal{R}^\circ$ iff $M, N \in \Lambda(\bar{x})$, and for all $\bar{V}$, $M[\bar{x} := \bar{V}]\ \mathcal{R}\ N[\bar{x} := \bar{V}]$ holds.*

The notion of open extension for a closed relation over values can be defined in a similar way (using the appropriate notion of substitution).

The notion of reflexivity, symmetry and transitivity straightforwardly extends to open λ-term relation (see e.g. [35]).

Definition 22. *Let $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$ be a λ-term relation. We say that $\mathcal{R}$ is* compatible *if the clauses in Figure 4 hold. We say that $\mathcal{R}$ is a* precongruence *if it is a compatible preorder. We say that $\mathcal{R}$ is a* congruence *if it is a compatible equivalence.*

The following lemma will be useful.

Lemma 9. *Let $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$ be a λ-term relation. If $\mathcal{R}$ is a preorder, then properties* (Comp4), (Comp5), (Comp6) *are equivalent to their 'unidirectional' versions:*

$$
\begin{array}{lr}
\forall \bar{x}.\ \forall V, V', W.\ \bar{x} \vdash V\ \mathcal{R}_{\mathcal{V}}\ V' \implies \bar{x} \vdash VW\ \mathcal{R}_{\Lambda}\ V'W & \text{(Comp4L)} \\
\forall \bar{x}.\ \forall V, W, W'.\ \bar{x} \vdash W\ \mathcal{R}_{\mathcal{V}}\ W' \implies \bar{x} \vdash VW\ \mathcal{R}_{\Lambda}\ VW' & \text{(Comp4R)} \\
\forall \bar{x}.\ \forall x \notin \bar{x}.\ \forall M, M', N.\ \bar{x} \vdash M\ \mathcal{R}_{\Lambda}\ M' \implies \bar{x} \vdash (M\ \mathsf{to}\ x.N)\ \mathcal{R}_{\Lambda}\ (M'\ \mathsf{to}\ x.N) & \text{(Comp5L)} \\
\forall \bar{x}.\ \forall x \notin \bar{x}.\ \forall M, N, N'.\ \bar{x} \cup \{x\} \vdash N\ \mathcal{R}_{\Lambda}\ N' \implies \bar{x} \vdash (M\ \mathsf{to}\ x.N)\ \mathcal{R}_{\Lambda}\ (M\ \mathsf{to}\ x.N') & \text{(Comp5R)} \\
\forall \bar{x}.\ \forall \sigma \in \Sigma.\ \forall M, N, \bar{M}, \bar{N}.\ \bar{x} \vdash M\ \mathcal{R}_{\Lambda}\ N \implies \bar{x} \vdash \sigma(\bar{M}, M, \bar{N})\ \mathcal{R}_{\Lambda}\ \sigma(\bar{M}, N, \bar{N}) & \text{(Comp6C)}
\end{array}
$$

where in (Comp6C) *$\bar{M}, \bar{N}$ are possibly empty finite tuples of terms such that the sum of their lengths is equal to the ariety of σ minus one.*

Proof. The proof is straightforward. As a paradigmatic example, we show that clause (Comp5) is equivalent to the conjunction of clauses (Comp5L) and (Comp5R). For the left to right implication, we assume that both (Comp5) and $\bar{x} \vdash M\ \mathcal{R}_\Lambda\ M'$ hold, and show that $\bar{x} \vdash M\ \mathsf{to}\ x.N\ \mathcal{R}_\Lambda\ M'\ \mathsf{to}\ x.N$ holds as well, thus proving that (Comp5) implies (Comp5L) (the proof that (Comp5) implies (Comp5R) is morally the same). To prove the thesis, we observe that since $\mathcal{R}$ is reflexive, we have $\bar{x} \cup \{x\} \vdash N\ \mathcal{R}_\Lambda\ N$. Applying (Comp5) to the latter and $\bar{x} \vdash M\ \mathcal{R}_\Lambda\ M'$, we conclude $\bar{x} \vdash M\ \mathsf{to}\ x.N\ \mathcal{R}_\Lambda\ M'\ \mathsf{to}\ x.N$.
Now for the right to left direction. Assume (Comp5L) and (Comp5R) to be valid, and suppose both $\bar{x} \vdash M\ \mathcal{R}_\Lambda\ M'$ and $\bar{x} \cup \{x\} \vdash N\ \mathcal{R}_\Lambda\ N'$ to hold. We can apply (Comp5L) to the former, obtaining $\bar{x} \vdash M\ \mathsf{to}\ x.N\ \mathcal{R}_\Lambda\ M'\ \mathsf{to}\ x.N$, and (Comp5R) to the latter, obtaining $\bar{x} \vdash M'\ \mathsf{to}\ x.N\ \mathcal{R}_\Lambda\ M'\ \mathsf{to}\ x.N'$. The thesis now follows by transitivity of $\mathcal{R}$. $\square$

It is useful to characterise compatible relations via the notion of compatible refinement.

$$\frac{}{\bar{x} \vdash x\ \widehat{\mathcal{R}}_{\mathcal{V}}\ x}\ x \in \bar{x} \qquad \frac{\bar{x} \cup \{x\} \vdash M\ \mathcal{R}_\Lambda\ N}{\bar{x} \vdash \lambda x.M\ \widehat{\mathcal{R}}_{\mathcal{V}}\ \lambda x.N}\ x \notin \bar{x}$$

$$\frac{\bar{x} \vdash V\ \mathcal{R}_{\mathcal{V}}\ W}{\bar{x} \vdash \mathsf{return}\ V\ \mathcal{R}_\Lambda\ \mathsf{return}\ W} \qquad \frac{\bar{x} \vdash V\ \mathcal{R}_{\mathcal{V}}\ V' \qquad \bar{x} \vdash W\ \mathcal{R}_{\mathcal{V}}\ W'}{\bar{x} \vdash VW\ \widehat{\mathcal{R}}_\Lambda\ V'W'}$$

$$\frac{\bar{x} \vdash M\ \mathcal{R}_\Lambda\ M' \qquad \bar{x} \cup \{x\} \vdash N\ \mathcal{R}_\Lambda\ N'}{\bar{x} \vdash M\ \mathsf{to}\ x.N\ \widehat{\mathcal{R}}_\Lambda\ M'\ \mathsf{to}\ x.N'}\ x \notin \bar{x} \qquad \frac{\bar{x} \vdash M_1\ \mathcal{R}_\Lambda\ N_1 \quad \ldots \quad \bar{x} \vdash M_n\ \mathcal{R}_\Lambda\ N_n}{\bar{x} \vdash \sigma(M_1, \ldots, M_n)\ \widehat{\mathcal{R}}_\Lambda\ \sigma(N_1, \ldots, N_n)}$$

Figure 5: Compatible Refinement Rules.

Definition 23. *Let $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$ be a λ-term relation. Define the compatible refinement $\widehat{\mathcal{R}}$ of $\mathcal{R}$ as the pair $(\widehat{\mathcal{R}}_\Lambda, \widehat{\mathcal{R}}_\mathcal{V})$, where $\mathcal{R}_\Lambda$ and $\mathcal{R}_\mathcal{V}$ are inductively defined by rules in Figure 5.*

Proposition 3. *A λ-term relation $\mathcal{R}$ is compatible iff $\widehat{\mathcal{R}} \subseteq \mathcal{R}$ holds.*

The above notion of precongruence can be justified by observing that when a relation $\mathcal{R}$ is a preorder, being a precongruence does exactly mean to be closed under the term constructors of the language. That could be formally expressed by saying that $\mathcal{R}$ is a precongruence if and only if $\bar{x} \vdash M\ \mathcal{R}\ N$ implies $\bar{x} \vdash C[M]\ \mathcal{R}\ C[N]$, for any term context $C[\cdot]$. Defining term contexts requires some care. In particular, when dealing with the contextual preorder it is not possible to reason modulo α-conversion, thus making definition syntactically involved (see [27, 26, 35] for details). As remarked in [35], it is possible to avoid those difficulties by giving a coinductive characterisation of the contextual preorder in the style of [27, 17]. Essentially, the contextual preorder (and, similarly the contextual equivalence) is defined as the largest compatible and preadequate (see Definition 24) λ-term relation. It is then easy to provide a more syntactic definition of contextual preorder and to prove that the two given definitions are equivalent [17, 27, 35].

