A Fibrational Tale of Operational Logical Relations

Francesco Dagnino, Francesco Gavazzo

A Fibrational Tale of Operational Logical Relations

Francesco Dagnino
University of Genova, Italy

Francesco Gavazzo
University of Bologna, Italy

Abstract

Logical relations built on top of an operational semantics are one of the most successful proof methods in programming language semantics. In recent years, more and more expressive notions of operationally-based logical relations have been designed and applied to specific families of languages. However, a unifying abstract framework for operationally-based logical relations is still missing. We show how fibrations can provide a uniform treatment of operational logical relations, using as reference example a λ-calculus with generic effects endowed with a novel, abstract operational semantics defined on a large class of categories. Moreover, this abstract perspective allows us to give a solid mathematical ground also to differential logical relations – a recently introduced notion of higher-order distance between programs – both pure and effectful, bringing them back to a common picture with traditional ones.

2012 ACM Subject Classification Theory of computation → Operational semantics

Keywords and phrases logical relations, operational semantics, fibrations, generic effects, program distance

Digital Object Identifier 10.4230/LIPIcs.FSCD.2022.3

Funding This work was partially funded by the MUR project T-LADIES (PRIN 2020TL3X8X). The second author is supported by the ERC Consolidator Grant DLV-818616 DIAPASoN.

Acknowledgements The authors would like to thank the anonymous reviewers for the many useful comments, some of which improved of our work.

1 Introduction

Logical relations [83] are one of the most successful proof techniques in logic and programming language semantics. Introduced in proof theory [93, 43] in their unary form, logical relations have soon became a main tool in programming language semantics. In fact, starting with the seminal work by Reynolds [83], Plotkin [79], and Statman [87], logical relations have been extensively used to study both the denotational and operational behaviour of programs.[1]

Logical relations (and predicates) mostly come in two flavours, depending on whether they are defined relying on the *operational* or *denotational* semantics of a language. We refer to logical relations of the first kind as *operational logical relations* and to logical relations of the second kind as *denotational logical relations*. Due to their link with denotational semantics, denotational logical relations have been extensively studied in the last decades, both for specific programming languages and in the abstract, this way leading to beautiful general theories of (logical) predicates and relations on program denotations. In particular, starting with the work by Reynolds and Ma [68], Mitchell and Scedrov [75], and Hermida [49], researchers have started to investigate notions of (logical) predicates and relations in a

[1] See the classic textbooks by Mitchell [74], Pierce [77], and Harper [48] (and references therein) for an introduction to both denotationally- and operationally-based logical relations.

general categorical setting, this way giving rise to an abstract understanding of relations over (the denotational semantics of) programs centred around the notions of *fibrations* [49, 39, 86, 62, 60, 61, 59] *reflexive graphs* [76, 84, 34, 51], and *factorisation systems* [57, 54, 46]. The byproduct of all of that is a general, highly modular theory of denotational logical relations that has been successfully applied to a large array of language features, ranging from parametricty and polymorphism [39, 86, 34] to computational effects [62, 60, 61, 59, 65, 57, 46].

On the operational side, researchers have focused more on the development and applications of expressive notions of logical relations for specific (families of) languages, rather than on their underlying theory. In fact, operational logical relations being based on operational semantics, they can be easily defined for languages for which finding the right denotational model is difficult, this way making operational logical relations a handy and lightweight technique, especially when compared to its denotational counterpart. As a paradigmatic example, consider the case of stochastic λ-calculi and their operational techniques [16, 21, 97] which can be (easily) defined relying on the category of measurable spaces and measurable functions, but whose denotational semantics have required the introduction of highly non-trivial mathematical structures [89, 88, 95, 35], since the category measurable spaces (and measurable functions) is not closed [6].

The wide applicability of operational logical relations, however, has also prevented the latter to organise as a uniform *corpus* of techniques with a common underlying theory. Operational logical relations result in a mosaic of powerful techniques applied to a variety of languages – including higher-order, functional, imperative, and concurrent languages [32, 3, 94, 4, 17, 33]; both pure and (co)effectful [56, 25, 1, 12, 52, 10, 11, 13] – whose relationship, however, is unclear. This situation creates a peculiar scenario where, on the one hand, the effectiveness of operational logical relations has been proved by their many applications but, on the other hand, a foundational understanding of operational logical relations is still missing, the main consequence of that being the lack of modularity in their development. All of that becomes even worst if one also takes into account more recent forms of logical relations, such as metric [82, 78] and differential [30, 22, 23, 25] ones, that go beyond traditional relational reasoning.

In this paper, we show that much in the same way denotational logical relations can be uniformly understood in terms of fibrations, it is possible to give a uniform account of *operational logical relations* relying on the language of *fibrations*. In this respect, our contribution is twofold.

Operational Logical Relations, Fibrationally. Our first contribution is the development of a general, abstract notion of an operational logical relation in terms of fibrations for a λ-calculus with generic effects. Fibrations are a mainstream formalism for general, categorical notions of predicates/relations. More precisely, a fibration is a suitable functor from a category of (abstract) predicates – the domain of the fibration – to a category of arguments – usually called the base category. In denotational logical relations, predicates usually apply to program denotations rather than on program themselves, with the main consequence that the base category is usually required to be cartesian closed. In this paper, we follow a different path and work with base categories describing the operational (and interactive) behaviour of programs, rather than their denotations. To do so, we introduce the novel notion of an *operational structure*, the latter being a cartesian category with arrows describing (monadic) evaluation semantics [80, 27] and satisfying suitable coherence conditions encoding the base dynamics of program evaluation. This way, we give not only an abstract account to

traditional set-based evaluation semantics, but also of evaluation semantics going beyond the category of sets and functions, the prime example being the evaluation semantics of stochastic λ-calculi, which is defined as a stochastic kernel [21, 97].

On top of our abstract operational semantics, we give a general notion of an operational logical relation in terms of (logical) fibrations and prove a general result (which we call the fundamental lemma of logical relations, following standard nomenclature of concrete, operational logical relations) stating that programs behave as arrows in the domain of the fibration. Remarkably, our general fundamental lemma subsumes several concrete instances of the fundamental lemma of logical relations appearing in the literature. Additionally, the operational nature of our framework immediately results in a wide applicability of our results, especially if compared with fibrational accounts of denotational logical relations. In particular, since our logical relations builds upon operational structures, they can be instantiated to non-cartesian-closed categories, this way reflecting at a general level the wider applicability of operational techniques with respect to denotational ones. As a prime example of that, we obtain operational logical relations (and their fundamental lemma) for stochastic λ-calculi for free, something that is simply not achievable denotationally, due to the failure of cartesian closedness of the category of measurable spaces.

Fibrational Differential Reasoning. Our second, main contribution is to show that our framework goes beyond traditional relational reasoning, as it gives a novel mathematical account of the recently introduced differential logical relations [31, 22, 23, 25] (DLRs, for short), both pure and effectful. DLRs are a new form logical relations introduced to define higher-order distances between programs, such distances being abstract notions reflecting the interactive complexity of the programs compared. DLRs have been studied *operationally* and on specific calculi only, oftentimes introducing new notions – such as the one of a *differential extension* of a monad [25] – whose mathematical status is still not well understood. The main consequence of that is that a general, structural account of pure and effectful DLRs is still missing. In this paper, we show how DLRs are a specific instance of our abstract operational logical relations and how the fundamental lemma of DLRs is an instance of our general fundamental lemma. We do so by introducing the novel construction of a *fibration of differential relations* and showing how the latter precisely captures the essence of DLRs, bringing them back to a common framework with traditional logical relations. Additionally, we show how our fibrational account sheds a new light on the mathematical status of effectful DLRs. In particular, we show that differential extensions of monads are precisely liftings of monads to the fibration of differential relations, and that the so-called coupling-based differential extension [25] – whose canonicity has been left as an open problem – is an instance of a general monadic lifting to the fibration of differential relations: remarkably, such a lifting is the extension of the well-known Barr lifting [8] to a differential setting.

Related Work. Starting with the seminal work by Hermida [49], fibrations have been used to give categorical notions of (logical) predicates and relations, and to model denotational logical relations [39, 86, 62, 60, 61, 59], as they provide a formal way to relate the denotational semantics of a programming language and a logic for reasoning about it. Other categorical approaches to denotational logical relations have been given in terms of reflexive graphs [76, 84, 34, 51] and factorisation systems [57, 54, 46].

On the operational side, fibrations have been used to give abstract accounts to induction and coinduction [50], both in the setting of initial algebra-final coalgebra semantics [42, 41, 40] and in the setting of up-to techniques [15, 14]. To the best of the authors' knowledge,

however, none of these approaches has been applied to operational reasoning for higher-order programming languages. Concerning the latter, general operational accounts of logical relations both for effectful [56] and combined effectful and coeffectful languages [1, 26] have been given in terms of relational reasoning. These approaches, however, are tailored on specific operational semantics and notions of relations, and thus they cannot be considered truly general. Finally, DLRs have been studied mostly operationally [31, 22, 23, 25], although some general *denotational* accounts of DLRs have been proposed [31, 78]. Even if not dealing with operational aspects of DLRs, the latter proposals can cope with *pure* DLRs only, and are too restrictive to incorporate computational effects.

2 The Anatomy of an Operational Semantics

To define a general notion of an operational logical relation, we first need to define a general notion of an operational semantics. This is precisely the purpose of this section. In particular, we introduce the notion of an *operational structure* on a category with finite products endowed with a strong monad as an axiomatisation of a general evaluation semantics. Operational structures prescribe the existence of basic interaction arrows (the latter describing basic program interactions as given by the usual reduction rules) and define program execution as a Kleisli arrow (this way giving monadic evaluation) satisfying suitable coherence laws reflecting evaluation dynamics. Remarkably, operational structures turn out to be more liberal – hence widely applicable – than categories used in denotational semantics (the latter being required to be cartesian closed).

2.1 A Calculus with Generic Effects

Our target calculus is a simply-typed fine grain call-by-value [66] $\mathbf{\Lambda}$ enriched with generic effects [45, 81]. Recall that for a strong monad[2] $\mathbb{T} = (T, \eta, \gg\!=)$ on a cartesian category $\mathcal{C}$, a generic effect [81, 80] of arity A, with A an object of $\mathcal{C}$, is an arrow $\boldsymbol{\gamma} : 1 \to TA$. Standard examples of generic effects are obtained by taking $\mathcal{C} = \mathcal{S}et$ and A equal to a finite set giving the arity of the effect, so that, for instance, one can model nondeterministic (resp. fair) coins as elements of $\mathcal{P}(2)$ (resp. $\mathcal{D}(2)$), where $\mathcal{P}$ (resp. $\mathcal{D}$) is the powerset (resp. distribution) monad. Other examples of generic effects include primitives for input-output, memory updates, exceptions, etc. Here, we assume to have a collection of generic effect symbols γ with an associated type σ_γ, leaving the interpretation of γ as an actual generic effect $\boldsymbol{\gamma}$ to the operational semantics. The syntax and static semantics of $\mathbf{\Lambda}$ are given in Figure 1, where ζ ranges over base types and c over constants of type ζ (for ease of exposition, we do include operations on base types, although those can be easily added).

