Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 352 (2020) 5–27

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Weakest Preconditions in Fibrations

Alejandro Aguirre [1,2]

IMDEA Software Institute & Universidad Politécnica de Madrid
Campus de Montegancedo, 28223 Pozuelo de Alarcón, Madrid, Spain

Shin-ya Katsumata[1,3]

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyodaku, Tokyo, Japan

Abstract

Weakest precondition transformers are useful tools in program verification. One of their key properties is compositionality, that is, the weakest precondition predicate transformer (wppt for short) associated to program $f;g$ should be equal to the composition of the wppts associated to f and g. In this paper, we study the categorical structure behind wppts from a fibrational point of view. We characterize the wppts that satisfy compositionality as the ones constructed from the Cartesian lifting of a monad. We moreover show that Cartesian liftings of monads along lax slice categories bijectively correspond to Eilenberg-Moore monotone algebras. We then instantiate our techniques by deriving wppts for commonplace effects such as the maybe monad, the non-empty powerset monad, the counter monad or the distribution monad. We also show how to combine them to derive the wppts appearing in the literature of verification of probabilistic programs.

Keywords: Weakest precondition predicate transformer, Monad, Computational effects, Fibered category theory, Hoare logic.

1 Introduction

Dijkstra's *weakest precondition predicate transformer* (*wppt* for short) [6] computes, for a given imperative program f and a predicate Q, the weakest predicate P such that, for any input x satisfying P, $f(x)$ is guaranteed to satisfy Q. When the imperative program f always terminates and only updates the memory deterministically, the behavior of f can be modeled as an endofunction $[\![f]\!]$ over the set M of memory

[1] This research was supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST. This research was conducted during the first author's visit to the National Institute of Informatics. The authors are grateful to Satoshi Kura for detailed comments on this paper, and to anonymous reviewers for careful reading of our manuscript and constructive feedback.

[2] Email: alejandro.aguirre@imdea.org

[3] Email: s-katsumata@nii.ac.jp

https://doi.org/10.1016/j.entcs.2020.09.002

configurations, and the weakest predicate P is defined so that $[\![P]\!] \subseteq M$ (memory configurations satisfying P) corresponds to the *inverse image* of $[\![Q]\!] \subseteq M$ along $[\![f]\!]$. One of the key properties of predicate transformers that make them suitable for program verification is *composability*: the weakest precondition $wp(f; g, Q)$ of a composite program $f; g$ is equal to the composition of the weakest preconditions of its components, that is, $wp(f, wp(g, Q))$. This allows for predicate transformers to be defined and applied inductively over the structure of a program.

Following Dijkstra's seminal work, wppts and their variants have been applied to program verification in different manners, such as computing expected values over outputs of probabilistic programs [29], estimating runtime [16], or estimating tail bounds of rewards over control-flow graphs [21]. The motivation of this paper is to identify a mathematical structure behind these variations of wppt-like semantics. Towards this goal, in this paper we set out to study wppts and their composability in *fibrations*. Roughly speaking, a fibration is a functor $p : \mathbb{P} \to \mathbb{C}$ that exposes a relationship between a category $\mathbb{P}$ of predicates and an underlying category $\mathbb{C}$ modeling computation - when $pP = X$ holds, we regard $P \in \mathbb{P}$ as a predicate over $X \in \mathbb{C}$; this viewpoint is shared with the categorical study of *refinement types* [30]. Fibrations are especially suited to interpret predicate transformers and their composability thanks to the *Cartesian lifting property*, which allows us to take the "inverse image" of $Q \in \mathbb{P}$ along a morphism $f : X \to pQ$ in $\mathbb{C}$, resulting an object $f^*Q \in \mathbb{P}$ such that $p(f^*Q) = X$. This is a categorical abstraction of the inverse image operation.