The notion of adequacy defines the available observation on values. Being in an untyped setting, it is customary not to observe them.

Definition 24. *Let $\mathcal{U}$ denote $\mathcal{V}_0 \times \mathcal{V}_0$ seen as a closed relation, i.e. the trivial relation relating all values. We say that a relation $\mathcal{R}$ on terms is preadequate if*

$$\emptyset \vdash M\ \mathcal{R}\ N \implies [\![M]\!]\ \Gamma\mathcal{U}\ [\![N]\!]$$

where $M, N \in \Lambda_0$. That is, a relation $\mathcal{R}$ on terms is preadequate if whenever $\mathcal{R}$ relates two closed terms, evaluating these programs produces the same side-effects. A λ-term relation $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$ is preadequate iff $\mathcal{R}_\Lambda$ is.

Example 8. *It is easy to check that the above notion of adequacy (together with the relators in Example 5) captures standard notions of adequacy used for untyped λ-calculi.*

- *Consider a calculus without operation symbols and with operational semantics over $(\mathcal{V}_0)_\perp$. A relation is preadequate if whenever $\emptyset \vdash M\ \mathcal{R}\ N$, then if M converges, then so does N.*
- *Consider a nondeterministic calculus with operational semantics over $\mathcal{P}\mathcal{V}_0$. A relation is preadequate if whenever $\emptyset \vdash M\ \mathcal{R}\ N$, then if there exists a value V to which M may converge (i.e. $V \in [\![M]\!]$), then there exists a value W to which N may converge (i.e. $W \in [\![N]\!]$).*
- *Consider a probabilistic calculus with operational semantics over $\mathcal{D}\mathcal{V}_0$. A relation is preadequate if whenever $\emptyset \vdash M\ \mathcal{R}\ N$, then the probability of convergence of M is smaller or equal than the probability of convergence of N.*

Following [27], we shall define the Γ-*contextual preorder* as the largest λ-term relation that is both compatible and preadequate.

Definition 25. *Let $\mathbb{CA}$ be the set of relations on terms that are both compatible and preadequate. Then define $\leq_\Gamma$ as $\bigcup \mathbb{CA}$.*

Proposition 4. *The Γ-contextual preorder $\leq_\Gamma$ is a compatible and preadequate preorder.*

Proof. We prove that $\leq_\Gamma \in \mathbb{CA}$. First of all note that $\mathbb{CA}$ contains the open identity relation. In fact, the latter is clearly compatible. To see it is also preadequate suppose $\emptyset \vdash M =_{\Lambda_0} M$ so that $[\![M]\!] =_{T\mathcal{V}_0} [\![M]\!]$. By (Rel-1), we have $=_{T\mathcal{V}_0} \subseteq \Gamma(=_{\mathcal{V}_0})$. Moreover, by very definition of $\mathcal{U}$, we also have $=_{\mathcal{V}_0} \subseteq \mathcal{U}$ so that we can conclude $[\![M]\!]\ \Gamma\mathcal{U}\ [\![M]\!]$, by monotonicity of Γ. As a consequence, $\leq \Gamma$ satisfies (Comp1). Observe also that (Comp1) implies, in particular, reflexivity of $\leq_\Gamma$.
We now show that $\leq_\Gamma$ satisfies (Comp2). Suppose $(\bar{x} \cup \{x\}, M, N) \in \leq_\Gamma$. That means there exists a λ-term relation $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V}) \in \mathbb{CA}$ such that $(\bar{x} \cup \{x\}, M, N) \in \mathcal{R}_\Lambda$. Since $\mathcal{R}$ is compatible it satisfies (Comp2), and thus we have $(\bar{x}, \lambda x.M, \lambda y.N) \in \mathcal{R}_\mathcal{V}$. It then follows $(\bar{x}, \lambda x.M, \lambda x.N) \in \leq_\Gamma$ (i.e. in its value component). Similarly, we can prove that $\leq_\Gamma$ satisfies (Comp3).

This approach does not work neither for (Comp4), (Comp5) nor for (Comp6). The reason is that all these clauses are multiple premises implications (and that badly interacts with the existential information obtained from being in $\leq_\Gamma$). Nonetheless, we can appeal to Lemma 9 to replace clauses (Comp4)-(Comp6) to single premiss implications (for which the proof works as for previous compatibility conditions). In order to use Lemma 9, we need to prove that $\leq_\Gamma$ is transitive, and thus a preorder. For, it is sufficient to prove that $\mathbb{CA}$ is closed under relation composition. The proof is rather standard and we just prove a couple of cases as examples.

We first show that if $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\Lambda)$ and $\mathcal{S} = (\mathcal{S}_\Lambda, \mathcal{S}_\mathcal{V})$ are preadequate, then so is $\mathcal{S} \circ \mathcal{R} = (\mathcal{S}_\Lambda \circ \mathcal{R}_\Lambda, \mathcal{S}_\mathcal{V} \circ \mathcal{R}_\mathcal{V})$. Suppose $\emptyset \vdash M\ \mathcal{R}_\Lambda\ L$ and $\emptyset \vdash L\ \mathcal{S}_\Lambda\ N$. Since both $\mathcal{R}$ and $\mathcal{S}$ are preadequate, we have $[\![M]\!]\ \Gamma\mathcal{U}\ [\![L]\!]$ and $[\![L]\!]\ \Gamma\mathcal{U}\ [\![N]\!]$. By very definition of relator we have $\Gamma\mathcal{U} \circ \Gamma\mathcal{U} \subseteq \Gamma(\mathcal{U} \circ \mathcal{U})$. The latter is contained in $\Gamma\mathcal{U}$, since Γ is monotone and we trivially have $\mathcal{U} \circ \mathcal{U}$.

Proving that the composition of compatible relations is compatible is a straightforward exercise. For instance, we show that if relations $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V}), \mathcal{S} = (\mathcal{S}_\Lambda, \mathcal{S}_\mathcal{V})$ satisfy (Comp5), then so does $\mathcal{S} \circ \mathcal{R}$. For, suppose $\bar{x} \vdash M\ (\mathcal{S}_\Lambda \circ \mathcal{R}_\Lambda)\ M'$ and $\bar{x} \cup \{x\} \vdash N\ (\mathcal{S}_\Lambda \circ \mathcal{R}_\Lambda)\ N'$. As a consequence, we have

$$\bar{x} \vdash M\ \mathcal{R}_\Lambda\ M'' \tag{1}$$
$$\bar{x} \vdash M''\ \mathcal{S}_\Lambda\ M' \tag{2}$$
$$\bar{x} \cup \{x\} \vdash N\ \mathcal{R}_\Lambda\ N'' \tag{3}$$
$$\bar{x} \cup \{x\} \vdash N''\ \mathcal{S}_\Lambda\ N'. \tag{4}$$

From (1) and (3) we infer $\bar{x} \vdash M \texttt{ to } x.N\ \mathcal{R}_\Lambda\ M'' \texttt{ to } x.N''$, since $\mathcal{R}$ satisfies (Comp5). Similarly, from (2) and (4) we infer $\bar{x} \vdash M'' \texttt{ to } x.N''\ \mathcal{S}_\Lambda\ M' \texttt{ to } x.N'$. We can conclude $\bar{x} \vdash M \texttt{ to } x.N\ (\mathcal{S}_\Lambda \circ \mathcal{R}_\Lambda)\ M' \texttt{ to } x.N'$. ☐

Finally, we define the notion of an *applicative* Γ*-simulation* observing that the collection of closed terms and values, together with the operational semantics defined in previous section, carries an ATS structure.