Notice that $\mathbf{\Lambda}$'s expressions are divided into two (disjoint) classes: *values* (notation $v, w, \ldots$) and *computations* (notation $t, s, \ldots$), the former being the result of a computation, and the latter being an expression that once evaluated may produce a value (the evaluation process might not terminate) as well as side effects. When the distinction between values and computations is not relevant, we generically refer to *terms* (and still denote them as $t, s, \ldots$). We adopt standard syntactic conventions [7] and identify terms up to renaming of bound variables: we say that a term is closed if it has no free variables and write $\mathcal{V}_\sigma$, Λ_σ for the sets of closed values and computations of type σ, respectively. We write $t[v_1, \ldots, v_n/x_1, \ldots, x_n]$

[2] We use the notions of a strong monad $(T, \eta, \mu, \mathsf{st})$ and of a strong Kleisli triple $\mathbb{T} = (T, \eta, \gg\!=)$ interchangeably, where for an arrow $f : X \times Y \to TZ$ in a cartesian category $\mathcal{C}$, we denote by $\gg\!= f : X \times TY \to TZ$ the strong Kleisli extension of f. We write $f^\dagger : TX \to TY$ for the Kleisli extension of $f : X \to TY$.

$$v, w ::= x \mid c \mid \langle\rangle \mid \lambda x.t \mid \langle v, w\rangle$$
$$t, s ::= \mathbf{val}\ v \mid vw \mid v.1 \mid v.2 \mid t\ \mathbf{to}\ x.s \mid \gamma$$

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma} \quad \frac{}{\Gamma \vdash c : \zeta} \quad \frac{\Gamma \vdash v : \sigma}{\Gamma \vdash \mathbf{val}\ v : \sigma} \quad \frac{}{\Gamma \vdash \gamma : \sigma_\gamma} \quad \frac{\Gamma, x : \sigma \vdash t : \tau}{\Gamma \vdash \lambda x.t : \sigma \to \tau} \quad \frac{\Gamma \vdash v : \sigma \to \tau \quad \Gamma \vdash w : \sigma}{\Gamma \vdash vw : \tau}$$

$$\frac{\Gamma \vdash t : \sigma \quad \Gamma, x : \sigma \vdash s : \tau}{\Gamma \vdash t\ \mathbf{to}\ x.s : \tau} \quad \frac{\Gamma \vdash v : \sigma \times \tau}{\Gamma \vdash v.1 : \sigma} \quad \frac{\Gamma \vdash v : \sigma \times \tau}{\Gamma \vdash v.2 : \tau} \quad \frac{\Gamma \vdash v : \sigma \quad \Gamma \vdash w : \tau}{\Gamma \vdash \langle v, w\rangle : \sigma \times \tau} \quad \frac{}{\Gamma \vdash \langle\rangle : \mathbf{1}}$$

Figure 1 Syntax and Static semantics of $\mathbf{\Lambda}$.

(and similarly for values) for the capture-avoiding (simultaneous) substitution of the values $v_1, \dots, v_n$ for all free occurrences of $x_1, \dots, x_n$ in t. Oftentimes, we will use the notation $\overrightarrow{\phi}$ for a sequence $\phi_1, \dots, \phi_n$ of symbols ϕ_i.

When dealing with denotational logical relations, one often organises $\mathbf{\Lambda}$ as a syntactic category having types as objects and (open) terms modulo the usual $\beta\eta$-equations as arrows. Operationally, however, terms are purely syntactic objects and cannot be taken modulo $\beta\eta$-equality. For that reason, we consider a *syntactic graph* rather than a syntactic category.[3]

▶ **Definition 2.1.** *The objects of the syntactic graph $\mathcal{Syn}$ are environments Γ, types σ, and expressions $\underline{\sigma}$, for σ a type. Arrows are defined thus:* $\hom(\Gamma, \sigma)$ *consists of values* $\Gamma \vdash v : \sigma$*, whereas* $\hom(\Gamma, \underline{\sigma})$ *consists of terms* $\Gamma \vdash t : \sigma$*; otherwise, the hom-set is empty.*

The definition of $\mathcal{Syn}$ reflects the call-by-value nature of $\mathbf{\Lambda}$: to each type σ we associate two objects, representing the type σ on values and on computations. Moreover, there is no arrow having environments as codomains nor having objects $\underline{\sigma}$ as domain: this reflects that in call-by-value calculi variables are placeholders for values, not for computations.

2.2 Operational Semantics: The Theoretical Minimum

Having defined the syntax of $\mathbf{\Lambda}$, we move to its operational semantics. Among the many style of operational semantics (small-step, big-step, etc), evaluation semantics turns out to be a convenient choice for our goals. Evaluation semantics are usually defined as monadic functions $\boldsymbol{e} : \Lambda_\sigma \to T(\mathcal{V}_\sigma)$, with T a monad encoding the possible effects produced during program evaluation (e.g. divergence or nondeterminism) [27, 80, 56]. To clarify the concept, let us consider an example [27].

▶ **Example 2.2.** Let $\mathbb{T}$ be a monad on $\mathcal{Set}$ with a generic effects $\boldsymbol{\gamma} \in T(\mathcal{V}_{\sigma_\gamma})$ for effect symbols γ in $\mathbf{\Lambda}$. The (monadic) evaluation (family of) map(s) $[\![-]\!] : \Lambda_\sigma \to T(\mathcal{V}_\sigma)$ is defined as follows (notice that $\mathbf{\Lambda}$ being simply-typed, $[\![-]\!]$ is well-defined[4]).

$$[\![\mathbf{val}\ v]\!] = \eta(v) \quad [\![(\lambda x.t)v]\!] = [\![t[v/x]]\!] \quad [\![t\ \mathbf{to}\ x.s]\!] = [\![s[\cdot/x]]\!]^\dagger [\![t]\!] \quad [\![\langle v_1, v_2\rangle.i]\!] = \eta(v_i) \quad [\![\gamma]\!] = \boldsymbol{\gamma}$$

[3] Recall that a graph is defined by removing from the definition of a category the axioms prescribing the existence of identity and composition. A diagram is a map from graphs to graphs (a diagram being defined by removing from the definition of a functor the clauses on identity and composition). Since any category is a graph, we use the word *diagram* also to denote maps from graphs to categories.

[4] It is worth remarking that termination is not an issue for our general notion of an evaluation semantics (Definition 2.3 below). In fact, the results presented in this paper simply require to have an evaluation

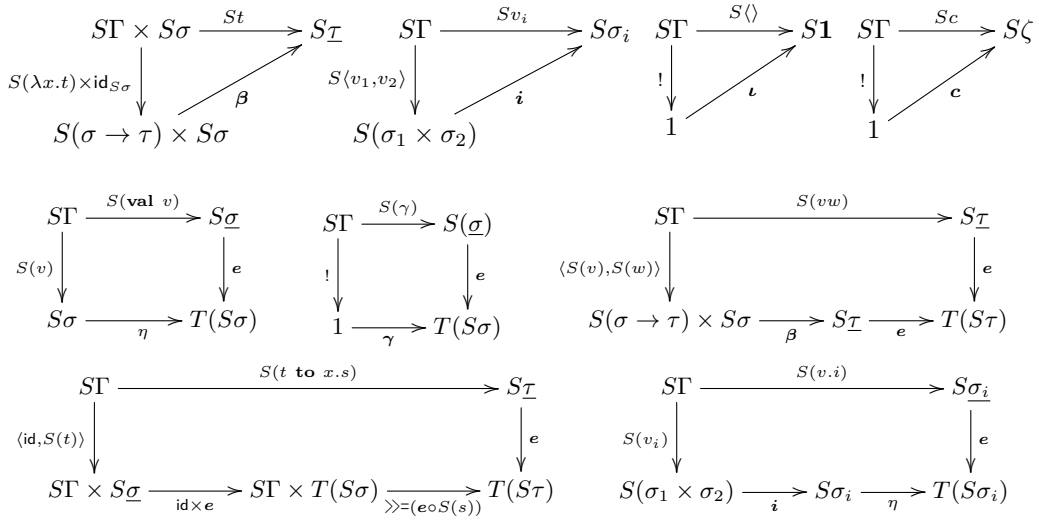


Figure 2 Coherence laws, where $-.i \in \{-.1, -.2\}$ and $\boldsymbol{i} \in \{\boldsymbol{p_1}, \boldsymbol{p_2}\}$.

Example 2.2 defines evaluation semantics as an arrow in the category $\mathcal{Set}$ of sets and functions relying on two main ingredients: the monad $\mathbb{T}$ and its algebraic operations; and primitive functions implementing the basic mechanism of β-reductions, viz. application/substitution and projections. The very same recipe has been used to define specific evaluation semantics beyond $\mathcal{Set}$, prime example being kernel-like evaluation semantics for stochastic λ-calculi [21, 97, 95, 35], where evaluation semantics are defined as Kleisli arrows on suitable categories of measurable spaces (see Example 2.5 below for details). Here, we propose a general notion of an operational semantics for $\mathbf{\Lambda}$ in an arbitrary cartesian category $\mathcal{B}$ and with respect to a monad $\mathbb{T}$. We call the resulting notion a *($\mathcal{Syn}$-)operational structure*.

▶ **Definition 2.3.** *Given a category $\mathcal{B}$ with finite products and a strong monad $\mathbb{T}$ on it, a* ($\mathcal{Syn}$-)operational structure *consists of a diagram $S : \mathcal{Syn} \to \mathcal{B}$ satisfying $S(\overrightarrow{x : \sigma}) = \prod \overrightarrow{S\sigma}$, together with the following (interaction) arrows (notice that $\boldsymbol{\gamma}$ is a generic effect) and satisfying the coherence laws in Figure 2.*

$$\boldsymbol{e} : S\underline{\sigma} \to T(S\sigma) \qquad \boldsymbol{\iota} : 1 \to S\mathbf{1} \qquad \boldsymbol{\beta} : S(\sigma \to \tau) \times S\sigma \to S\underline{\tau} \qquad \boldsymbol{c} : 1 \to S\zeta$$
$$\boldsymbol{p_1} : S(\sigma \times \tau) \to S\sigma \qquad \boldsymbol{p_2} : S(\sigma \times \tau) \to S\tau \qquad \boldsymbol{\gamma} : 1 \to T(S\sigma_\gamma)$$

Notice how the first four coherence laws in Figure 2 ensure the intended behaviour of the arrows $\boldsymbol{\beta}, \boldsymbol{\iota}, \boldsymbol{p_i}, \boldsymbol{c}$, whereas the remaining laws abstractly describe the main dynamics of program execution. Notice also that Definition 2.3 prescribes the existence of an evaluation arrow $\boldsymbol{e}$: it would be interesting to find conditions on $\mathcal{B}$ (probably a domain-like enrichment [64] or partial additivity [71]) ensuring the existence of $\boldsymbol{e}$. We can now instantiate Definition 2.3 to recover standard $\mathcal{Set}$-based evaluation semantics as well as operational semantics on richer categories. In particular, since $\mathcal{B}$ need not be closed, we can give $\mathbf{\Lambda}$ an operational semantics in the category $\mathcal{Meas}$ of measurable spaces and measurable functions.

map satisfying suitable coherence conditions. Consequently, we could rephrase this example ignoring termination by requiring the monad to be enriched in an ω-complete partial order [2] and defining evaluation semantics as a least fixed point of a suitable map, as it is customary in monadic evaluation semantics [27].

▶ **Example 2.4.** Let $\mathcal{B} = \mathcal{S}et$ and define $S : \mathcal{S}yn \to \mathcal{S}et$ thus:

$$S\sigma = \mathcal{V}_\sigma \qquad S\underline{\sigma} = \Lambda_\sigma \qquad S(\overrightarrow{x : \sigma}) = \prod \overrightarrow{S\sigma} \qquad S(\Gamma \vdash t : \sigma)(\overrightarrow{v}) = t[\overrightarrow{v}/\overrightarrow{x}]$$

We obtain an operational structure by defining the maps $\boldsymbol{\beta}, \boldsymbol{p_1}, \boldsymbol{p_2}$, and $\boldsymbol{\iota}$ in the obvious way (e.g. $\boldsymbol{\beta}(\lambda x.t, v) = t[v/x]$ and $\boldsymbol{p_i}\langle v_1, v_2\rangle = v_i$). Finally, given a monad $\mathbb{T}$ with generic effects $\boldsymbol{\gamma} \in T\mathcal{V}_{\sigma_\gamma}$ for each effect symbol γ, we define $\boldsymbol{e}$ as the evaluation map of Example 2.2.