We now consider extending the imperative programming language (whose programs are modeled as functions of type $M \to M$) and its Hoare logic with side effects.[4] To give the denotational semantics to both systems in the monadic style [32], we employ a pair of monads $\mathcal{T}$ on $\mathbb{C}$ and $\dot{\mathcal{T}}$ on $\mathbb{P}$ such that p strictly preserve the monad structure - the latter is called a *lifting* of $\mathcal{T}$ along p. We then regard morphisms in $\mathbb{P}_{\dot{\mathcal{T}}}$ as interpretations of Hoare triples. This induces a natural wppt that characterizes the Hoare triple, but it does *not* satisfy the composability in general. This raises the question of when monadic computations do induce compositional predicate transformers. In this paper, we answer to this question by introducing the property called *Cartesian-ness* on monad liftings, and contribute to its understanding as follows:

(i) We show that the wppt defined using a monad lifting is composable if and only if the monad lifting is Cartesian. This result establishes the tight connection between the composability of wppts in monadic setting and the Cartesian-ness of liftings.

(ii) We study Cartesian liftings of monads along domain fibrations from lax slice categories. For this class of fibrations, there is a bijective correspondence between Cartesian liftings of a monad $\mathcal{T}$ and Eilenberg-Moore monotone algebras

[4] Here side effects refer to those that can *not* be modeled by memory update functions of type $M \to M$, such as input, output, nondeterministic choice, probabilistic choice, and manipulating *an external memory device*. The last one may be modeled by the state monad $S \Rightarrow (- \times S)$ employing the set S of states taken independently from M.

of $\mathcal{T}$; this is exhibited through the monoidal isomorphism between the category of algebras (of varying endofunctors) over the slicing object and the category of fibered functors between domain fibrations. This result simplifies the task of exploring Cartesian liftings of monads along such fibrations. We also relate *strongest postcondition predicate transformers* as left adjoints to wppts, and discuss when they are available in domain fibrations.

(iii) Computational effects are often modeled by composite monads via *distributive laws*. We extend the correspondence given in (ii) to Cartesian liftings of composite monads and pairs of Eilenberg-Moore monotone algebras satisfying an extra coherence law given in [4,27]. This correspondence provides a modular method to compute Cartesian liftings of composite monads.

(iv) To compute the wppts of the programs containing effectful commands (such as probabilistic choice and counting), we study the interaction between Plotkin and Power's *algebraic operations* [34], which are a categorical models of effectful commands, and wppts studied in (ii).

(v) Apart from domain fibrations, we illustrate a few examples of Cartesian liftings of monads along relational fibrations. They are outside of the framework presented in [11] and [13].

These studies allow us to define liftings of monads (and therefore, wppts) from their algebras, as well as defining transformers for composite monads by composing individual algebras. We illustrate this by defining predicate transformers for monads in the literature: the maybe monad, the non-empty powerset monad, the counting monad, the distribution monad and the indexed distribution monad. Then we show how to combine these to recover predicate transformers in the literature, such as the expected runtime transformer by Kaminski et al. [16] and the higher-order moment transformer by Kura et al. [21].

2 Preliminaries

Composition of functors, and whiskering [42] of functors and natural transformations are denoted by juxtaposition. The vertical and horizontal composition of natural transformations are denoted by $\star$ and $*$ respectively.

We omit the definitions of monads, Eilenberg-Moore (EM for short) algebras of monads and Kleisli categories of monads; see e.g. [24, Section VI]. Let $(\mathcal{T}, \eta^{\mathcal{T}}, \mu^{\mathcal{T}})$ be a monad on a category $\mathbb{C}$. The *Kleisli lifting* of $f : X \to \mathcal{T}Y$ is $f^{\#_{\mathcal{T}}} \triangleq \mu^{\mathcal{T}}_Y \circ \mathcal{T}f$. When $\mathcal{T}$ is obvious from context, we simply write $f^{\#}$ to mean $f^{\#_{\mathcal{T}}}$. The Kleisli category of $\mathcal{T}$ is denoted by $\mathbb{C}_{\mathcal{T}}$, and composition in it is denoted by $\bullet$. The adjunction between $\mathbb{C}$ and $\mathbb{C}_{\mathcal{T}}$ is called the *Kleisli resolution* of $\mathcal{T}$. Following [32], we regard morphisms in $\mathbb{C}_{\mathcal{T}}$ as abstract representations of programs causing computational effects.