Definition 26. *A closed relation $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$ respects values if for all closed values V, W, $V\ \mathcal{R}_\mathcal{V}\ W$ implies $VU\ \mathcal{R}_\Lambda\ WU$, for any closed value U.*

Definition 27. *Define the ATS of closed λ-terms as follows:*
- *The state space is given by the pair $(\Lambda_0, \mathcal{V}_0)$;*
- *The evaluation function is $[\![\cdot]\!] : \Lambda_0 \to T\mathcal{V}_0$;*
- *The application function $\cdot : \mathcal{V}_0 \to \mathcal{V}_0 \to \Lambda_0$ is defined as term application: $V \cdot W = VW$.*

As a consequence, we can apply the general definition of applicative Γ-simulation to the ATS of λ-terms. Instantiating the general definition of applicative Γ-simulation we obtain:

Definition 28. *Let Γ be a relator for the monad T. A closed relation $\mathcal{R} = (\mathcal{R}_\Lambda, \mathcal{R}_\mathcal{V})$ is an applicative* Γ*-simulation if:*

- $M \; \mathcal{R}_\Lambda \; N \implies [\![M]\!] \; \Gamma\mathcal{R}_\mathcal{V} \; [\![N]\!]$;
- $\mathcal{R}$ *respects values.*

We can then define applicative Γ-similarity $\precsim_\Gamma$ as the largest applicative Γ-simulation, which we know to be a preorder by Proposition 2. Most of the time the relator Γ will be fixed; in those cases we will often write $\precsim$ in place of $\precsim_\Gamma$.

Example 9. *It is immediate to see that using the relators in Example 5 we recover well-known notions of simulation and bisimulation.*

We want to prove that applicative similarity is a sound proof technique for contextual preorder. That is, we want to prove that $\precsim_\Gamma \subseteq \leq_\Gamma$ holds. The relation $\leq_\Gamma$ being defined as the largest preadequate compatible relation, the above inclusion is established by proving that $\precsim_\Gamma$ is a precongruence.

7 Howe's Method and Its Soundness

In this section we generalise Howe's technique to show that applicative similarity is a precongruence, thus a sound proof technique for the contextual preorder. Our generalisation shows how Howe's method crucially (but only!) depends on the structure of the monad modelling side-effects and the relators encoding their associated notion of observation.

Definition 29. *Let $\mathcal{R}$ be a closed λ-term relation. The Howe extension $\mathcal{R}^H$ of $\mathcal{R}$ is defined as the least relation $\mathcal{S}$ such that $\mathcal{S} = \mathcal{R}^\circ \circ \widehat{\mathcal{S}}$.*

It was observed in [28] that the above equation actually defines a unique relation.

Lemma 10. *Let $\mathcal{R}$ be a closed λ-term relation. Then there is a unique relation $\mathcal{S}$ such that $\mathcal{S} = \mathcal{R}^\circ \circ \widehat{\mathcal{S}}$.*

As a consequence, $\mathcal{R}^H$ can be characterised both inductively and coinductively. Here we give two (well-known) equivalent inductive characterisations of $\mathcal{R}^H$.

Lemma 11. *The following are equivalent and all define the relation $\mathcal{R}^H$.*

1. *The Howe extension $\mathcal{R}^H = (\mathcal{R}^H_\Lambda, \mathcal{R}^H_\mathcal{V})$ of $\mathcal{R}$ is defined as the least relation closed under the following rules:*

$$\frac{\bar{x} \vdash M \; \widehat{\mathcal{R}^H_\Lambda} \; L \qquad \bar{x} \vdash L \; \mathcal{R}^\circ_\Lambda \; N}{\bar{x} \vdash M \; \mathcal{R}^H_\Lambda \; N} \qquad \frac{\bar{x} \vdash V \; \widehat{\mathcal{R}^H_\mathcal{V}} \; U \qquad \bar{x} \vdash U \; \mathcal{R}^\circ_\mathcal{V} \; W}{\bar{x} \vdash V \; \mathcal{R}^H_\mathcal{V} \; W}$$

2. *The Howe extension $\mathcal{R}^H$ of $\mathcal{R}$ is the relation inductively defined by rules in Figure 6.*

Proof. It is easy to see that the functional $\mathcal{F}$ on λ-term relations associated to Definition 29 (i.e. defined by $\mathcal{F}(\mathcal{S}) = \mathcal{R}^\circ \circ \widehat{\mathcal{S}}$) is also the functional induced by rules in point 1. We can prove by induction the equivalence between the relations defined in point 1 and point 2 (in fact, these are both defined inductively). This is tedious but easy, and thus the proof is omitted. $\square$

The following lemma states some nice properties of Howe's lifting of preorder relations. The proof is standard and can be found in, e.g., [10].

Lemma 12. *Let $\mathcal{R}$ be a preorder. The following hold:*

1. $\mathcal{R} \circ \mathcal{R}^H \subseteq \mathcal{R}^H$.
2. $\mathcal{R}^H$ *is compatible, and thus reflexive.*
3. $\mathcal{R} \subseteq \mathcal{R}^H$.

Remark 8. *To prove properties 2 and 3 it is actually sufficient to require $\mathcal{R}$ to be reflexive, whereas property 1, which we refer to transitivity of $\mathcal{R}^H$ wrt $\mathcal{R}$, requires $\mathcal{R}$ to be transitive. It is easy to see that a compatible relation is also reflexive.*

$$\frac{\bar{x} \vdash x \ \mathcal{R}^{\circ}_{\mathcal{V}} \ V}{\bar{x} \vdash x \ \mathcal{R}^{H}_{\mathcal{V}} \ V} \ (\mathsf{How1})$$

$$\frac{\bar{x} \cup \{x\} \vdash M \ \mathcal{R}^{H}_{\Lambda} \ L \qquad \bar{x} \vdash \lambda x.L \ \mathcal{R}^{\circ}_{\mathcal{V}} \ V}{\bar{x} \vdash \lambda x.M \ \mathcal{R}^{H}_{\mathcal{V}} \ V} \ (\mathsf{How2})$$

$$\frac{\bar{x} \vdash V \ \mathcal{R}^{H}_{\mathcal{V}} \ W \qquad \bar{x} \vdash \mathsf{return}\ W \ \mathcal{R}^{\circ}_{\Lambda} \ N}{\bar{x} \vdash \mathsf{return}\ V \ \mathcal{R}^{H}_{\Lambda} \ N} \ (\mathsf{How3})$$

$$\frac{\bar{x} \vdash V \ \mathcal{R}^{H}_{\mathcal{V}} \ V' \qquad \bar{x} \vdash W \ \mathcal{R}^{H}_{\mathcal{V}} \ W' \qquad \bar{x} \vdash V'W' \ \mathcal{R}^{\circ}_{\Lambda} \ N}{\bar{x} \vdash VW \ \mathcal{R}^{H}_{\Lambda} \ N} \ (\mathsf{How4})$$

$$\frac{\bar{x} \vdash M \ \mathcal{R}^{H}_{\Lambda} \ L \qquad \bar{x} \cup \{x\} \vdash M' \ \mathcal{R}^{H}_{\Lambda} \ L' \qquad \bar{x} \vdash L \ \mathsf{to}\ x.L' \ \mathcal{R}^{\circ}_{\Lambda} \ N}{\bar{x} \vdash M \ \mathsf{to}\ x.M' \ \mathcal{R}^{H}_{\Lambda} \ N} \ (\mathsf{How5})$$

$$\frac{\bar{x} \vdash M_k \ \mathcal{R}^{H}_{\Lambda} \ N_k \ (\forall k \geq n) \qquad \bar{x} \vdash \sigma(N_1, \ldots, N_n) \ \mathcal{R}^{\circ}_{\Lambda} \ N}{\bar{x} \vdash \sigma(M_1, \ldots, M_n) \ \mathcal{R}^{H}_{\Lambda} \ N} \ (\mathsf{How6})$$

Figure 6: Howe's Extension Rules.