▶ **Example 2.5** (Stochastic λ-calculus)**.** Let us consider the instance of $\boldsymbol{\Lambda}$ with a base type $\mathbf{R}$ for real numbers, constants c_r for each real number r, and the generic effect symbol $\mathcal{U}$ standing for the uniform distribution over the unit interval. Recall that $\mathcal{M}eas$ has countable products and coproducts (but not exponentials [6]). To define the diagram $S : \mathcal{S}yn \to \mathcal{M}eas$, we rely on the well-known fact [36, 89, 16] that both $\mathcal{V}_\sigma$ and Λ_σ can be endowed with a σ-algebra making them measurable (actually Polish) spaces in such a way that $\mathcal{V}_{\mathbf{R}} \cong \mathbb{R}$ and that the substitution map is measurable. We write Σ_σ and $\Sigma_{\underline{\sigma}}$ for the σ-algebras associated to $\mathcal{V}_\sigma$ and Λ_σ, respectively. We thus define $S : \mathcal{S}yn \to \mathcal{M}eas$ as follows:

$$S\sigma = (\mathcal{V}_\sigma, \Sigma_\sigma) \qquad S\underline{\sigma} = (\Lambda_\sigma, \Sigma_{\underline{\sigma}}) \qquad S(\overrightarrow{x : \sigma}) = \prod \overrightarrow{S\sigma} \qquad S(\Gamma \vdash t : \sigma)(\overrightarrow{v}) = t[\overrightarrow{v}/\overrightarrow{x}]$$

We obtain an operational structure by observing that the maps $\boldsymbol{\beta}, \boldsymbol{p_1}, \boldsymbol{p_2}$, and $\boldsymbol{\iota}$ of previous example extend to $\mathcal{M}eas$, in the sense that they are all measurable functions. Next, we consider the Giry monad [44] $\mathcal{G} : \mathcal{M}eas \to \mathcal{M}eas$ which associates to each measurable space the space of probability measures on it. By Fubini-Tonelli theorem, $\mathcal{G}$ is strong. Let $\boldsymbol{\mathcal{U}}$ be the Lebesgue measure on $[0, 1]$, which we regard as an arrow $\boldsymbol{\mathcal{U}} : 1 \to \mathcal{G}(S\mathcal{V}_{\mathbf{Real}})$, and thus as a generic effect in $\mathcal{M}eas$. We then define [21] $\boldsymbol{e} : \Lambda_\sigma \to \mathcal{G}(\mathcal{V}_\sigma)$ as in Example 2.2.

3 Operational Logical Relations, Fibrationally

Having defined what an operational semantics for $\boldsymbol{\Lambda}$ is, we now focus on *operational* reasoning. In this section, we propose a general notion of an operational logical relation in terms of fibrations over (the underlying category of) an operational structure and prove that a general version of the fundamental lemma of logical relations holds for our operational logical relations. But before that, let us recall some preliminary notions on (bi)fibrations (we refer to [49, 55, 92] for more details).

3.1 Preliminaries on Fibrations

Let $p\colon \mathcal{E} \to \mathcal{B}$ be a functor and $f\colon X \to Y$ an arrow in $\mathcal{E}$ with $p(f) = u$. We say that f is *cartesian* over u if, for every arrow $h\colon Z \to Y$ in $\mathcal{E}$ such that $p(h) = u \circ v$, there is a unique arrow $g\colon Z \to X$ such that $p(g) = v$ and $h = f \circ g$. Dually, f is *cocartesian* over u if, for every arrow $h\colon X \to Z$ in $\mathcal{E}$ such that $p(h) = v \circ u$, there is a unique arrow $g\colon Y \to Z$ such that $p(g) = v$ and $h = g \circ f$. We say that f is *vertical* if u is an identity.

A *fibration* is a functor $p\colon \mathcal{E} \to B$ such that, for every object X in $\mathcal{E}$ and every arrow $u\colon I \to p(X)$ in $\mathcal{B}$, the exists a cartesian arrow over u with codomain X. Dually, an *opfibration* is a functor $p\colon \mathcal{E} \to \mathcal{B}$ such that for every object X in $\mathcal{E}$ and every arrow $f\colon p(X) \to I$ in $\mathcal{B}$, the exists a cocartesian arrow over u with domain X. A *bifibration* is a functor which is both a fibration and an opfibration. We refer to $\mathcal{E}$ and $\mathcal{B}$ as the domain and the base of the (bi/op)fibration. A (op)fibration is *cloven* if it comes together with a choice of (co)cartesian liftings: for an object X in $\mathcal{E}$, we denote by $\overline{u}_X\colon u^*X \to X$ the

chosen cartesian arrow over $u\colon I \to p(X)$ and by $\underline{u}_X\colon X \to u_!X$ the chosen cocartesian arrow over $u\colon p(X) \to I$. A bifibration is cloven if it has choices both for cartesian and cocartesian liftings. From now on, we assume all (bi/op)fibrations to be cloven.

Let $p\colon \mathcal{E} \to B$ be a functor and I an object in $\mathcal{B}$. The *fibre* over I is the category $\mathcal{E}_I$ where objects are objects X in $\mathcal{E}$ such that $p(X) = I$ and arrows are arrows $f\colon X \to Y$ in $\mathcal{E}$ such that $p(f) = \mathsf{id}_I$, namely, vertical arrows. Then, for every arrow $u\colon I \to J$, if p is a fibration, we have a functor $u^*\colon \mathcal{E}_J \to \mathcal{E}_I$ called *reindexing along* u and, if p is an opfibration, we have a functor $u_!\colon \mathcal{E}_I \to \mathcal{E}_J$ called *image along* u. If p is a bifibration, we have an adjunction $u_! \dashv u^*$.

▶ **Example 3.1.** n-ary predicates form a bifibration $\mathsf{Pred}_{\mathcal{S}et}\colon \mathsf{p}(\mathcal{S}et) \to \mathcal{S}et$, with $\mathsf{p}(\mathcal{S}et)$ having pairs of sets (X, A) with $A \subseteq X^n$ as objects and functions $f\colon X \to Y$ such that $A \subseteq (\prod f)^{-1}(B)$ as arrows $f\colon (X, A) \to (Y, B)$, and $\mathsf{Pred}_{\mathcal{S}et}(X, A) = X$. The cartesian and cocartesian liftings of f are given by inverse and direct images along $\prod f$, respectively. Special bifibrations are obtained for $n = 1$ (unary predicates) and $n = 2$ (binary relations). In those cases, we specialise the notation and write $\mathsf{Sub}_{\mathcal{S}et}\colon \mathsf{s}(\mathcal{S}et) \to \mathcal{S}et$ and $\mathsf{Rel}_{\mathcal{S}et}\colon \mathsf{r}(\mathcal{S}et) \to \mathcal{S}et$.

▶ **Example 3.2** (Weak subobjects)**.** Let $\mathcal{C}$ be a category with weak pullbacks. We define the bifibration $\Psi_{\mathcal{C}}\colon \mathsf{ws}(\mathcal{C}) \to \mathcal{C}$ of weak subobjects in $\mathcal{C}$ [47, 70]. Objects of $\mathsf{ws}(\mathcal{C})$ are pairs (X, R) where X is an object of $\mathcal{C}$ and R is an object of the poset reflection of the slice $\mathcal{C}/X$, hence it is an equivalence class $[\alpha]$ for an arrow $\alpha\colon A \to X$ in $\mathcal{C}$. An arrow $f\colon (X, [\alpha]) \to (Y, [\beta])$ is an arrow $f\colon X \to Y$ in $\mathcal{C}$ such that $f \circ \alpha = \beta \circ g$ for some arrow g in $\mathcal{C}$. Composition and identities in $\mathsf{ws}(\mathcal{C})$ are those of $\mathcal{C}$. The functor $\Psi_{\mathcal{C}}$ maps (X, R) to X and is the identity on arrows. It is easy to check that, for every arrow $f\colon X \to Y$ in $\mathcal{C}$ and every object $(X, [\alpha])$, the image along f is $f_!(X, [\alpha]) = (Y, [f \circ \alpha])$; and for every object $(Y, [\beta])$, the reindexing along f is $f^*(Y, [\beta]) = (X, [\beta'])$, where $f \circ \beta' = \beta \circ f'$ is a weak pullback square. Therefore, $\Psi_{\mathcal{C}}$ is a bifibration.

Note that, given objects $(X, [\alpha])$ and $(X, [\beta])$ in $\mathsf{ws}(\mathcal{C})$, there is at most one vertial arrow between them (the identity on X), hence we will write $[\alpha] \leq_X [\beta]$ when such arrow exists. Moreover, we have that $[\alpha] \leq_X [\beta]$ and $[\beta] \leq_X [\alpha]$ implies $[\alpha] = [\beta]$, by definition of poset reflection, hence the only vertical isomorphisms are identities. Finally, observe that the bifibrations $\Psi_{\mathcal{S}et}$ and $\mathsf{Sub}_{\mathcal{S}et}$ are equivalent in the sense that there is a functor $U\colon \mathsf{Sub}_{\mathcal{S}et} \to \mathsf{ws}(\mathcal{S}et)$ which is an equivalence satisfying $\Psi_{\mathcal{S}et} \circ U = \mathsf{Sub}_{\mathcal{S}et}$ and preserving (co)cocartesian arrows. The functor U maps (X, A) to $(X, [\iota_A])$ where $\iota_A\colon A \to X$ is the inclusion function.

Fibrations nicely carry a logical content: logical operations, in fact, can be described as categorical structures on the fibration. We now define the logical structure underlying logical relations, namely conjunctions, implications, and universal quantifiers. Recall that a fibration $p\colon \mathcal{E} \to \mathcal{B}$ has *finite products* if $\mathcal{E}$ and $\mathcal{B}$ have finite products and p strictly preserves them. We denote by $\dot{\times}$ and $\dot{1}$ finite products in $\mathcal{E}$ and recall [55] that in a fibration with finite products every fibre has finite products $\wedge$ and $\top$ preserved by reindexing functors; additionally, we have the isomorphisms $A \dot{\times} B \simeq \pi_1^* A \wedge \pi_2^* B$ and $\dot{1} \simeq \top_1$.

▶ **Definition 3.3.** *A fibration $p\colon \mathcal{E} \to \mathcal{B}$ with finite products is a* logical fibration *if it is fibred cartesian closed and has universal quantifiers, where:*

1. *p is* fibred cartesian closed *if every fibre $\mathcal{E}_I$ has exponentials, denoted by $X \Rightarrow Y$, and reindexings preserve them.*
2. *p has* universal quantifiers *if for every projection $\pi\colon I \times J \to I$ in $\mathcal{B}$ the reindexing functor $\pi^*\colon \mathcal{E}_I \to \mathcal{E}_{I \times J}$ has a right adjoint $\forall_J^I\colon \mathcal{E}_{I \times J} \to \mathcal{E}_I$ satisfying the Beck-Chevalley condition [55].*

Note that in a fibration with universal quantifiers, we have right adjoints along any tuple of distinct projections $\langle \pi_{i_1}, \ldots, \pi_{i_k} \rangle \colon I_1 \times \ldots \times I_n \to I_{i_1} \times \ldots \times I_{i_k}$, where $i_1, \ldots, i_k \in \{1, \ldots, n\}$ are all distinct. We denote such a right adjoint by $\forall_{\langle \pi_{i_1}, \ldots, \pi_{i_k} \rangle}$. The following proposition shows under which condition the fibration of weak subobjects is a logical fibration.