Let $(\mathcal{S}, \eta^{\mathcal{S}}, \mu^{\mathcal{S}})$ be another monad on $\mathbb{C}$. A *distributive law* [4] of $\mathcal{S}$ over $\mathcal{T}$ is a natural transformation $\alpha : \mathcal{S}\mathcal{T} \to \mathcal{T}\mathcal{S}$ such that

$$\eta^{\mathcal{T}}\mathcal{S} = \alpha \star \mathcal{S}\eta^{\mathcal{T}} \qquad \alpha \star \mathcal{S}\mu^{\mathcal{T}} = \mu^{\mathcal{T}}\mathcal{S} \star \mathcal{T}\alpha \star \alpha\mathcal{T} \qquad \mathcal{T}\eta^{\mathcal{S}} = \alpha \star \eta^{\mathcal{S}}\mathcal{T} \qquad \alpha \star \mu^{\mathcal{S}}\mathcal{T} = \mathcal{T}\mu^{\mathcal{S}} \star \alpha\mathcal{S} \star \mathcal{S}\alpha$$

The distributive law yields the monad $(\mathcal{TS}, \eta^{\mathcal{T}} * \eta^{\mathcal{S}}, (\mu^{\mathcal{T}} * \mu^{\mathcal{S}}) \star (\mathcal{T}\alpha\mathcal{S}))$ over the composite functor. We denote this monad by $\mathcal{T} \circ_\alpha \mathcal{S}$.

2.1 Fibrations

Dijkstra's weakest preconditions manipulate *predicates* over memory configurations. In the abstract study of predicate transformers, however, it is convenient to extend the concept of predicates so that they can be defined over arbitrary objects on a category. For this abstract and general treatment of predicates, we leverage the notions of *fibered category theory* (see e.g. [14]). Before introducing these concepts, we give an informal account of them.

Given a category $\mathbb{C}$ (whose objects we denote with letters X, Y, Z) we aim to define predicates over its objects by introducing a category $\mathbb{P}$ (whose objects we denote with P, Q, R) and a functor $p : \mathbb{P} \to \mathbb{C}$. We understand this general situation as follows:

- An object in $\mathbb{P}$ is a predicate above some object in $\mathbb{C}$, which is recorded by the functor $p : \mathbb{P} \to \mathbb{C}$. That is, $pP = X$ means that $P \in \mathbb{P}$ is a predicate over $X \in \mathbb{C}$.
- A morphism $f : P \to Q$ in $\mathbb{P}$ is a *witness* of the fact that the underlying morphism $pf : pP \to pQ$ in $\mathbb{C}$ "respects" these predicates, that is, pf maps elements satisfying the predicate P to those satisfying the predicate Q. In particular, if $f : P \to Q$ satisfies $pf = \mathrm{id}_X$, then f witnesses that P implies Q.

Inverse images under the functor $p : \mathbb{P} \to \mathbb{C}$ for an object $X \in \mathbb{C}$ form a category denoted by $\mathbb{P}_X$, known as the *fiber category* above X. Formally, an object of $\mathbb{P}_X$ is a $\mathbb{P}$-object $P \in \mathbb{P}$ such that $pP = X$, and a $\mathbb{P}_X$-morphism from P to Q is a $\mathbb{P}$-morphism $f : P \to Q$ such that $pf = \mathrm{id}_X$. Intuitively, objects of this category are predicates over X and morphisms witness the implications between them.

Underlying weakest precondition predicate transformers, the strength of predicates are compared by an order relation, so we focus our attention on *posetal fibrations*. Recall that the weakest precondition predicate transformer collects *all* the memory configurations that entail the postcondition. Set-theoretically, this operation is called *inverse image*, and *fibrations* offers a more general and flexible treatment of inverse images. Roughly speaking, a fibration is a functor $p : \mathbb{P} \to \mathbb{C}$ such that for any $f \in \mathbb{C}(X, Y)$ and $P \in \mathbb{P}_Y$, we can find the *inverse image* $f^*P \in \mathbb{P}_X$ of P along f. Formal definition of posetal fibration follows.