We now consider the Howe extension $\precsim^{H}_{\Gamma}$ of applicative Γ-similarity. Since $\precsim_{\Gamma}$ is a preorder (Proposition 2), $\precsim^{H}_{\Gamma}$ is a compatible relation containing $\precsim_{\Gamma}$.

Definition 30. *A λ-term relation $\mathcal{R} = (\mathcal{R}_{\Lambda}, \mathcal{R}_{\mathcal{V}})$ is value-substitutive if $x \vdash M \ \mathcal{R}_{\Lambda} \ N$ and $\emptyset \vdash V \ \mathcal{R}_{\mathcal{V}} \ W$ imply $\emptyset \vdash M[x := V] \ \mathcal{R}_{\Lambda} \ N[x := W]$.*

Lemma 13. *The relation $\precsim^{H}_{\Gamma}$ is value-substitutive.*

Proof. The proof is standard, see e.g. [10]. □

Summing up, we have defined a compatible relation $\precsim^{H}_{\Gamma}$ which is value-substitutive and contains $\precsim_{\Gamma}$. As a consequence, to prove that the latter is compatible it is sufficient to prove $\precsim^{H}_{\Gamma} \subseteq \precsim_{\Gamma}$. We can proceed coinductively, showing that $\precsim^{H}_{\Gamma}$ is an applicative Γ-simulation. This is proved via the so-called Key Lemma. Before proving the Key Lemma it is useful to spell out basic facts on the Howe extension of applicative similarity that we will extensively use. In the following we assume to have fixed a relator Γ, thus omitting subscripts. Let Γ be a relator.

Lemma 14. *The following hold:*
1. $\precsim \circ \precsim^{H} \subseteq \precsim^{H}$.
2. $(\Gamma \precsim) \circ (\Gamma \precsim^{H}) \subseteq \Gamma \precsim^{H}$.

Lemma 15 (Key Lemma)**.** *Let $\precsim^{H} = (\precsim^{H}_{\Lambda}, \precsim^{H}_{\mathcal{V}})$ be the Howe extension of applicative similarity. If $\emptyset \vdash M \precsim^{H}_{\Lambda} N$ and $M \Downarrow_n X$, then $X \ \Gamma \precsim^{H}_{\mathcal{V}} \ [\![N]\!]$.*

Proof. We proceed by induction on the derivation of the judgment $M \Downarrow_n X$.

Case (bot)**.** Suppose to have $M \Downarrow_0 \bot$. We are done since Γ is inductive, and thus $\bot \ \Gamma \precsim^{H}_{\mathcal{V}} \ [\![N]\!]$ trivially holds (see property (ω-comp 2)).

Case (ret)**.** Suppose to have $\mathsf{return}\ V \Downarrow_{n+1} \eta(V)$. By hypothesis we have $\emptyset \vdash \mathsf{return}\ V \precsim^{H}_{\Lambda} N$, so that the latter must have been obtained as the conclusion of an instance of rule (How3). As a consequence, we have $\emptyset \vdash V \precsim^{H}_{\mathcal{V}} W$ and $\mathsf{return}\ W \precsim_{\Lambda} N$, for some value W. We can now appeal to (Lax-Unit), thus inferring $\eta(V) \ \Gamma \precsim^{H}_{\mathcal{V}} \ \eta(W)$ from $\emptyset \vdash V \precsim^{H}_{\mathcal{V}} W$. By very definition of applicative similarity, $\mathsf{return}\ W \precsim_{\Lambda} N$ implies $[\![\mathsf{return}\ W]\!] \ \Gamma \precsim_{\mathcal{V}} \ [\![N]\!]$, i.e.

$\eta(W)\ \Gamma \precsim_{\mathcal{V}}\ [\![N]\!]$. Therefore, we have $\eta(V)\ (\Gamma \precsim_{\mathcal{V}}) \circ (\Gamma \precsim^{H}_{\mathcal{V}})\ [\![N]\!]$, from which the thesis follows by Lemma 14.

Case (app)**.** Suppose the judgment $(\lambda x.M)V \Downarrow_{n+1} X$ has been obtained from the judgment $M[x := V] \Downarrow_n X$. By hypothesis we have $\emptyset \vdash (\lambda x.M)V \precsim^{H}_{\Lambda} N$, meaning that the latter must have been obtained as the conclusion of an instance of (How4). We thus obtain $\emptyset \vdash \lambda x.M \precsim^{H}_{\mathcal{V}} W$, $\emptyset \vdash V \precsim^{H}_{\mathcal{V}} U$ and $WU \precsim_{\Lambda} N$, for values W, U. Looking at the first of these three judgments, we see that it must be the conclusion of an instance of rule (How2). Therefore, we have $\{x\} \vdash M \precsim^{H}_{\Lambda} L$ and $\lambda x.L \precsim_{\mathcal{V}} W$. Since $\precsim^{H}$ is value-substitutive, from $\{x\} \vdash M \precsim^{H}_{\Lambda} L$ and $\emptyset \vdash V \precsim^{H}_{\mathcal{V}} U$ we conclude $\emptyset \vdash M[x := V] \precsim^{H}_{\Lambda} L[x := U]$. We can now apply the induction hypothesis on the latter and $M[x := V] \Downarrow_n X$, obtaining $X\ \Gamma(\precsim^{H}_{\mathcal{V}})\ [\![L[x := U]]\!]$. Since $\precsim$ respects values, from $\lambda x.L \precsim_{\mathcal{V}} W$ we infer $(\lambda x.L)U \precsim_{\Lambda} WU$, which gives, by very definition of applicative similarity, $[\![(\lambda x.L)U]\!]\ \Gamma(\precsim_{\mathcal{V}})\ [\![WU]\!]$. By Lemma 6, $[\![(\lambda x.L)U]\!] = [\![L[x := U]]\!]$, and thus, $X\ \Gamma(\precsim^{H}_{\mathcal{V}})\ [\![WU]\!]$, by Lemma 14. Finally, from $WU \precsim_{\mathcal{V}} N$ we obtain $[\![WU]\!]\ \Gamma(\precsim_{\mathcal{V}})\ [\![N]\!]$, which allows us to conclude $X\ \Gamma(\precsim_{\mathcal{V}})\ [\![N]\!]$ by Lemma 14.

Case (seq)**.** Suppose the judgment $(M\ \mathsf{to}\ x.M') \Downarrow_{n+1} X \mathbin{>\!\!>\!=} (V \mapsto Y_V)$ has been obtained from $M \Downarrow_n X$ and $M'[x := V] \Downarrow_n Y_V$. By hypothesis we have $\emptyset \vdash M\ \mathsf{to}\ x.M' \precsim^{H}_{\Lambda} N$, which must have been obtained via an instance of rule (How5) thus giving $\emptyset \vdash M \precsim^{H}_{\Lambda} L$, $\{x\} \vdash M' \precsim^{H}_{\Lambda} L'$ and $\emptyset \vdash L\ \mathsf{to}\ x.L' \precsim_{\Lambda} N$. We can apply the induction hypothesis on $M \Downarrow_n X$ and $\emptyset \vdash M \precsim^{H}_{\Lambda} L$ obtaining $X\ \Gamma(\precsim^{H}_{\mathcal{V}})\ [\![L]\!]$. We now claim to have

$$X \mathbin{>\!\!>\!=} (V \mapsto Y_V)\ \Gamma(\precsim^{H}_{\mathcal{V}})\ [\![L]\!] \mathbin{>\!\!>\!=} (V \mapsto [\![L'[x := V]]\!]).$$