▶ **Proposition 3.4** ([55, 69]). *Let $\mathcal{C}$ be a category with finite products and weak pullbacks.*
1. *If $\mathcal{C}$ is cartesian closed, then $\Psi_{\mathcal{C}}$ has universal quantifiers;*
2. *If $\mathcal{C}$ is slicewise weakly cartesian closed, then $\Psi_{\mathcal{C}}$ is fibred cartesian closed.*

▶ **Example 3.5.** Since $\mathcal{S}et$ is locally cartesian closed, Proposition 3.4 implies that both $\Psi_{\mathcal{S}et}$ and $\mathsf{Sub}_{\mathcal{S}et}$ are logical fibrations. Also $\mathsf{Rel}_{\mathcal{S}et}$ is a logical fibration, as it can be obtained from $\mathsf{Sub}_{\mathcal{S}et}$ by pulling back along the product-preserving $\mathcal{S}et$-functor $X^2 = X \times X$.

A 2-category of (bi)fibrations. Fibrations can be organised in a 2-category. This is important because many standard categorical concepts can be internalised in any 2-category, hence we will be able to define them also for fibrations. In particular, we will be interested in (strong) monads on a fibration, as they will allow us to define effectful logical relations. We consider the 2-category **Fib** of fibrations defined as follows. Objects are fibrations $p \colon \mathcal{E} \to \mathcal{B}$. A 1-arrow $(F, G) : p \to q$ between fibrations $p \colon \mathcal{E} \to \mathcal{B}$and $q \colon \mathcal{D} \to \mathcal{C}$is a pair of functors $F \colon \mathcal{B} \to \mathcal{C}$ and $G \colon \mathcal{E} \to \mathcal{D}$ such that $F \circ p = q \circ G$.[5] A 2-arrow $(\phi, \psi) : (F, G) \Rightarrow (H, K)$ between 1-arrows $(F, G), (H, K) : p \to q$ is a pair of natural transformations $\phi : F \dot{\to} H$ and $\psi : G \dot{\to} K$ such that $\phi p = q\psi$.. Compositions and identities are defined componentwise. The 2-category **biFib** of bifibrations is defined in the same way. We can define [90] strong monads on fibrations as strong monads in the 2-category **Fib**. That is, a monad $\mathbb{T}$ on a fibration $p \colon \mathcal{E} \to \mathcal{B}$ consists of the following data: a 1-arrow $(T, S) : p \to p$ and two 2-arrows $(\mu_T, \mu_S) : (T^2, S^2) \Rightarrow (T, S)$, $(\eta_T, \eta_S) : (\mathsf{Id}_{\mathcal{E}}, \mathsf{Id}_{\mathcal{B}}) \Rightarrow (T, S)$, and $(\mathsf{st}_T, \mathsf{st}_S) : (- \times T -, - \dot{\times} S -) \Rightarrow (T(- \times -), S(- \dot{\times} -))$ such that $(T, \mu_T, \eta_T, \mathsf{st}_T)$ is a strong monad on $\mathcal{B}$ and $(S, \mu_S, \eta_S, \mathsf{st}_S)$ is a strong monad on $\mathcal{E}$. In particular, given a strong monad $\mathbb{T} = (T, \mu, \eta, \mathsf{st})$ on $\mathcal{B}$, a lifting of $\mathbb{T}$ to $p \colon \mathcal{E} \to \mathcal{B}$ is a tuple $\dot{\mathbb{T}} = (\dot{T}, \dot{\mu}, \dot{\eta}, \dot{\mathsf{st}})$ such that $(T, \dot{T}, \mu, \dot{\mu}, \eta, \dot{\eta}, \mathsf{st}, \dot{\mathsf{st}})$ is a strong monad on $p \colon \mathcal{E} \to \mathcal{B}$. We write $\dot{\gg\!=}$ for the lifting of $\gg\!=$.

3.2 Operational Logical Relations and Their Fundamental Lemma

We are now ready to define a general notion of an operational logical relation. Let us consider an operational structure over a cartesian category $\mathcal{B}$ with a strong monad $\mathbb{T}$ on $\mathcal{B}$, as in Definition 2.3. Let $p \colon \mathcal{E} \to \mathcal{B}$ be a logical fibration and $\dot{\mathbb{T}}$ be a lifting of $\mathbb{T}$ to p.

▶ **Definition 3.6.** *A logical relation is a mapping R from objects of $\mathcal{S}yn$ to objects of $\mathcal{E}$ such that $p(Rx) = Sx$, for any object x of $\mathcal{S}yn$,*[6] *and the following hold.*

$$R\mathbf{1} = \top_{S\mathbf{1}} \qquad R(\sigma \times \tau) = \boldsymbol{p_1^*}(R\sigma) \wedge \boldsymbol{p_2^*}(R\tau) \qquad R\underline{\sigma} = \boldsymbol{e}^*(\dot{T}(R\sigma))$$
$$R(\overrightarrow{x : \sigma}) = \dot{\prod} \overrightarrow{R\sigma} \qquad R(\sigma \to \underline{\tau}) = \forall_{\pi_1} \pi_2^*(R\sigma) \Rightarrow \boldsymbol{\beta}^*(R\underline{\tau})$$

Notice that giving a logical relation essentially amounts to specify the action of R on basic types, since the action of R on complex types is given by Definition 3.6. The defining clauses of a logical relation exploit both the logic of a (logical) fibration and the operational

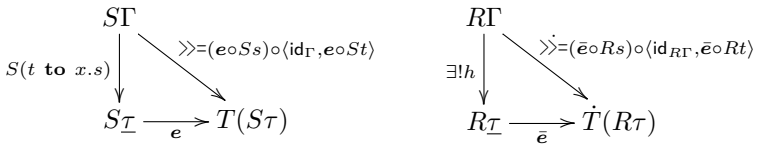
[5] Note that we do not require G to preserve cartesian arrows.
[6] Actually, it suffices to have $p(R\zeta) = S\zeta$ for basic types only, as the second part of the definition assures it for other types.

semantics of $\mathbf{\Lambda}$. The reader should have recognised in Definition 3.6 the usual definition of a logical relation, properly generalised to rely on the logic of a fibration only. For instance, $R\underline{\sigma}$ intuitively relates computations whose evaluations are related by the lifting of the monad. Notice also that the clause of arrow types has a higher logical complexity than other clauses, as it involves *two* logical connectives, viz. implication and universal quantification.

Operational logical relations come with their so-called *fundamental lemma*, which states that (open) terms maps (via substitution) related values to related terms. In our abstract framework that to any term t we can associate a suitable arrow Rt in $\mathcal{E}$ lying above St. To prove our general version of the fundamental theorem, we have to assume it for the parameters of our calculus (constants of basic types and generic effects). Accordingly, we say that a logical relation R is ($\mathbf{\Lambda}$-)*stable* if: (i) for every constant c of a base type ζ, we have an arrow $\dot{\boldsymbol{c}} : \dot{1} \to R\zeta$ above $\boldsymbol{c}$; (ii) we have an arrow $\dot{\boldsymbol{\gamma}} : \dot{1} \to \dot{T}(R\sigma)$ above $\boldsymbol{\gamma}$.

▶ **Theorem 3.7** (Fundamental Lemma). *Let R be a stable logical relation. The map R extends to a diagram $R : \mathcal{Syn} \to \mathcal{E}$ such that $p \circ R = S$. In particular, for any term $\Gamma \vdash t : \sigma$, there is an arrow $Rt : R\Gamma \to R\underline{\sigma}$ in $\mathcal{E}$ above St (similarly, for values).*

Proof sketch. Given $\Gamma \vdash t : \sigma$, we construct the desired arrow Rt by induction on t. The case for values lifts commutative triangles in Figure 2 using the universal property of cartesian liftings of interaction arrows and then constructs the desired arrow using the logical structure of p and $R\sigma$. The case of terms, just lifts the commutative diagrams in Figure 2 using the universal property of the cartesian lifting of the $\boldsymbol{e}$. As a paradigmatic example, we show the case of sequencing. First, let us notice that for any type σ, the evaluation arrow $\boldsymbol{e} : S\underline{\sigma} \to T(S\sigma)$ gives a cartesian arrow $\bar{\boldsymbol{e}} : \boldsymbol{e}^*(\dot{T}(R\sigma)) \to \dot{T}(R\sigma)$, i.e. $\bar{\boldsymbol{e}} : R\underline{\sigma} \to \dot{T}(R\sigma)$ (since $R\underline{\sigma} = \boldsymbol{e}^*(\dot{T}(R\sigma))$). Let us now consider the case of $\Gamma \vdash t \ \mathbf{to}\ x.s : \underline{\tau}$ as obtained from $\Gamma \vdash t : \sigma$ and $\Gamma, x : \sigma \vdash s : \tau$. By induction hypothesis, we have arrows $Rt : R\Gamma \to R\underline{\sigma}$ and $Rs : R\Gamma \dot{\times} R\sigma \to R\underline{\tau}$. By postcomposing the former with $\bar{\boldsymbol{e}}$, we obtain the arrow $\bar{\boldsymbol{e}} \circ Rt : R\Gamma \to \dot{T}(R\sigma)$, and thus $\langle \mathsf{id}_{R\Gamma}, \bar{\boldsymbol{e}} \circ Rt \rangle : R\Gamma \to R\Gamma \dot{\times} \dot{T}(R\sigma)$. In a similar fashion, we have $\bar{\boldsymbol{e}} \circ Rs : R\Gamma \dot{\times} R\sigma \to \dot{T}(R\tau)$ and thus, using the extension of the monad, $\dot{\gg\!=}(\bar{\boldsymbol{e}} \circ Rs) : R\Gamma \dot{\times} \dot{T}(R\sigma) \to \dot{T}(R\tau)$. Altogether, we obtain the arrow $\dot{\gg\!=}(\bar{\boldsymbol{e}} \circ Rs) \circ \langle \mathsf{id}_{R\Gamma}, \bar{\boldsymbol{e}} \circ Rt \rangle : R\Gamma \to \dot{T}(R\tau)$. Using the commutative diagram of sequencing in Figure 2 and the very definition of a fibration, we obtain



We choose h as $R(t \ \mathbf{to}\ x.s)$. ◀

We can now instantiate Theorem 3.7 with the operational structures and fibrations seen so far to recover traditional logical relations (and their fundamental lemmas). For instance, the operational structure of Example 2.5 and the fibraration obtained by pulling back $\mathsf{Rel}_{\mathcal{Set}}$ along the forgetful from $\mathcal{Meas}$ to $\mathcal{Set}$ (together with a lifting of the Giry monad [62, 60]) give operational logical relations for stochastic λ-calculi. Theorem 3.7 then gives compositionality (i.e. congruence and substitutivity) of the logical relation. But that is not the end of the story. In fact, our general results go beyond the realm of traditional logical relations.

4 The fibration of differential relations

In this section, we describe the construction of the *fibration of differential relations*, which can serve as a fibrational foundation of *differential logical relations* [31, 22, 23, 25] (DLRs), a recently introduced form of logical relations defining higher-order distances between programs. DLRs are ternary relations relating pairs of terms with elements representing distances between them: such distances, however, need not be numbers. More precisely, with each type σ one associates a set $(\!|\sigma|\!)$ of (higher-order) distances between terms of type σ, and then defines DLRs as relating terms of type σ with distances in $(\!|\sigma|\!)$ between them. Elements of $(\!|\sigma|\!)$ reflect the interactive complexity of programs, the latter being given by the type σ. Thus, for instance, the main novelty of DLRs is that a distance between two values $\lambda x.t, \lambda x.s$ of type $\sigma \to \tau$ is not *just* a number, but a function $dt : \mathcal{V}_\sigma \times (\!|\sigma|\!) \to (\!|\tau|\!)$ mapping a value v and an error/perturbation dv to an error/perturbation $dt(v, dv)$. A DLR then relates $\lambda x.t$ and $\lambda x.s$ to dt if for all values v, w related to dv (meaning that v and w are dv-apart), then $t[v/x]$ and $s[w/x]$ are related to $dt(v, dv)$.