- For objects $P, Q \in \mathbb{P}$ and a morphism $f \in \mathbb{C}(pP, pQ)$, we define the set $\mathbb{P}_f(P, Q)$ of $\mathbb{P}$-morphisms above f by $\mathbb{P}_f(P, Q) = \{k \in \mathbb{P}(P, Q) \mid pk = f\}$.
- A morphism $k \in \mathbb{P}(P, Q)$ is *Cartesian* if for any $R \in \mathbb{P}$ and $h \in \mathbb{C}(pR, pP)$, the postcomposition of k, regarded as a function of type $\mathbb{P}_h(R, P) \to \mathbb{P}_{pk \circ h}(R, Q)$, is a bijection. This is the universal property of Cartesian morphism. A Cartesian morphism $k : P \to Q$ abstractly represents the situation that P is an inverse image of Q along pk.
- A functor $p : \mathbb{P} \to \mathbb{C}$ is a *fibration* if for any $f \in \mathbb{C}(X, Y)$ and $Q \in \mathbb{P}_Y$, there is an object $P \in \mathbb{P}_X$ and a Cartesian morphism $k \in \mathbb{P}_f(P, Q)$ called the *cartesian*

lifting of f with Q.

- A fibration $p : \mathbb{P} \to \mathbb{C}$ is *posetal* if each fiber category $\mathbb{P}_X$ is a poset. In any posetal fibration the Cartesian lifting of $f \in \mathbb{C}(X, Y)$ with $Q \in \mathbb{P}_Y$ uniquely exists; we therefore denote it by $\overline{f}Q$ and its domain by f^*Q. We note that posetal fibrations are faithful. From these, a morphism $f : P \to Q$ in $\mathbb{P}$ is Cartesian if and only if $P = (pf)^*Q$.
- For $f \in \mathbb{C}(X, Y)$, the assignment $Q \in \mathbb{P}_Y \mapsto f^*Q \in \mathbb{P}_P$ extends to a functor of type $\mathbb{P}_Y \to \mathbb{P}_X$, which we call the *reindexing functor*. The assignment $f \mapsto f^*$ furthermore satisfies $(\mathrm{id}_X)^* = \mathrm{id}_{\mathbb{P}_X}$ and $(g \circ f)^* = f^* \circ g^*$.

A *fibered functor* between posetal fibrations $p : \mathbb{P} \to \mathbb{C}$ and $q : \mathbb{Q} \to \mathbb{D}$ is a pair of functors $F : \mathbb{C} \to \mathbb{D}$ and $\dot{F} : \mathbb{P} \to \mathbb{Q}$ such that $q \circ \dot{F} = F \circ p$, and $\dot{F}$ preserves Cartesian morphisms. This is equivalent to: for any object $P \in \mathbb{P}$ and morphism $f : X \to pP$ in $\mathbb{C}$, $\dot{F}(f^*P) = (Ff)^*(\dot{F}P)$. Given two fibered functors $(F, \dot{F}), (G, \dot{G})$ from $p : \mathbb{P} \to \mathbb{C}$ to $q : \mathbb{Q} \to \mathbb{D}$, a *2-cell* from the former to the latter is a pair of natural transformations $\alpha : F \to G$ and $\dot{\alpha} : \dot{F} \to \dot{G}$ such that $q\dot{\alpha} = \alpha p$. The 2-category determined by these data is denoted by **Pos-Fib**. We say that a 2-cell $(\alpha, \dot{\alpha})$ is *Cartesian* if $\dot{F}P = \alpha_{pP}^*(\dot{G}P)$. The subcategory of **Pos-Fib**(p, q) whose 2-cells are restricted to Cartesian ones is denoted by **Pos-Fib**$_c(p, q)$.