The latter is equal to $[\![L\ \mathsf{to}\ x.L']\!]$, by Lemma 6. Besides, $\emptyset \vdash L\ \mathsf{to}\ x.L' \precsim_{\Lambda} N$ entails $[\![L\ \mathsf{to}\ x.L']\!]\ \Gamma(\precsim_{\mathcal{V}})\ [\![N]\!]$: we conclude $X \mathbin{>\!\!>\!=} (V \mapsto Y_V)\ \Gamma(\precsim^{H}_{\mathcal{V}})\ [\![N]\!]$, by Lemma 14.
The above claim directly follows from (Lax-Bind). In fact, since $X\ \Gamma(\precsim^{H}_{\mathcal{V}})\ [\![L]\!]$ holds, by (Lax-Bind) it is sufficient to prove that $V \precsim^{H}_{\mathcal{V}} W$ implies $Y_V\ \Gamma(\precsim^{H}_{\mathcal{V}})\ [\![L'[x := W]]\!]$. Assume $V \precsim^{H}_{\mathcal{V}} W$, i.e. $\emptyset \vdash V \precsim^{H}_{\mathcal{V}} W$. The latter, together with $\{x\} \vdash M' \precsim^{H}_{\Lambda} L'$, implies $\emptyset \vdash M'[x := V] \precsim^{H}_{\Lambda} L'[x := W]$, since $\precsim^{H}$ is value-substitutive. We can finally apply the inductive hypothesis on the latter and $M'[x := V] \Downarrow_n Y_V$, thus concluding the wanted thesis.

Case (op)**.** Suppose the judgment $\sigma(M_1, \ldots, M_k) \Downarrow_{n+1} \sigma^T(X_1, \ldots, X_k)$ has been obtained from $M_1 \Downarrow_n X_1, \ldots, M_k \Downarrow_n X_k$. By hypothesis we have $\emptyset \vdash \sigma(M_1, \ldots, M_k) \precsim^{H}_{\Lambda} N$, which must be the conclusion of an instance of rule (How6). As a consequence, judgments $\emptyset \vdash M_1 \precsim^{H}_{\Lambda} N_1, \ldots, \emptyset \vdash M_k \precsim^{H}_{\Lambda} N_k$ and $\emptyset \vdash \sigma(N_1, \ldots, N_k) \precsim_{\Lambda} N$ hold, for some terms $N_1, \ldots, N_k$. We can repeatedly apply the induction hypothesis on $M_i \Downarrow_n X_i$ and $\emptyset \vdash M_i \precsim^{H}_{\Lambda} N_i$, inferring $X_i\ \Gamma \precsim^{H}_{\mathcal{V}}\ [\![N_i]\!]$, for all $i \in \{1, \ldots, k\}$. (Σ-comp) allows to conclude $\sigma^T(X_1, \ldots, X_k)\ \Gamma \precsim^{H}_{\mathcal{V}}\ \sigma^T([\![N_1]\!], \ldots, [\![N_k]\!])$. By Lemma 6 the latter is equal to $[\![\sigma(N_1, \ldots, N_k)]\!]$. Finally, from $\emptyset \vdash \sigma(N_1, \ldots, N_k) \precsim_{\Lambda} N$ we infer $[\![\sigma(N_1, \ldots, N_k)]\!]\ \Gamma \precsim_{\mathcal{V}}\ [\![N]\!]$ from which the thesis follows by Lemma 14.

□

Corollary 2. *The relation $\precsim^{H}_{\Gamma}$ is an applicative Γ-simulation.*

Proof. Suppose $M \precsim^{H}_{\Lambda} N$. We have to prove $[\![M]\!]\ \Gamma \precsim^{H}_{\mathcal{V}}\ [\![N]\!]$, i.e. $\bigsqcup_{M \Downarrow_n X} X\ \Gamma(\precsim^{H}_{\mathcal{V}})\ [\![N]\!]$. The latter follows from (ω-comp 1) by the Key Lemma. Finally, since $\precsim^{H}$ is compatible, it clearly respects values. □

Theorem 1. *Similarity is a precongruence. Moreover, it is sound for contextual preorder $\leq_{\Gamma}$.*

Proof. We already know $\precsim_{\Gamma}$ is a preorder. By previous corollary it follows that $\precsim_{\Gamma}$ coincides with $\precsim^{H}_{\Gamma}$, so that $\precsim_{\Gamma}$ is also compatible, and thus a precongruence. Now for soundness. We have to prove $\precsim_{\Gamma} \subseteq \leq_{\Gamma}$. Since $\leq_{\Gamma}$ is defined as the largest preadequate compatible relation, it is sufficient to prove that $\precsim_{\Gamma}$ is preadequate (we have already showed it is compatible), which directly follows from Sim-1, since $\precsim_{\mathcal{V}} \subseteq \mathcal{U}$. □

8 Bisimilarity, Two-similarity and Contextual Equivalence

In this section we extend previous definitions and results to come up with sound proof techniques for contextual *equivalence*. In particular, by observing that contextual equivalence always coincides with the intersection between the contextual preorder and its converse, Theorem **??** implies that two-way similarity (i.e. the intersection between applicative similarity and its converse) is contained in contextual equivalence. Applicative bisimilarity being finer than two-way similarity, we can also conclude the former to be a sound proof technique for contextual equivalence.

Given a relator Γ, we can extract a canonical notion of Γ-bisimulation from the one of Γ-simulation following the idea that a bisimulation is a relation $\mathcal{R}$ such that both $\mathcal{R}$ and $\mathcal{R}^c$ are simulations. Recall that given a relator Γ we can define a converse operation Γ^c as $\Gamma^c(\mathcal{R}) = (\Gamma(\mathcal{R}^c))^c$. Γ^c is indeed a relator. Similarly, we have proved that the intersection of relators is again a relator.

Definition 31 (Γ-bisimulation)**.** *Given a relator Γ, we say that a relation $\mathcal{R}$ is a Γ-bisimulation if it is a $(\Gamma \cap \Gamma^c)$-simulation.*

Proposition 5. *Let Γ be a relator. A relation $\mathcal{R}$ is a Γ-bisimulation if and only if both $\mathcal{R}$ and $\mathcal{R}^c$ are Γ-simulation.*

Since, by Lemma 7, $\Gamma \cap \Gamma^c$ is a relator, we can define Γ-bisimilarity $\sim_\Gamma$ as $(\Gamma \cap \Gamma^c)$-similarity.

Lemma 16. *Let Γ be a relator. Γ-bisimilarity is an equivalence relation.*

Proof. From Lemma 2 we know that $\sim_\Gamma$ is a preorder, whereas Lemma 7 shows that $\Gamma \cap \Gamma^c$ is conversive. We conclude $\sim_\Gamma$ to be an equivalence relation. □

Definition 32. *Let Γ be a relator. Define Γ-cosimilarity $\gtrsim_\Gamma$ as $(\precsim_\Gamma)^c$. Define Γ two-way similarity $\simeq_\Gamma$ as $\precsim_\Gamma \cap \gtrsim_\Gamma$.*

As usual, bisimilarity is finer than two-way similarity, meaning that $\sim_\Gamma \subseteq \simeq_\Gamma$. Moreover, taking Γ to be the simulation relator for the powerset monad (see Example 5), we have that $\sim_\Gamma$ and $\simeq_\Gamma$ do not coincide. See e.g. [27, 35].

Proposition 6. *Let Γ be a relator. Then, $\sim_\Gamma \subseteq \simeq_\Gamma$, and the inclusion is, in general, strict.*

Recall that we have defined the Γ-contextual preorder $\leq_\Gamma$ as the largest relation that is both compatible and Γ-preadequate. In analogy with what we did for simulation and bisimulation we can give the following:

Definition 33. *Let Γ be a relator. Define Γ-contextual equivalence $\equiv_\Gamma$ as the largest relation that is both compatible and $(\Gamma \cap \Gamma^c)$-preadequate. That is, define $\equiv_\Gamma$ as $\leq_{\Gamma \cap \Gamma^c}$.*

Lemma 17. *Let Γ be a relator. The cocontextual preorder $\geq_\Gamma$ is the largest relation that is both Γ^c-preadequate and compatible.*

Proof. First of all observe that if a relation $\mathcal{R}$ is Γ-preadequate, then $\mathcal{R}^c$ is Γ^c-preadequate. For, suppose $N \; \mathcal{R}^c \; M$, so that $M \; \mathcal{R} \; N$. Since $\mathcal{R}$ is Γ-preadequate, we have $[\![M]\!] \; \Gamma\mathcal{U} \; [\![N]\!]$, and thus $[\![N]\!] \; (\Gamma\mathcal{U})^c \; [\![M]\!]$. From $\mathcal{U}^c = \mathcal{U}$ we can conclude $[\![N]\!] \; \Gamma^c(\mathcal{U}) \; [\![M]\!]$.