Semantically, DLRs give rise to generalised distance spaces and differential extensions of monads, the former being relational structures $(X, (\!|X|\!), \delta_X)$ with $\delta_X \subseteq X \times (\!|X|\!) \times X$ acting as the semantic counterpart of a DLR, and the latter being reminiscent[7] of extensions of monads to the category of generalised distance spaces. Here, we show how our general notion of an operational logical relation subsumes the one of a DLR (and, consequently, that the fundamental lemma of DLRs is an instance of Theorem 3.7). Additionally, we show how differential extensions are precisely liftings of monads to the fibration of differential relations and how the so-called coupling-based differential extension [25] is an instance of a general monadic lifting to such a fibration, viz. the well-known Barr lifting properly fitted to a differential setting.

4.1 Going Differential, Fibrationally

Let $p\colon \mathcal{E} \to \mathcal{B}$ be a fibration with finite products.[8] We define the category $\mathsf{dr}(p)$ of *differential relations* in p as follows:

Objects are triples $\mathsf{X} = (|\mathsf{X}|, (\!|\mathsf{X}|\!), \delta_\mathsf{X})$, where $|\mathsf{X}|$ and $(\!|\mathsf{X}|\!)$ are objects in $\mathcal{B}$ and δ_X is an object in the fibre $\mathcal{E}_{|\mathsf{X}| \times (\!|\mathsf{X}|\!) \times |\mathsf{X}|}$.

Arrows $f\colon \mathsf{X} \to \mathsf{Y}$ are triples $f = (|f|, \mathsf{d}f, \varphi_f)$, where $|f|\colon |\mathsf{X}| \to |\mathsf{Y}|$ and $\mathsf{d}f\colon |\mathsf{X}| \times (\!|\mathsf{X}|\!) \to (\!|\mathsf{Y}|\!)$ are arrows in $\mathcal{B}$, and $\varphi_f\colon \delta_\mathsf{X} \to \delta_\mathsf{Y}$ is an arrow in $\mathcal{E}$ over $\langle |f|\pi_1, \mathsf{d}f\langle \pi_1, \pi_2\rangle, |f|\pi_3\rangle$.

Composition of arrows $f\colon \mathsf{X} \to \mathsf{Y}$ and $g\colon \mathsf{Y} \to \mathsf{Z}$ is defined thus:

$$|g \circ f| = |g| \circ |f| \qquad \mathsf{d}(g \circ f) = \mathsf{d}g \circ \langle |f|\pi_1, \mathsf{d}f\rangle \qquad \varphi_{g\circ f} = \varphi_g \circ \varphi_f.$$

Identity on X is given by $\mathsf{id}_\mathsf{X} = (\mathsf{id}_{|\mathsf{X}|}, \pi_2, \mathsf{id}_{\delta_\mathsf{X}})$.

▶ **Proposition 4.1.** *$\mathsf{dr}(p)$ is a category with finite products.*

▶ Remark 4.2. Note that an arrow $f\colon \mathsf{X} \to \mathsf{Y}$ in $\mathsf{dr}(p)$ can be equivalently described as a triple $(|f|, \mathsf{D}f, \varphi_f)$ where $|f|$ is as before, $\mathsf{D}f\colon |\mathsf{X}| \times (\!|\mathsf{X}|\!) \to |\mathsf{Y}| \times (\!|\mathsf{Y}|\!)$ is such that $|f| \circ \pi_1 = \pi_1 \circ \mathsf{D}f$ and $\varphi_f\colon \delta_\mathsf{X} \to \delta_\mathsf{Y}$ is above $\mathsf{D}f \times |f|\colon |\mathsf{X}| \times (\!|\mathsf{X}|\!) \times |\mathsf{X}| \to |\mathsf{Y}| \times (\!|\mathsf{Y}|\!) \times |\mathsf{Y}|$. This presentation is perhaps more in the spirit of fibrations, but we opted for the other one, which follows the original presentation of generalised distance spaces [25].

[7] Whether differential extensions indeed define a monadic lifting is left as an open problem in [25].
[8] Actually, to carry out our construction, binary products in the base are enough, but we use a richer structure as we need it in the following part of this paper.

In the following, we denote by $\nabla\mathsf{X}$ the object $|\mathsf{X}| \times (\!|\mathsf{X}|\!) \times |\mathsf{X}|$ of $\mathcal{B}$ and by ∇f the arrow $\langle |f|\pi_1, \mathsf{d}f\langle\pi_1,\pi_2\rangle, |f|\pi_3\rangle$. These data define a functor $\nabla\colon \mathsf{dr}(p) \to \mathcal{B}$.

▶ **Example 4.3.** For the fibration $\mathsf{Sub}_{\mathcal{S}et}\colon \mathsf{s}(\mathcal{S}et) \to \mathcal{S}et$ of Example 3.1, the category $\mathsf{dr}(\mathsf{Sub}_{\mathcal{S}et})$ is the category of generalised distance spaces [25]. An object in $\mathsf{dr}(\mathsf{Sub}_{\mathcal{S}et})$ is essentially a triple (X, V, R) consisting of a set X of points, a set V of distance values, and a ternary relation $R \subseteq X \times V \times X$ specifying at which distance two elements of X are related: that is, $(x, v, y) \in R$ means that x and y are related at distance v. For instance, a metric $d\colon X \times X \to [0,\infty]$ on X can be seen as a ternary relation $R_d \subseteq X \times [0,\infty] \times X$ defined by $(x, v, y) \in R_d$ iff $d(x, y) \leq v$. An arrow from (X, V, R) to (Y, U, S) in $\mathsf{dr}(\mathsf{Sub}_{\mathcal{S}et})$ consists of a function $|f|\colon X \to Y$ transforming points together with a function $\mathsf{d}f\colon X \times V \to U$ transforming distance values such that, for all $x, y \in X$ and $v \in V$, $(x, v, y) \in R$ implies $(|f|(x), \mathsf{d}f(x, v), |f|(y)) \in S$.

The assignments $\mathsf{DRel}^p(\mathsf{X}) = |\mathsf{X}|$ and $\mathsf{DRel}^p(f) = |f|$ determine a functor $\mathsf{DRel}^p\colon \mathsf{dr}(p) \to \mathcal{B}$.

▶ **Proposition 4.4.** *The functor* $\mathsf{DRel}^p\colon \mathsf{dr}(p) \to \mathcal{B}$ *is a fibration with finite products.*

Proof. Define the cartesian lifting along an arrow $u\colon I \to J$ of an object Y with $|\mathsf{Y}| = J$ by $(u, \pi_2, \overline{\langle u\pi_1, \pi_2, u\pi_3\rangle}_{\delta_\mathsf{Y}})\colon u^*\mathsf{Y} \to \mathsf{Y}$, where $u^*\mathsf{Y} = (I, (\!|\mathsf{Y}|\!), \langle u\pi_1, \pi_2, u\pi_3\rangle^*\delta_\mathsf{Y})$. ◀

▶ Remark 4.5. The fibration DRel^p can be also obtained from the *simple fibration* [55] $s\colon \mathsf{s}(\mathcal{B}) \to \mathcal{B}$, where objects of $\mathsf{s}(\mathcal{B})$ are pairs (I, X) of objects in $\mathcal{B}$; arrows $(u, f)\colon (I, X) \to (J, Y)$ are pairs of arrows $u\colon I \to J$ and $f\colon I \times X \to Y$ in $\mathcal{B}$; and s projects both objects and arrows on the first component. Indeed, we have that $\mathsf{DRel}^p = s \circ p'$ where $p'\colon \mathsf{dr}(p) \to \mathsf{s}(\mathcal{B})$ is obtained by the pullback $\begin{array}{ccc} \mathsf{dr}(p) & \longrightarrow & \mathcal{E} \\ {\scriptstyle p'}\downarrow & p.b. & \downarrow{\scriptstyle p} \\ \mathsf{s}(\mathcal{B}) & \longrightarrow & \mathcal{B} \end{array}$, with the functor $\mathsf{s}(\mathcal{B}) \to \mathcal{B}$ mapping (I, X) to $I \times X \times I$ and (u, f) to $\langle u\pi_1, f\langle\pi_1,\pi_2\rangle, u\pi_3\rangle$. Therefore, we have $p'(\mathsf{X}) = (|\mathsf{X}|, (\!|\mathsf{X}|\!))$ and $p'(f) = (|f|, \mathsf{d}f)$. This is somewhat similar to the simple coproduct completion of a fibration [53]. We leave a precise comparison between these constructions for future work.

We now show under which conditions the fibration DRel^p is a logical fibration.

▶ **Proposition 4.6.** *If* $p\colon \mathcal{E} \to \mathcal{B}$ *is a logical fibration and* $\mathcal{B}$ *is cartesian closed, then* DRel^p *is a logical fibration.*

We report the definition of the logical structure on DRel^p: let $\mathsf{X}, \mathsf{Y}, \mathsf{Z}$ objects in $\mathsf{dr}(p)$ with $|\mathsf{X}| = |\mathsf{Y}| = I$ and $|\mathsf{Z}| = I \times J$.

$$\widehat{\top}_I = (I, 1, \top_{I\times 1\times I}) \qquad \mathsf{X}\widehat{\wedge}_I\mathsf{Y} = (I, (\!|\mathsf{X}|\!) \times (\!|\mathsf{Y}|\!), \langle\pi_1,\pi_2,\pi_4\rangle^*\delta_\mathsf{X} \wedge \langle\pi_1,\pi_3,\pi_4\rangle^*\delta_\mathsf{Y})$$
$$\mathsf{X}\widehat{\Rightarrow}_I\mathsf{Y} = (I, [(\!|\mathsf{X}|\!), (\!|\mathsf{Y}|\!)], \forall_{\langle\pi_1,\pi_2,\pi_4\rangle}(\langle\pi_1,\pi_3,\pi_4\rangle^*\delta_\mathsf{X} \Rightarrow \langle\pi_1, \mathsf{ev}^{(\!|\mathsf{X}|\!)}_{(\!|\mathsf{Y}|\!)}\langle\pi_2,\pi_3\rangle, \pi_4\rangle^*\delta_\mathsf{Y}))$$
$$\widehat{\forall^I_J}\mathsf{Z} = (I, [J, (\!|\mathsf{Z}|\!)], \forall_{\langle\pi_1,\pi_3,\pi_5\rangle}(\langle\pi_1,\pi_2, \mathsf{ev}^J_{(\!|\mathsf{Z}|\!)}\langle\pi_3,\pi_2\rangle, \pi_4, \pi_5\rangle^*\delta_\mathsf{Z}))$$

▶ **Example 4.7.** Let us consider the fibration $\mathsf{DRel}^{\mathsf{Sub}_{\mathcal{S}et}}\colon \mathsf{dr}(\mathsf{Sub}_{\mathcal{S}et}) \to \mathcal{S}et$ and instantiate the above constructions to it. We have $\widehat{\top}_X = (X, 1, X \times 1 \times X)$, that is, in $\widehat{\top}_X$ there is just one distance value and all elements of X are related. Consider now objects $\mathsf{X} = (X, V, R)$ and $\mathsf{Y} = (X, U, S)$ in $\mathsf{dr}(\mathsf{Sub}_{\mathcal{S}et})$. Then, we have $\mathsf{X}\widehat{\wedge}\mathsf{Y} = (X, V \times U, R \sqcap S)$ and $\mathsf{X}\widehat{\Rightarrow}\mathsf{Y} = (X, [V, U], R \to S)$, where

- $(x,(v,u),y) \in R \sqcap S$ iff $(x,v,y) \in R$ and $(x,u,y) \in S$,
- $(x,f,y) \in R \to S$ iff, for all $v \in V$, $(x,v,y) \in R$ implies $(x,f(v),y) \in S$.