Further restriction of posetal fibrations is possible. Let $\mathbb{K}$ be a subcategory of the category **Pos** of posets and monotone functions. A $\mathbb{K}$-fibration $p : \mathbb{P} \to \mathbb{C}$ is a posetal fibration such that each fiber category $\mathbb{P}_X$ and each reindexing functor f^* belong to $\mathbb{K}$. A $\mathbb{K}$-fibered functor $(F, \dot{F})$ from a $\mathbb{K}$-fibration $p : \mathbb{P} \to \mathbb{C}$ to another one $q : \mathbb{Q} \to \mathbb{D}$ is a fibered functor from p to q such that the restriction of $\dot{F}$ to each fiber $\mathbb{P}_X$ becomes a morphism of type $\mathbb{P}_X \to \mathbb{Q}_{FX}$ in $\mathbb{K}$. We write $\mathbb{K}$-**Fib** for the sub-2-category of **Pos-Fib** (resp. **Pos-Fib**$_c$) where 0-cells are $\mathbb{K}$-fibrations and 1-cells are $\mathbb{K}$-fibered functors. 2-cells remain the same.

3 Dijkstra Structures and Weakest Precondition Predicate Transformers

In the previous section, we have seen how to define a category of predicates $\mathbb{P}$ over a category $\mathbb{C}$. We next add computational effects modeled by some monad $\mathcal{T}$ to this situation. We aim to model the statement that a monadic computation $f : X \to \mathcal{T}Y$ respects pre- and post-conditions, given as predicates over X and Y respectively. For this purpose, the first step is defining a lifting of $\mathcal{T}$ into a monad $\dot{\mathcal{T}}$ over $\mathbb{P}$.

Definition 3.1 Let $p : \mathbb{P} \to \mathbb{C}$ be a posetal fibration and $(\mathcal{T}, \eta, \mu)$ be a monad on $\mathbb{C}$. A monad $(\dot{\mathcal{T}}, \dot{\eta}, \dot{\mu})$ on $\mathbb{P}$ satisfying $p\dot{\mathcal{T}} = \mathcal{T}p, p\dot{\eta} = \eta p$ and $p\dot{\mu} = \mu p$ is called a *lifting* of $\mathcal{T}$ (along p). We say that the lifting is:

- *fibered* if $(\mathcal{T}, \dot{\mathcal{T}})$ is a fibered functor from p to p, and
- *Cartesian* if it is fibered and $\dot{\mu}$ and $\dot{\eta}$ are Cartesian, that is, $P = \eta_{pP}^*(\dot{\mathcal{T}}P)$ and $\dot{\mathcal{T}}\dot{\mathcal{T}}P = \mu_{pP}^*\dot{\mathcal{T}}P$.

The concept of monad lifting is not new; it appeared as a semantic counterpart

of *logical relations for monads* [7,8,10,17,18]; Hermida considered the comonadic case earlier than these works [12, Chapter 5]. The definition of Cartesian lifting of monads makes sense when p is a non-posetal fibration. When $\mathbb{C}$ is a category with pullbacks, a monad $\mathcal{T}$ on $\mathbb{C}$ is Cartesian (see e.g. [23, Section 4.1]) if and only if the evident lifting $\mathcal{T}^{\to} : \mathbb{C}^{\to} \to \mathbb{C}^{\to}$ of $\mathcal{T}$ to the arrow category $\mathbb{C}^{\to}$ along the codomain fibration $\mathrm{cod} : \mathbb{C}^{\to} \to \mathbb{C}$ is Cartesian.

The tuple consisting of a posetal fibration p, a monad $\mathcal{T}$ and its lifting $\dot{\mathcal{T}}$ along p provides us a setting where we define an abstract notion of Hoare triple and the wppt associated to it.

Definition 3.2 [Dijkstra Structure] A (resp. fibered, Cartesian) *Dijkstra structure* is a tuple $(p, \mathcal{T}, \dot{\mathcal{T}})$ of a posetal fibration $p : \mathbb{P} \to \mathbb{C}$, a monad $\mathcal{T}$ on $\mathbb{C}$ and a (resp. fibered, Cartesian) lifting $\dot{\mathcal{T}}$ of $\mathcal{T}$ along p.