As a consequence, since $\leq_\Gamma$ is Γ-preadequate, we have that $\geq_\Gamma$ is Γ^c-preadequate. Moreover, compatibility of $\leq_\Gamma$ implies compatibility of $\geq_\Gamma$. It remains to prove that $\geq_\Gamma$ is the largest Γ^c-preadequate and compatible relation. Let $\mathcal{R}$ be a Γ^c-preadequate and compatible relation. We show $\mathcal{R} \subseteq \geq_\Gamma$ by showing $\mathcal{R}^c \subseteq (\geq_\Gamma)^c$, i.e. $\mathcal{R}^c \subseteq \leq_\Gamma$. We proceed by coinduction showing that $\mathcal{R}^c$ is Γ-preadequate and compatible. Compatibility of $\mathcal{R}^c$ directly follows from that of $\mathcal{R}$. Moreover, since $\mathcal{R}$ is Γ^c-preadequate, $\mathcal{R}^c$ is $(\Gamma^c)^c$-preadequate. A simple calculation shows that $(\Gamma^c)^c = \Gamma$, so that we are done. □

Although bisimilarity is finer than two-way similarity, this is not the case for contextual equivalence and the associated contextual preorders.

Proposition 7. *Let* Γ *be a relator. Then,* $\equiv_\Gamma = \leq_\Gamma \cap \geq_\Gamma$.

Proof. First of all observe that since $\mathcal{U} = \mathcal{U}^c$, a relation $\mathcal{R}$ is $(\Gamma \cap \Gamma^c)$-preadequate if $M\ \mathcal{R}_\Lambda\ N$ implies that both $[\![M]\!]\ \Gamma\mathcal{U}\ [\![N]\!]$ and $[\![N]\!]\ \Gamma\mathcal{U}\ [\![M]\!]$ hold. Since $\equiv_\Gamma$ is defined coinductively, to prove that it contains $\leq_\Gamma \cap \geq_\Gamma$ it is sufficient to prove that $\leq_\Gamma \cap \geq_\Gamma$ is compatible and $(\Gamma \cap \Gamma^c)$-preadequate. Standard calculations show that the set of compatible relations is closed under converse and intersection. Since $\leq_\Gamma$ is compatible, then so is $\geq_\Gamma$ and thus $\leq_\Gamma \cap \geq_\Gamma$. We show that $\leq_\Gamma \cap \geq_\Gamma$ is $(\Gamma \cap \Gamma^c)$-preadequate. Suppose $M\ (\leq_\Gamma \cap \geq_\Gamma)\ N$, so that both $M \leq_\Gamma N$ and $M \geq_\Gamma N$ hold. From the former it follows $[\![M]\!]\ \Gamma\mathcal{U}\ [\![N]\!]$, whereas from the latter we infer $N \leq_\Gamma M$ and thus $[\![N]\!]\ \Gamma\mathcal{U}\ [\![M]\!]$.

We now show that $\equiv_\Gamma$ is contained in $\leq_\Gamma \cap \geq_\Gamma$. Since $\leq_\Gamma$ is defined coinductively, to prove $\equiv_\Gamma \subseteq \leq_\Gamma$ it is sufficient to prove that $\equiv_\Gamma$ is compatible and Γ-preadequate, which is indeed the case. Thanks to Lemma 17 we can proceed coinductively to prove $\equiv_\Gamma \subseteq \geq_\Gamma$ as well. In fact, it is sufficient to prove that $\equiv_\Gamma$ is Γ^c-preadequate, which is trivially the case. □

We can finally prove our soundness result.

Theorem 2 (Soundness). *Let* Γ *be a relator. Two-way similarity* $\simeq_\Gamma$ *is a congruence, and thus sound for contextual equivalence* $\equiv_\Gamma$*. Since bisimilarity* $\sim_\Gamma$ *is finer than* $\simeq_\Gamma$*, it is sound for* $\equiv_\Gamma$ *as well.*

Proof. From Theorem 1 we know that $\precsim_\Gamma$ is a precongruence and that $\precsim_\Gamma \subseteq \leq_\Gamma$. It follows $\succsim_\Gamma$ is a precongruence as well, and that $\succsim_\Gamma \subseteq \geq_\Gamma$ holds. We can conclude $\simeq_\Gamma$ is a congruence and $\simeq_\Gamma \subseteq \equiv_\Gamma$. Since $\sim_\Gamma \subseteq \simeq_\Gamma$, we also have $\sim_\Gamma \subseteq \equiv_\Gamma$. □

Noticeably, Theorem 2 can be seen as a proof of soundness for applicative bisimilarity in any calculus Λ_Σ which respects our requirements (see Definition 16, 17), and in particular for those described in Example 5. The case of probabilistic calculi is illuminating: the apparent complexity of all proofs of congruence from the literature [10, 9] has been confined to the proof that the relator for subdistributions satisfies our axioms.

We can rely on Theorem 2 to prove that the terms W^{raise} and Z^{raise}, our example programs from Section 2, being bisimilar, are indeed contextually equivalent. This only requires checking that the map $\Gamma_\mathcal{D} \circ \Gamma_\mathcal{E}$ (see Example 5) is an inductive relator for the monad $TX = \mathcal{D}(X + E)$ (which trivially carries a continuous Σ-algebra structure) respecting operations in Σ. This is an easy exercise, and does not require any probabilistic reasoning.

Let $(\mathcal{D}, \delta, (\cdot)^\mathcal{D})$ denote the subdistributions monad, where we write $f^\mathcal{D}$ for the Kleisli lifting of f and δ_X for the Dirac distribution on the set X. Similarly, let $(\mathcal{E}, \epsilon, (\cdot)^\mathcal{E})$ denote the exception monad, where we write $f^\mathcal{E}$ for the Kleisli lifting of f, and ϵ for unit of $\mathcal{E}$ (see Example 2 for formal definitions). Moreover, recall that we have relators $\Gamma_\mathcal{D}$ and $\Gamma_\mathcal{E}$ for $\mathcal{D}$ and $\mathcal{E}$, respectively (see Example 5). A standard calculation shows that we have the following:

Proposition 8. *The functor* $\mathcal{D} \circ \mathcal{E}$ *induces a Kleisli triple* $(\mathcal{D} \circ \mathcal{E}, \eta, (\cdot)^\dagger)$*, where the unit* η *is defined, for any set* X*, by* $\eta_X = \delta_{\mathcal{E}(X)} \circ \epsilon_X$*, whereas for a function* $f : X \to \mathcal{D}\mathcal{E}(Y)$ *the Kleisli extension* $f^\dagger$ *of* f *is defined as* $(f_*)^\mathcal{D}$*, where* $f_* : \mathcal{E}(X) \to \mathcal{D}\mathcal{E}(Y)$ *is defined by*

$$f_*(u) = \begin{cases} f(x) & \text{if } u = \mathsf{in}_l(x); \\ \delta_{\mathcal{E}(Y)}(u) & \text{otherwise.} \end{cases}$$

Being defined as composition of relators, the map $\Gamma_\mathcal{D} \circ \Gamma_\mathcal{E}$ (also written $\Gamma_\mathcal{D}\Gamma_\mathcal{E}$) is a relator for the functor $\mathcal{D} \circ \mathcal{E}$. We show that it also satisfies conditions (Lax-Unit) and (Lax-Bind), meaning that it is a relator for $\mathcal{D} \circ \mathcal{E}$, regarded as a monad. In order to have a more readable proof we use a couple of simple auxiliary lemmas. In the following, let $f : X \to \mathcal{D}\mathcal{E}Z$ and $g : Y \to \mathcal{D}\mathcal{E}W$ be maps, and $\mathcal{R} \subseteq X \times Y$, $\mathcal{S} \subseteq Z \times W$ be relations.