That is, two elements x and y are related in $R \sqcap S$ at a distance (v,u) such that x and y are at distance v in R and u in S. Instead, x and y are related in $R \to S$ at a distance f transforming distances in V into distances in U, respecting R and S. Finally, if $\mathsf{Z} = (X \times Y, V, R)$ is an object in $\mathsf{dr}(\mathsf{Sub}_{\mathcal{C}}\mathcal{S}et])$, then we have $\widehat{V^X_Y}\mathsf{Z} = (X, [Y,V], Y \to R)$, where $(x,f,x') \in Y \to R$, iff for all $y, y' \in Y$, $((x,y), f(y), (x',y')) \in R$. That is, elements x and x' are related by $Y \to R$ at a distance f returning for each $y \in Y$ a distance in V such that R relates (x,y) and (x',y') at distance $f(y)$, for each $y, y' \in Y$.

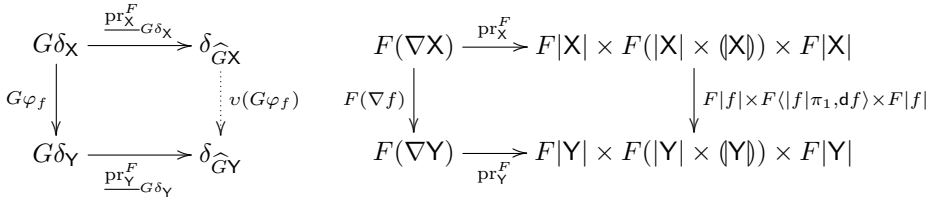
We now extend this construction to 1- and 2-arrows. To do so, we work with bifibrations whose bases have finite products. Let p and q be such bifibrations and $(F,G) : p \to q$ be a 1-arrow in **biFib**. We define a functor $\widehat{G}\colon \mathsf{dr}(p) \to \mathsf{dr}(q)$ thus: for X an object in $\mathsf{dr}(p)$, we set

$$|\widehat{G}\mathsf{X}| = F|\mathsf{X}| \qquad (\!|\widehat{G}\mathsf{X}|\!) = F(|\mathsf{X}| \times (\!|\mathsf{X}|\!)) \qquad \delta_{\widehat{G}\mathsf{X}} = \mathrm{pr}^F_{\mathsf{X}\,!} G\delta_{\mathsf{X}},$$

where $\mathrm{pr}^F_{\mathsf{X}} = \langle F\pi_1, F\langle \pi_1, \pi_2\rangle, F\pi_3\rangle\colon F(|\mathsf{X}| \times (\!|\mathsf{X}|\!) \times |\mathsf{X}|) \to F|\mathsf{X}| \times F(|\mathsf{X}| \times (\!|\mathsf{X}|\!)) \times F|\mathsf{X}|$ is an arrow in $\mathcal{B}$. For every arrow $f\colon \mathsf{X} \to \mathsf{Y}$ in $\mathsf{dr}(p)$, we set

$$|\widehat{G}f| = F|f| \qquad \mathsf{d}(\widehat{G}f) = (F\langle |f|\pi_1, \mathsf{d}f\rangle)\pi_2 \qquad \varphi_{\widehat{G}f} = \upsilon(G\varphi_f),$$

where $\upsilon(G\varphi_f)$ is the unique arrow over $F|f| \times F\langle |f|\pi_1, \mathsf{d}f\rangle \times F|f|$ making the diagram on the left commute, which exists as the diagram on the right commutes and $\underline{\mathrm{pr}^F_{\mathsf{X}}}_{G\delta_{\mathsf{X}}}$ is cocartesian.



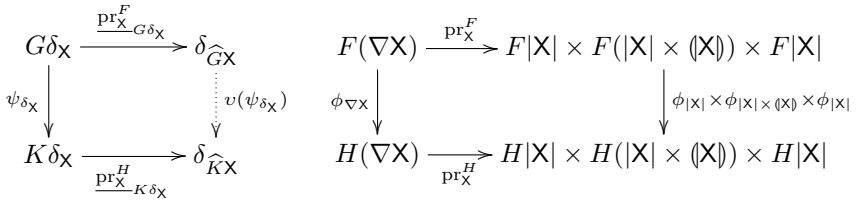
Notice that cocartesian liftings are essential to appropriately define $\widehat{G}$ on the relational part of X and f, as we do not assume any compatibility with products for F.

▶ **Proposition 4.8.** *$(F,\widehat{G}) : \mathsf{DRel}^p \to \mathsf{DRel}^q$ is a 1-arrow in* **Fib**.

Similarly, given a 2-arrow $(\phi,\psi) : (F,G) \Rightarrow (H,K)$ in **biFib** between 1-arrows $(F,G) : p \to q$ and $(H,K) : p \to q$, we define a natural transformation $\widehat{\psi} : \widehat{G} \dot{\to} \widehat{K}$ as follows: for every object X in DRel^p, we set:

$$|\widehat{\psi}_{\mathsf{X}}| = \phi_{|\mathsf{X}|} \qquad \mathsf{d}(\widehat{\psi}_{\mathsf{X}}) = \phi_{|\mathsf{X}|\times(\!|\mathsf{X}|\!)}\pi_2 \qquad \varphi_{\widehat{\psi}_{\mathsf{X}}} = \upsilon(\psi_{\delta_{\mathsf{X}}}),$$

where $\upsilon(\psi_{\delta_{\mathsf{X}}})$ is the unique arrow over $\phi_{|\mathsf{X}|} \times \phi_{|\mathsf{X}|\times(\!|\mathsf{X}|\!)} \times \phi_{|\mathsf{X}|}$ making the diagram on the left commute, which exists as the diagram on the right commutes and $\underline{\mathrm{pr}^F_{\mathsf{X}}}_{G\delta_{\mathsf{X}}}$ is cocartesian.



▶ **Proposition 4.9.** $(\phi, \widehat{\psi}) : (F, \widehat{G}) \Rightarrow (H, \widehat{K})$ *is a 2-arrow in* **Fib**.

We are getting closer to the definition of a 2-functor DR from (certain) bifibrations to fibrations given by the following assignments:

$$\mathrm{DR}(p) = \mathsf{DRel}^p \qquad \mathrm{DR}(F, G) = (F, \widehat{G}) \qquad \mathrm{DR}(\phi, \psi) = (\psi, \widehat{\psi}).$$

However, this is not the case as DR does not preserve identity 1-arrows. Indeed, given the identity $(\mathsf{Id}_{\mathcal{B}}, \mathsf{Id}_{\mathcal{E}})$ on a bifibration $p \colon \mathcal{E} \to \mathcal{B}$, we have that for every object X in $\mathsf{dr}(p)$, $\widehat{\mathsf{Id}}_{\mathcal{E}}(\mathsf{X}) = (|\mathsf{X}|, |\mathsf{X}| \times (\!|\mathsf{X}|\!), \mathrm{pr}^{\mathsf{Id}_{\mathcal{B}}}_{\mathsf{X}!}\delta_{\mathsf{X}})$, which is not isomorphic to X, in general.

We can recover a form of functoriality by restricting to a 2-subcategory of **biFib**, notably, considering only 1-arrows which are cocartesian. We say that a 1-arrow $(F, G) : p \to q$ in **biFib** is *cocartesian* if the functor G preserves cocartesian arrows. This implies that G commutes with cocartesian liftings up to isomorphism. Denote by $\mathbf{biFib}_c$ the 2-subcategory of **biFib** where objects have finite products and 1-arrows are cocartesian.

▶ **Theorem 4.10.** *DR*: $\mathbf{biFib}_c \to \mathbf{Fib}$ *is a lax functor.*

Note that we just have a *lax functor*, as identities and composition of 1-arrows are preserved only up to a family of a mediating 2-arrow. Nonetheless, this is enough to get important properties of the construction.

4.2 Lifting monads: the differential extension

An important problem when dealing with effectful languages is the lifting of monads from the category where the semantics of the language is expressed to the category used to reason about programs, e.g., the domain of a fibration or a category of relations. The most famous one is perhaps the so-called *Barr extension* [8] of a $\mathcal{Set}$-monad to the category of (endo)relations, which is fibred over $\mathcal{Set}$ (other notions of lifting include $\top\top$- and codensity lifting [62, 60]). In the differential setting, the notion of a *differential extension* has been recently proposed [25] as a way to lift monads to generalised distance spaces. To what extent such a construction is canonical and whether it defines an actual monadic lifting, however, have been left as open questions. Here, we answer both questions in the affirmative.

▶ **Definition 4.11.** *Let* $p \colon \mathcal{E} \to \mathcal{B}$ *be a fibration with finite products and* $\mathbb{T} = (T, \mu, \eta, \mathsf{st})$ *be a strong monad on* $\mathcal{B}$*. A* differential extension *of* $\mathbb{T}$ *along* p *is a lifting* $\dot{\mathbb{T}} = (\dot{T}, \dot{\mu}, \dot{\eta}, \dot{\mathsf{st}})$ *of* $\mathbb{T}$ *along the fibration* $\mathsf{DRel}^p \colon \mathsf{dr}(p) \to \mathcal{B}$*.*

A differential extension of $\mathbb{T}$ along the fibration p is thus a monad on the category of differential relations in p which is above $\mathbb{T}$ with respect to the fibration DRel^p. We describe two techniques to build differential extensions of monads. The first one is an immediate consequence of Theorem 4.10. Indeed, DR: $\mathbf{biFib}_c \to \mathbf{Fib}$ being a lax functor, every cocartesian monad on a bifibration p induces a monad on the fibration DRel^p, hence a differential extension. This follows from general properties of lax functors [9, 91].

▶ **Corollary 4.12.** *Let* $\mathbb{T} = (T, S, \eta_T, \eta_S, \mu_T, \mu_S)$ *be a monad on* p *in* $\mathbf{biFib}_c$*. Then,* $\widehat{\mathbb{T}} = (T, \widehat{S}, \eta_T, \widehat{\eta_S}, \mu_T, \widehat{\mu_S})$ *is a monad on* DRel^p *in* **Fib**.

To get a lifting of strong monads, we need an additional hypothesis: we have to require that the product functor $\dot{\times}$ on the total category $\mathcal{E}$ preserves cocartesian arrows, as often happens when dealing with monoidal bifibrations [85, 73].[9] This is sensible as the action of DR on 1-arrows is defined using cocartesian liftings.

[9] Here the monoidal structure is given just by cartesian product.

▶ **Theorem 4.13.** *Let $p\colon \mathcal{E} \to \mathcal{B}$ be a bifibration with finite products where $\dot{\times}$ preserves cocartesian arrows and $\mathbb{T} = (T, S, \mu_T, \mu_S, \eta_T, \eta_S, \mathsf{st}_T, \mathsf{st}_S)$ be a strong monad on p. Then, $(\widehat{S}, \widehat{\mu_S}, \widehat{\eta_S}, \widehat{\mathsf{st}_S})$ is a differential extension of $(T, \mu_T, \eta_T, \mathsf{st}_T)$.*

In other words, this result provides us with a differential extension of a monad (T, μ_T, η_T) starting from a usual extension (S, μ_S, η_S) along the fibrations p. Therefore, to build a differential extension of (T, μ_T, η_T) we can use existing techniques to lift it, obtaining a monad on the fibration p and then apply our construction.

We conclude this section by instantiating this technique to a special class of bifibrations, notably, bifibrations of weak subobjects as defined in Example 3.2. The resulting construction applies to (strong) monads on categories with weak pullbacks and finite products, and provides a differential version of the Barr extension, which we dub a *differential Barr extension*.

First of all, we show that the construction of the bifibration of weak subobjects extends to a 2-functor. Let us denote by $\mathbf{Cat}_{cwp}$ the 2-category of categories with weak pullbacks and finite products, functors, and natural transformations. Given a functor $F\colon \mathcal{C} \to \mathcal{D}$ in $\mathbf{Cat}_{cwp}$, define $\overline{F}\colon \mathsf{ws}(\mathcal{C}) \to \mathsf{ws}(\mathcal{D})$ as $\overline{F}(X, [\alpha]) = (FX, [F\alpha])$ and $\overline{F}f = Ff$. It is easy to check that this is indeed a functor. Moreover, note that if $f\colon (X, [\alpha]) \to (Y, [\beta])$ is cocartesian in $\Psi_{\mathcal{C}}$, that is, $[\beta] = [f\alpha]$, then $\overline{F}f\colon (FX, [F\alpha]) \to (FY, [F\beta])$ is cocartesian in $\Psi_{\mathcal{D}}$, as $[F\beta] = [Ff \circ F\alpha]$. Consider now a natural transformation $\phi : F \dot{\to} G$ in $\mathbf{Cat}_{cwp}$ and define $\overline{\phi}_{(X,[\alpha])} : \overline{F} \dot{\to} \overline{G}$ as $\overline{\phi}_{(X,[\alpha])} = \phi_X$. This is well-defined because, if $\alpha\colon A \to X$, then we have $\phi_X \circ F\alpha = G\alpha \circ \phi_A$, by naturality of ϕ. We define $\Psi_F = (F, \overline{F})$ and $\Psi_\phi = (\phi, \overline{\phi})$.

▶ **Proposition 4.14.** *$\Psi\colon \mathbf{Cat}_{cwp} \to \mathbf{biFib}_c$ is a strict 2-functor.*

Therefore, composing Ψ and DR we get a lax functor from $\mathbf{Cat}_{cwp}$ to $\mathbf{Fib}$, this way extending Theorem 4.13.

▶ **Theorem 4.15.** *Let $\mathbb{T} = (T, \mu, \eta, \mathsf{st})$ be a strong monad on a category $\mathcal{C}$ with weak pullbacks and finite products. Then, $(\widehat{\widehat{T}}, \widehat{\widehat{\mu}}, \widehat{\widehat{\eta}}, \widehat{\widehat{\mathsf{st}}})$ is a differential extension of $\mathbb{T}$ along $\Psi_{\mathcal{C}}$.*

▶ **Example 4.16.** Let (T, μ, η) a monad on $\mathcal{Set}$. The *coupling-based lifting* of (T, μ, η) to $\mathsf{dr}(\mathsf{Sub}_{\mathcal{Set}})$ [25] maps an object (X, V, R) to $(TX, T(X \times V), \tau(R))$ where $(x, v, y) \in \tau(R)$ iff there exists $\varphi \in TR$ such that $T\pi_1(T\iota_R(\varphi)) = x$, $T\langle \pi_1, \pi_2\rangle(T\iota_R(\varphi)) = v$, and $T\pi_3(T\iota_R(\varphi)) = y$. Here, $\iota_R\colon R \to X \times V \times X$ denotes the inclusion function and the element φ is called a (ternary) coupling, borrowing the terminology from optimal transport [96]. Basically, this lifting states that monadic elements x and y are related with monadic distance v iff we can find a coupling φ that projected along the first and third component gives x and y, respectively, and projected along the first two components gives the monadic distance v. Relying on the equivalence between $\mathsf{Sub}_{\mathcal{Set}}$ and $\Psi_{\mathcal{Set}}$, we get that the above lifting of (T, μ, η) is an instance of the differential Barr extension $\widehat{\widehat{T}}\colon \mathsf{dr}(\mathsf{ws}(\mathcal{Set})) \to \mathsf{dr}(\mathsf{ws}(\mathcal{Set}))$. Indeed, for an object $(X, V, [\alpha])$ in $\mathsf{dr}(\Psi_{\mathcal{Set}})$ we can choose a canonical representative $\iota_R\colon R \to X \times V \times X$ such that $[\alpha] = [\iota_R]$, where R is the image of α and ι_R is the inclusion function.

5 Conclusion and Future Work

We have shown how fibrations can be used to give a uniform account to operational logical relations for higher-order languages with generic effects and a rather liberal operational semantics. Our framework encompasses both traditional, set-based logical relations and logical relations on non-cartesian-closed categories – such as the one of measurable spaces – as well as the recently introduced differential logical relations. In particular, our analysis sheds

a new light on the mathematical foundation of both pure and effectful differential logical relations. Further examples of logical relations that can be described in our framework include classic Kripke logical relations [74] (take fibrations of poset-indexed relations) as well as logical relations for information flow (the latter could be approached both as suitable Kripke logical relations or by considering a shallow semantics on the category of classified sets [63]). Additionally, since differential logical relations can be used to reason about nontrivial notions such as program sensitivity and cost analysis [25], our framework can be used for reasoning about the same notions too.

Even if general, the vehicle calculus of this work lacks some important programming language features, such as recursive types and polymorphism. Our framework being operational, the authors suspect that the addition of polymorphism should not be problematic, whereas the addition of full recursion may require to come up with abstract notions of step-indexed logical relations [5, 32]. In general, the proposed framework looks easily extensible and adaptable. To reason about a given calculus, one should pick a fibration with a logical structure supporting the kind of analysis one is interested in and whose base category has enough structure to model its interactive behaviour. For instance, in this work we just need products and a monad on the base category but, e.g., to model sum types, we would need also coproducts to describe case analysis. On the logical side, we have just considered standard intuitionistic connectives but, e.g., to support more quantitative analysis, one may need to use fibrations supporting linear connectives and modalities [85, 72, 67, 19, 18].

Besides the extension of our framework to richer languages and features, an interesting direction for future work is to formally relate our results with general theories of denotational logical relations. Particularly relevant for that seems the work by Katsumata [58] who observes that the closed structure of base categories is not necessarily essential. Another interesting direction for future work is the development of fibrational theories of coinductive reasoning for higher-order languages. Fibrational accounts of coinductive techniques have been given in the general setting of coalgebras [15, 14], whereas general accounts of coinductive techniques for higher-order languages have been obtained in terms of relational reasoning [28, 27, 37, 21, 20, 29, 24, 38, 26]. It would be interesting to see whether these two lines of research could be joined in our fibrational framework. That may also be a promising path to the development of conductive differential reasoning.

References

1 Andreas Abel and Jean-Philippe Bernardy. A unified view of modalities in type systems. *Proc. ACM Program. Lang.*, 4(ICFP):90:1–90:28, 2020. doi:10.1145/3408972.

2 S. Abramsky and A. Jung. Domain theory. In *Handbook of Logic in Computer Science*, pages 1–168. Clarendon Press, 1994.

3 Amal J. Ahmed. Step-indexed syntactic logical relations for recursive and quantified types. In *Proc. of ESOP 2006*, pages 69–83, 2006. doi:10.1007/11693024_6.

4 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.

5 A.W. Appel and D.A. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Trans. Program. Lang. Syst.*, 23(5):657–683, 2001. doi:10.1145/504709.504712.

6 Robert J. Aumann. Borel structures for function spaces. *Illinois Journal of Mathematics*, 5(4):614–630, 1961.

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

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

9 Jean Bénabou. Introduction to bicategories. In *Reports of the Midwest Category Seminar*, pages 1–77. Springer, 1967. doi:10.1007/BFb0074299.

10 Nick Benton, Martin Hofmann, and Vivek Nigam. Abstract effects and proof-relevant logical relations. In *Proc. of POPL 2014*, pages 619–632, 2014.

11 Dariusz Biernacki, Maciej Piróg, Piotr Polesiuk, and Filip Sieczkowski. Handle with care: relational interpretation of algebraic effects and handlers. *PACMPL*, 2(POPL):8:1–8:30, 2018.

12 Lars Birkedal, Guilhem Jaber, Filip Sieczkowski, and Jacob Thamsborg. A kripke logical relation for effect-based program transformations. *Inf. Comput.*, 249:160–189, 2016.

13 A. Bizjak and L. Birkedal. Step-indexed logical relations for probability. In *Proc. of FOSSACS 2015*, pages 279–294, 2015. doi:10.1007/978-3-662-46678-0_18.

14 Filippo Bonchi, Barbara König, and Daniela Petrisan. Up-to techniques for behavioural metrics via fibrations. In *Proc. of CONCUR 2018*, pages 17:1–17:17, 2018.

15 Filippo Bonchi, Daniela Petrisan, Damien Pous, and Jurriaan Rot. Coinduction up-to in a fibrational setting. In *Proc. of CSL-LICS '14*, pages 20:1–20:9, 2014.

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

17 Karl Crary and Robert Harper. Syntactic logical relations for polymorphic and recursive types. *Electr. Notes Theor. Comput. Sci.*, 172:259–299, 2007.

18 Francesco Dagnino and Fabio Pasquali. Logical foundations of quantitative equality. In *Proceedings of the 37th ACM/IEEE Symposium on Logic in Computer Science, LICS 2022*. ACM, 2022. to appear. doi:10.1145/3531130.3533337.

19 Francesco Dagnino and Giuseppe Rosolini. Doctrines, modalities and comonads. *Mathematical Structures in Computer Science*, pages 1–30, 2021. doi:10.1017/S0960129521000207.

20 Ugo Dal Lago and Francesco Gavazzo. Effectful normal form bisimulation. In *Proc. of ESOP 2019*, pages 263–292, 2019. doi:10.1007/978-3-030-17184-1_10.

21 Ugo Dal Lago and Francesco Gavazzo. On bisimilarity in lambda calculi with continuous probabilistic choice. In *Proc. of MFPS 2019*, pages 121–141, 2019. doi:10.1016/j.entcs.2019.09.007.

22 Ugo Dal Lago and Francesco Gavazzo. Differential logical relations part II: increments and derivatives. In Gennaro Cordasco, Luisa Gargano, and Adele A. Rescigno, editors, *Proc. of ICTCS 2020*, volume 2756 of *CEUR Workshop Proceedings*, pages 101–114, 2020.

23 Ugo Dal Lago and Francesco Gavazzo. Differential logical relations, part II increments and derivatives. *Theor. Comput. Sci.*, 895:34–47, 2021.

24 Ugo Dal Lago and Francesco Gavazzo. Resource transition systems and full abstraction for linear higher-order effectful programs. In *Proc. of FSCD 2021*, volume 195 of *LIPIcs*, pages 23:1–23:19, 2021. doi:10.4230/LIPIcs.FSCD.2021.23.

25 Ugo Dal Lago and Francesco Gavazzo. Effectful program distancing. *Proc. ACM Program. Lang.*, 6(POPL):1–30, 2022.

26 Ugo Dal Lago and Francesco Gavazzo. A relational theory of effects and coeffects. *Proc. ACM Program. Lang.*, 6(POPL):1–28, 2022.

27 Ugo Dal Lago, Francesco Gavazzo, and Paul Blain Levy. Effectful applicative bisimilarity: Monads, relators, and howe's method. In *Proc. of LICS 2017*, pages 1–12, 2017. doi:10.1109/LICS.2017.8005117.

28 Ugo Dal Lago, Francesco Gavazzo, and Ryo Tanaka. Effectful applicative similarity for call-by-name lambda calculi. In *Proc. of ICTCS 2017*, pages 87–98, 2017.

29 Ugo Dal Lago, Francesco Gavazzo, and Ryo Tanaka. Effectful applicative similarity for call-by-name lambda calculi. *Theor. Comput. Sci.*, 813:234–247, 2020. doi:10.1016/j.tcs.2019.12.025.

30 Ugo Dal Lago, Francesco Gavazzo, and Akira Yoshimizu. Differential logical relations, part I: the simply-typed case. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and

Stefano Leonardi, editors, *Proc. of ICALP 2019*, volume 132 of *LIPIcs*, pages 111:1–111:14, 2019.

31 Ugo Dal Lago, Francesco Gavazzo, and Akira Yoshimizu. Differential logical relations, part I: the simply-typed case. In *Proc. of ICALP 2019*, pages 111:1–111:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.111.