Lemma 18. *The following implication holds*

$$\mathcal{R} \subseteq (f \times g)^{-1}(\Gamma_\mathcal{D}\Gamma_\mathcal{E}\mathcal{S}) \implies \Gamma_\mathcal{E}\mathcal{R} \subseteq (f_* \times g_*)^{-1}(\Gamma_\mathcal{D}\Gamma_\mathcal{E}\mathcal{S})$$

Proof. Suppose $\mathcal{R} \subseteq (f \times g)^{-1}(\Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{S})$ and let $u\ \Gamma_{\mathcal{E}}\mathcal{R}\ v$. We prove $f_*(u)\ \Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{S}\ g_*(v)$. By very definition of f_* we have two possible cases.

Case 1. Suppose $f_*(u) = \delta(u)$ and $u = \mathsf{in}_r(e)$, for some $e \in E$ (we will omit subscripts in the Dirac's functions). Since $u\ \Gamma_{\mathcal{E}}\mathcal{R}\ v$ we have $v = \mathsf{in}_r(e)$ as well, meaning that $g_*(v) = \delta(v)$. The thesis can now be rewritten as $\delta(u)\ \Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{R}\ \delta(v)$. Since $\Gamma_{\mathcal{D}}$ satisfies property (Lax-Unit), we have $\Gamma_{\mathcal{E}}\mathcal{S} \subseteq (\delta_{\mathcal{E}(X)} \times \delta_{\mathcal{E}(Y)})^{-1}(\Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{S})$, meaning that the thesis follows from $u\ \Gamma_{\mathcal{E}}\mathcal{S}\ v$. The latter indeed holds (since both $u = \mathsf{in}_r(e) = v$), by very definition of $\Gamma_{\mathcal{E}}$.

Case 2. Suppose now $f_*(u) = f(x)$ and $u = \mathsf{in}_l(x)$, for some $x \in X$. The latter, together with $u\ \Gamma_{\mathcal{E}}\mathcal{R}\ v$ implies $v = \mathsf{in}_l(y)$, for some $y \in Y$ such $x\ \mathcal{R}\ y$. In particular, $v = \mathsf{in}_l(y)$ implies $g_*(v) = g(y)$. By hypothesis, $x\ \mathcal{R}\ y$ implies $f(x)\ \Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{S}\ g(y)$, i.e. $f_*(v)\ \Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{S}\ g_*(v)$.

□

Lemma 19. *The following holds*

$$\mathcal{R} \subseteq (\eta_X \times \eta_Y)^{-1}(\Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{R}).$$

Proof. Suppose $x\ \mathcal{R}\ y$ and recall that $\eta_Z = \delta_{\mathcal{E}(Z)} \circ \epsilon_Z$. Since $\Gamma_{\mathcal{E}}$ satisfies property (Lax-Unit) (wrt $\mathcal{E}$), from $x\ \mathcal{R}\ y$, we infer $\epsilon_X(x)\ \Gamma_{\mathcal{E}}\mathcal{R}\ \epsilon_Y(y)$. We conclude that $\delta_{\mathcal{E}(X)}(\epsilon_X(x))\ \Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{R}\ \delta_{\mathcal{E}(Y)}(\epsilon_Y(y))$ holds, since $\Gamma_{\mathcal{D}}$ satisfies (Lax-Bind) as well (wrt $\mathcal{D}$). □

Corollary 3. *The map $\Gamma_{\mathcal{D}} \circ \Gamma_{\mathcal{E}}$ is a relator for the monad $\mathcal{D} \circ \mathcal{E}$.*

Proof. By previous lemma it is sufficient to prove that given $\mathcal{R} \subseteq (f \times g)^{-1}(\Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{S})$ we have $\Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{R} \subseteq (f^\dagger \times g^\dagger)^{-1}(\Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{S})$, i.e. $\Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{R} \subseteq (f_*^{\mathcal{D}} \times g_*^{\mathcal{D}})^{-1}(\Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{S})$. Since $\Gamma_{\mathcal{D}}$ is a relator for the monad $\mathcal{D}$, the latter is implied by $\Gamma_{\mathcal{E}}\mathcal{R} \subseteq (f_* \times g_*)^{-1}(\Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{S})$, which itself follows from $\mathcal{R} \subseteq (f \times g)^{-1}(\Gamma_{\mathcal{D}}\Gamma_{\mathcal{E}}\mathcal{S})$ and Lemma 18. □

To conclude, we have to show that the monad $\mathcal{D} \circ \mathcal{E}$ and the relator $\Gamma_{\mathcal{D}} \circ \Gamma_{\mathcal{E}}$ have the required order-theoretic properties. First of all note that the monad $\mathcal{D} \circ \mathcal{E}$ carries a continuous Σ-algebra structure, with the ω**CPPO** order given by the functor $\mathcal{D}$. Moreover, trivial calculations show that the monad is ω**CPPO**-enriched (this essentially follows from the order-enrichment of $\mathcal{D}$, together with the validity of the equation $(\bigsqcup_{n<\omega} f_n)_* = \bigsqcup_{n<\omega} f_{n*}$), and thus the associated bind operator is continuous. Finally, it is immediate to observe that $\Gamma_{\mathcal{D}} \circ \Gamma_{\mathcal{E}}$ is inductive, since $\Gamma_{\mathcal{D}}$ is.

9 Related Work

As mentioned in the Introduction, this is certainly not the first paper about program equivalence for higher-order effectful calculi. Denotational semantics of calculi having this nature, has been studied since Moggi's seminal work [32], thus implicitly providing a notion of equivalence. All this has been given a more operational flavour starting with Plotkin and Power account on adequacy for algebraic effects [36], from which the operational semantics presented in this paper is greatly inspired. The literature also offers abstract accounts on logical relations for effectful calculi. The first of them is due to Goubault-Larrecq, Lasota and Nowak [18], which is noticeably able to deal with nondeterministic and probabilistic effects, but also with dynamic name creation, for which applicative bisimilarity is known to be unsound. Another piece of work which is related to ours is due to Johann, Simpson, and Voigtländer [20], who focused on algebraic effects and observational equivalence, and their characterisation via CIU theorems and a form of logical relation based $\top\top$-lifting. In both cases, the target language is typed. Similar in spirit to our approach (which is based on the notion of relator), the work of Katsumata and Sato [22] analyses monadic lifting of relations in the context of $\top\top$-lifting.

Although no abstract account exists on applicative coinductive techniques for calculi with algebraic effects, some work definitely exists in some specific cases. As a noticeable example, the works by Ong [34] and Lassen [27] deal with nondeterminism, and establish soundness in all relevant cases, although full abstraction fails. The first author, together with Alberti, Crubillé and Sangiorgi [10, 9] have studied the probabilistic case, where full abstraction can indeed be obtained if call-by-value evaluation is employed.

10 Conclusion

This is the first abstract account on applicative bisimilarity for calculi with effects. The main result is an abstract soundness theorem for a notion of applicative similarity which can be naturally defined as soon as a monad and an associated relator are given which on the one hand serve to give an operational semantics to the algebraic operations, and on the other need to satisfy some mild conditions in order for similarity to be a precongruence. Soundness of bisimilarity is then obtained as a corollary. Many concrete examples are shown to fit into the introduced axiomatics. A notable example is the output monad, for which a definition of applicative similarity based on labeled transition systems as in e.g. [8] is unsound, a fact that the authors discovered after noticing the anomaly, and not vice versa. Nevertheless, we defined a different notion of applicative similarity that fits into our framework and whose associated notion of bisimilarity (Definition 31) coincide with the usual notion of bisimilarity.

A question that we have not addressed in this work, but which is quite natural, is whether an abstract full-abstraction result could exist, analogously to what, e.g., Johann, Simpson, and Voigtländer obtained for their notion of logical relation. This is a very interesting topic for future work. It is however impossible to get such a theorem without imposing some further, severe, constraints on the class of effects (i.e. monads and relators) of interest, e.g., applicative bisimilarity is well-known not to be fully-abstract in calculi with nondeterministic effects, which perfectly fit in the picture we have drawn in this paper. A promising route towards this challenge would be to understand which class of *tests* (if any) characterise applicative bisimilarity, depending on the underlying monad and relator, this way generalising results by van Breugel, Mislove, Ouaknine and Worrell [42] or Ong [9].

Finally, environmental bisimilarity is known [24] to overcome the limits of applicative bisimilarity in presence of information hiding. Studying the applicability of the methodology developed in this work to environmental bisimilarity is yet another interesting topic for future researches.

Acknowledgment

The authors would like to thank Raphaëlle Crubillé and the anonymous reviewers for the many useful comments, some of which led to a substantial improvement of our work. Special thanks go to Davide Sangiorgi, Ryo Tanaka, and Valeria Vignudelli for many insightful discussions about the topics of this work.

References

[1] Samson Abramsky. The lazy lambda calculus. In D. Turner, editor, *Research Topics in Functional Programming*, pages 65–117. Addison Wesley, 1990.

[2] Samson Abramsky and Achim Jung. Domain theory. In *Handbook of Logic in Computer Science*, pages 1–168. Clarendon Press, 1994.

[3] Andrew W. Appel and David A. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Trans. Program. Lang. Syst.*, 23(5):657–683, 2001.

[4] Hendrik P. Barendregt. *The lambda calculus: its syntax and semantics*. Studies in logic and the foundations of mathematics. North-Holland, 1984.

[5] M. Barr. Relational algebras. *Lect. Notes Math.*, 137:39–55, 1970.

[6] Nick Benton, Andrew Kennedy, Lennart Beringer, and Martin Hofmann. Relational semantics for effect-based program transformations: higher-order store. In *Proc. of PPDP 2009*, pages 301–312, 2009.

[7] Ales Bizjak and Lars Birkedal. Step-indexed logical relations for probability. In *Proc. of FOSSACS 2015*, pages 279–294, 2015.

[8] Roy L. Crole and Andrew D. Gordon. Relating operational and denotational semantics for input/output effects. *Mathematical Structures in Computer Science*, 9(2):125–158, 1999.

[9] Raphaëlle Crubillé and Ugo Dal Lago. On probabilistic applicative bisimulation and call-by-value λ-calculi. In *Proc. of ESOP 2014*, volume 8410 of *LNCS*, pages 209–228. Springer, 2014.

[10] Ugo Dal Lago, Davide Sangiorgi, and Michele Alberti. On coinductive equivalences for higher-order probabilistic functional programs. In *Proc. of POPL 2014*, pages 297–308, 2014.

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

[12] Vincent Danos and Russell Harmer. Probabilistic game semantics. *ACM Transactions on Computational Logic*, 3(3):359–382, 2002.

[13] Brian A. Davey and Hilary A. Priestley. *Introduction to lattices and order.* Cambridge University Press, 1990.

[14] Ugo de'Liguoro and Adolfo Piperno. Non deterministic extensions of untyped lambda-calculus. *Inf. Comput.*, 122(2):149–177, 1995.

[15] Joseph A. Goguen, James W. Thatcher, Eric G. Wagner, and Jesse B. Wright. Initial algebra semantics and continuous algebras. *J. ACM*, 24(1):68–95, 1977.

[16] Sergey Goncharov and Lutz Schröder. A relatively complete generic hoare logic for order-enriched effects. In *Proc. of LICS 2013*, pages 273–282. IEEE Computer Society, 2013.

[17] Andrew D. Gordon. A tutorial on co-induction and functional programming. In *Workshops in Computing*, pages 78–95. Springer London, September 1994.

[18] Jean Goubault-Larrecq, Slawomir Lasota, and David Nowak. Logical relations for monadic types. *Mathematical Structures in Computer Science*, 18(6):1169–1217, 2008.

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

[20] Patricia Johann, Alex Simpson, and Janis Voigtländer. A generic operational metatheory for algebraic effects. In *Proc. of LICS 2010*, pages 209–218. IEEE Computer Society, 2010.

[21] Claire Jones. *Probabilistic Non-determinism.* PhD thesis, University of Edinburgh, 1990. Available as Technical Report CST-63–90.

[22] Shin-ya Katsumata and Tetsuya Sato. *Preorders on Monads and Coalgebraic Simulations*, pages 145–160. Springer, 2013.

[23] Gregory M. Kelly. Basic concepts of enriched category theory. *Reprints in Theory and Applications of Categories*, (10):1–136, 2005.

[24] Vasileios Koutavas, Paul Blain Levy, and Eijiro Sumii. From applicative to environmental bisimulation. *Electr. Notes Theor. Comput. Sci.*, 276:215–235, 2011.

[25] Alexander Kurz and Jiri Velebil. Relation lifting, a survey. *J. Log. Algebr. Meth. Program.*, 85(4):475–499, 2016.

[26] Søren B. Lassen. Relational reasoning about contexts. In Andrew D. Gordon and Andrew M. Pitts, editors, *Higher Order Operational Techniques in Semantics*, pages 91–136. 1998.

[27] Søren B. Lassen. *Relational Reasoning about Functions and Nondeterminism.* PhD thesis, Dept. of Computer Science, University of Aarhus, May 1998.

[28] Paul Blain Levy. Infinitary Howe's method. *Electr. Notes Theor. Comput. Sci.*, 164(1):85–104, 2006.

[29] Paul Blain Levy. Similarity quotients as final coalgebras. In *Proc. of FOSSACS 2011*, volume 6604 of *LNCS*, pages 27–41, 2011.

[30] Paul Blain Levy, John Power, and Hayo Thielecke. Modelling environments in call-by-value programming languages. *Inf. Comput.*, 185(2):182–210, 2003.

[31] Saunders MacLane. *Categories for the Working Mathematician.* Springer-Verlag, 1971.

[32] Eugenio Moggi. Computational lambda-calculus and monads. In *Proc. of (LICS 1989*, pages 14–23. IEEE Computer Society, 1989.

[33] J. Morris. *Lambda Calculus Models of Programming Languages.* PhD thesis, MIT, 1969.

[34] C.-H. Luke Ong. Non-determinism in a functional setting. In *Proc. of LICS 1993*, pages 275–286. IEEE Computer Society, 1993.

[35] Andrew M. Pitts. Howe's method for higher-order languages. In D. Sangiorgi and J. Rutten, editors, *Advanced Topics in Bisimulation and Coinduction*, volume 52 of *Cambridge Tracts in Theoretical Computer Science*, chapter 5, pages 197–232. Cambridge University Press, November 2011.

[36] Gordon D. Plotkin and John Power. Adequacy for algebraic effects. In *Proc. of FOSSACS 2001*, pages 1–24, 2001.

[37] Gordon D. Plotkin and John Power. Notions of computation determine monads. In *Proc. of FOSSACS 2002*, pages 342–356, 2002.

[38] Gordon D. Plotkin and John Power. Algebraic operations and generic effects. *Applied Categorical Structures*, 11(1):69–94, 2003.

[39] Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000.

[40] Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, Inc., New York, NY, USA, 1986.

[41] Albert Marchienus Thijs et al. *Simulation and fixpoint semantics.* Rijksuniversiteit Groningen, 1996.

[42] Franck van Breugel, Michael W. Mislove, Joël Ouaknine, and James Worrell. Domain theory, testing and simulation for labelled markov processes. *Theor. Comput. Sci.*, 333(1-2):171–197, 2005.