32 Derek Dreyer, Amal Ahmed, and Lars Birkedal. Logical step-indexed logical relations. *Logical Methods in Computer Science*, 7(2), 2011.

33 Derek Dreyer, Georg Neis, and Lars Birkedal. The impact of higher-order state and control effects on local relational reasoning. *J. Funct. Program.*, 22(4-5):477–528, 2012.

34 Brian P. Dunphy and Uday S. Reddy. Parametric limits. In *Proc. of LICS 2004*, pages 242–251. IEEE Computer Society, 2004.

35 Thomas Ehrhard, Michele Pagani, and Christine Tasson. Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming. *Proc. of POPL 2017*, 2:1–28, 2017.

36 Thomas Ehrhard, Michele Pagani, and Christine Tasson. Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming. *PACMPL*, 2(POPL):59:1–59:28, 2018.

37 Francesco Gavazzo. Quantitative behavioural reasoning for higher-order effectful programs: Applicative distances. In *Proc. of LICS 2018*, pages 452–461, 2018. doi:10.1145/3209108.3209149.

38 Francesco Gavazzo. *Coinductive Equivalences and Metrics for Higher-order Languages with Algebraic Effects*. PhD thesis, University of Bologna, Italy, 2019. URL: http://amsdottorato.unibo.it/9075/.

39 Neil Ghani, Patricia Johann, Fredrik Nordvall Forsberg, Federico Orsanigo, and Tim Revell. Bifibrational functorial semantics of parametric polymorphism. In Dan R. Ghica, editor, *Proc. of MFPS 2015*, volume 319 of *Electronic Notes in Theoretical Computer Science*, pages 165–181. Elsevier, 2015.

40 Neil Ghani, Patricia Johann, and Clément Fumex. Fibrational induction rules for initial algebras. In Anuj Dawar and Helmut Veith, editors, *Proc. of CSL 2010*, volume 6247 of *Lecture Notes in Computer Science*, pages 336–350. Springer, 2010.

41 Neil Ghani, Patricia Johann, and Clément Fumex. Generic fibrational induction. *Log. Methods Comput. Sci.*, 8(2), 2012.

42 Neil Ghani, Patricia Johann, and Clément Fumex. Indexed induction and coinduction, fibrationally. *Log. Methods Comput. Sci.*, 9(3), 2013.

43 J.Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1989.

44 Michèle Giry. A categorical approach to probability theory. In *Categorical Aspects of Topology and Analysis*, pages 68–85. Springer Berlin Heidelberg, 1982.

45 A.D. Gordon. A tutorial on co-induction and functional programming. In *Workshops in Computing*, pages 78–95. Springer London, September 1994. doi:10.1007/978-1-4471-3573-9_6.

46 Jean Goubault-Larrecq, Slawomir Lasota, and David Nowak. Logical relations for monadic types. *Math. Struc. Comput. Sci.*, 18(6):1169–1217, 2008. doi:10.1017/S0960129508007172.

47 Marco Grandis. Weak subobjects and the epi-monic completion of a category. *Journal of Pure and Applied Algebra*, 154(1-3):193–212, 2000.

48 Robert Harper. *Practical Foundations for Programming Languages (2nd. Ed.)*. Cambridge University Press, 2016.

49 Claudio Hermida. *Fibrations, logical predicates and indeterminates*. PhD thesis, University of Edinburgh, UK, 1993.

50 Claudio Hermida and Bart Jacobs. Structural induction and coinduction in a fibrational setting. *Inf. Comput.*, 145(2):107–152, 1998.

51 Claudio Hermida, Uday S. Reddy, and Edmund P. Robinson. Logical relations and parametricity - A reynolds programme for category theory and programming languages. *Electronic Notes on Theoretical Computer Science*, 303:149–180, 2014. doi:10.1016/j.entcs.2014.02.008.

52 Martin Hofmann. Logical relations and nondeterminism. In *Software, Services, and Systems - Essays Dedicated to Martin Wirsing on the Occasion of His Retirement from the Chair of Programming and Software Engineering*, pages 62–74, 2015.

53 Pieter Hofstra. The dialectica monad and its cousins. In *Models, Logics, and Higher-dimensional Categories: A Tribute to the Work of Mihaly Makkai*, number 53 in CRM proceedings & lecture notes, pages 107–139, 2011.

54 Jesse Hughes and Bart Jacobs. Factorization systems and fibrations: Toward a fibred birkhoff variety theorem. In Richard Blute and Peter Selinger, editors, *CTCS 2002*, volume 69 of *Electronic Notes in Theoretical Computer Science*, pages 156–182. Elsevier, 2002.

55 Bart P. F. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in logic and the foundations of mathematics*. North-Holland, 2001. URL: http://www.elsevierdirect.com/product.jsp?isbn=9780444508539.

56 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. doi:10.1109/LICS.2010.29.

57 Ohad Kammar and Dylan McDermott. Factorisation systems for logical relations and monadic lifting in type-and-effect system semantics. In Sam Staton, editor, *Proc. of MFPS 2018*, volume 341 of *Electronic Notes in Theoretical Computer Science*, pages 239–260. Elsevier, 2018.

58 Shin-ya Katsumata. *A generalisation of pre-logical predicates and its applications*. PhD thesis, University of Edinburgh, UK, 2005.

59 Shin-ya Katsumata. A semantic formulation of tt-lifting and logical predicates for computational metalanguage. In C.-H. Luke Ong, editor, *Proc. of CSL 2005*, volume 3634 of *Lecture Notes in Computer Science*, pages 87–102. Springer, 2005.

60 Shin-ya Katsumata. Relating computational effects by $\top\top$-lifting. *Inf. Comput.*, 222:228–246, 2013.

61 Shin-ya Katsumata. Parametric effect monads and semantics of effect systems. In *Proc. of POPL 2014*, pages 633–646, 2014.

62 Shin-ya Katsumata, Tetsuya Sato, and Tarmo Uustalu. Codensity lifting of monads and its dual. *Logical Methods in Computer Science*, 14(4), 2018.

63 G. A. Kavvos. Modalities, cohesion, and information flow. *Proc. ACM Program. Lang.*, 3(POPL):20:1–20:29, 2019.

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

65 Daan Leijen. Implementing algebraic effects in c. In Bor-Yuh Evan Chang, editor, *Programming Languages and Systems*, pages 339–363, Cham, 2017. Springer International Publishing.

66 P.B. Levy, J. Power, and H. Thielecke. Modelling environments in call-by-value programming languages. *Inf. Comput.*, 185(2):182–210, 2003. doi:10.1016/S0890-5401(03)00088-9.

67 Daniel R. Licata, Michael Shulman, and Mitchell Riley. A fibrational framework for substructural and modal logics. In Dale Miller, editor, *2nd International Conference on Formal Structures for Computation and Deduction, FSCD 2017*, volume 84 of *LIPIcs*, pages 25:1–25:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.FSCD.2017.25.

68 QingMing Ma and John C. Reynolds. Types, abstractions, and parametric polymorphism, part 2. In *Proc. of MFPS 1991*, volume 598 of *Lecture Notes in Computer Science*, pages 1–40. Springer, 1991.

69 Maria Emilia Maietti, Fabio Pasquali, and Giuseppe Rosolini. Quasi-toposes as elementary quotient completions, 2021. arXiv:arXiv:2111.15299.

70 Maria Emilia Maietti and Giuseppe Rosolini. Elementary quotient completion. *Theory and Application of Categories*, 27(17):445–463, 2013.

71 Ernest G. Manes and Michael A. Arbib. *Algebraic Approaches to Program Ssemantics*. Texts and Monographs in Computer Science. Springer, 1986.

72 Paul-André Melliès and Noam Zeilberger. Functors are type refinement systems. In Sriram K. Rajamani and David Walker, editors, *Proceedings of the 42nd Annual ACM Symposium on Principles of Programming Languages, POPL 2015*, pages 3–16. ACM, 2015. doi:10.1145/2676726.2676970.

73 Paul-André Melliès and Noam Zeilberger. A bifibrational reconstruction of lawvere's presheaf hyperdoctrine. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16*, pages 555–564. ACM, 2016. doi:10.1145/2933575.2934525.

74 John C. Mitchell. *Foundations for programming languages*. Foundation of computing series. MIT Press, 1996.

75 John C. Mitchell and Andre Scedrov. Notes on sconing and relators. In *Proc. of CSL '92*, volume 702 of *Lecture Notes in Computer Science*, pages 352–378. Springer, 1992.

76 Peter W. O'Hearn and Robert D. Tennent. Parametricity and local variables. *J. ACM*, 42(3):658–709, 1995.

77 Benjamin C. Pierce. *Types and programming languages*. MIT Press, 2002.

78 Paolo Pistone. On generalized metric spaces for the simply typed lambda-calculus. In *Proc. of LICS 2021*, pages 1–14, 2021.

79 Gordon Plotkin. Lambda-definability and logical relations. Technical Report SAI-RM-4, School of A.I., University of Edinburgh, 1973.

80 Gordon D. Plotkin and John Power. Adequacy for algebraic effects. In *Proc. of FOSSACS 2001*, pages 1–24, 2001. doi:10.1007/3-540-45315-6_1.

81 Gordon D. Plotkin and John Power. Algebraic operations and generic effects. *Applied Categorical Structures*, 11(1):69–94, 2003. doi:10.1023/A:1023064908962.

82 J. Reed and B.C. Pierce. Distance makes the types grow stronger: a calculus for differential privacy. In *Proc. of ICFP 2010*, pages 157–168, 2010. doi:10.1145/1863543.1863568.

83 J.C. Reynolds. Types, abstraction and parametric polymorphism. In *IFIP Congress*, pages 513–523, 1983.

84 Edmund P. Robinson and Giuseppe Rosolini. Reflexive graphs and parametric polymorphism. In *Proc. of LICS '94*, pages 364–371. IEEE Computer Society, 1994.

85 Michael A. Shulman. Framed bicategories and monoidal fibrations. *Theory and Applications of Categories*, (18):650–738, 2008.

86 Kristina Sojakova and Patricia Johann. A general framework for relational parametricity. In Anuj Dawar and Erich Grädel, editors, *Proce. of LICS 2018,*, pages 869–878. ACM, 2018.

87 Richard Statman. Logical relations and the typed lambda-calculus. *Inf. Control.*, 65(2/3):85–97, 1985.

88 Sam Staton. Commutative semantics for probabilistic programming. In *Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, pages 855–879, 2017.

89 Sam Staton, Hongseok Yang, Frank D. Wood, Chris Heunen, and Ohad Kammar. Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 525–534, 2016.

90 Ross Street. The formal theory of monads. *Journal of Pure and Applied Algebra*, 2(2):149–168, 1972. doi:10.1016/0022-4049(72)90019-9.

91 Ross Street. Two constructions on lax functors. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 13(3):217–264, 1972.

92 Thomas Streicher. Fibered categories a la jean benabou, 2018. arXiv:arXiv:1801.02927.

93 William W. Tait. Intensional interpretations of functionals of finite type I. *J. Symb. Log.*, 32(2):198–212, 1967.

94 Aaron Joseph Turon, Jacob Thamsborg, Amal Ahmed, Lars Birkedal, and Derek Dreyer. Logical relations for fine-grained concurrency. In Roberto Giacobazzi and Radhia Cousot, editors, *Proc. of POPL '13*, pages 343–356. ACM, 2013.

95 Matthijs Vákár, Ohad Kammar, and Sam Staton. A domain theory for statistical probabilistic programming. *Proc. ACM Program. Lang.*, 3(POPL):36:1–36:29, 2019.

96 C. Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008.

97 Mitchell Wand, Ryan Culpepper, Theophilos Giannakopoulos, and Andrew Cobb. Contextual equivalence for a probabilistic language with continuous random variables and recursion. *PACMPL*, 2(ICFP):87:1–87:30, 2018.