Definition 3.3 Let $(p : \mathbb{P} \to \mathbb{C}, \mathcal{T}, \dot{\mathcal{T}})$ be a Dijkstra structure. Below X, Y range over $\mathbb{C}$-objects.

(i) For any $f : X \to \mathcal{T}Y$ in $\mathbb{C}$ and $P \in \mathbb{P}_X, Q \in \mathbb{P}_Y$, we define the *Hoare triple* $P\{f\}Q$ by

$$P\{f\}Q \overset{\triangle}{\iff} \exists \dot{f} \in \mathbb{P}(P, \dot{\mathcal{T}}Q) \,.\, p\dot{f} = f. \tag{1}$$

Such $\dot{f}$ is unique because p is faithful.

(ii) We define the *weakest precondition predicate transformer* (*wppt* for short) $\mathrm{wp} : \mathbb{C}(X, \mathcal{T}Y) \times \mathbb{P}_Y \to \mathbb{P}_X$ by

$$\mathrm{wp}(f, Q) \overset{\triangle}{=} f^*(\dot{\mathcal{T}}Q). \tag{2}$$

These two concepts are linked by the equivalence $P \leq \mathrm{wp}(f, Q) \iff P\{f\}Q$. The Hoare triple and the wppt in our categorical setting are more general than the standard ones since we can supply any $\mathbb{P}$-object and $\mathbb{C}$-morphism to the wppt. This liberation allows us to relate the *composability* of the wppt and the *Cartesian-ness* of $\dot{\mathcal{T}}$. Since wp takes a Kleisli morphism as an argument, the composability should be discussed with respect to the Kleisli composition.

Theorem 3.4 *Let $(p, \mathcal{T}, \dot{\mathcal{T}})$ be a Dijkstra structure. We have inequalities $\mathrm{wp}(\eta_{pP}, P) \geq P$ and $\mathrm{wp}(f \bullet g, P) \geq \mathrm{wp}(g, \mathrm{wp}(f, P))$ for any f, g, P of appropriate type. Moreover, they become equalities if and only if the Dijkstra structure is Cartesian.*

Proof. The unit $\dot{\eta}_P : P \to \dot{\mathcal{T}}P$ of $\dot{\mathcal{T}}$ is above η_{pP}. Therefore

$$\mathrm{wp}(\eta_{pP}, P) = \eta_{pP}^*(\dot{\mathcal{T}}P) \geq P.$$

If $\dot{\eta}_P$ is Cartesian, this inequality becomes an equality. Next, the multiplication $\dot{\mu}_P : \dot{\mathcal{T}}\dot{\mathcal{T}}P \to \dot{\mathcal{T}}P$ of $\dot{\mathcal{T}}$ is above μ_{pP}. Therefore $\dot{\mathcal{T}}\dot{\mathcal{T}}P \leq \mu_{pP}^*\dot{\mathcal{T}}P$; this becomes an equality if $\dot{\mu}$ is Cartesian. Moreover, for any $X \in \mathbb{C}, P \in \mathbb{P}$ and $f : X \to pP$, we have $\dot{\mathcal{T}}(f^*P) \leq (\mathcal{T}f)^*(\dot{\mathcal{T}}P)$; this becomes an equality if $\dot{\mathcal{T}}$ if fibered. From these,

we obtain the following inequality, which becomes an equality if $\dot{\mu}$ is Cartesian and $\dot{\mathcal{T}}$ is fibered.

$$\begin{aligned}(f \bullet g)^*(\dot{\mathcal{T}}P) &= (\mu \circ \mathcal{T}f \circ g)^*(\dot{\mathcal{T}}P) = g^*((\mathcal{T}f)^*(\mu^*(\dot{\mathcal{T}}P))) \\ &\geq g^*((\mathcal{T}f)^*(\dot{\mathcal{T}}\dot{\mathcal{T}}P)) \geq g^*(\dot{\mathcal{T}}(f^*(\dot{\mathcal{T}}P)) = \mathrm{wp}(g, \mathrm{wp}(f, P)).\end{aligned}$$

We have thus proved the inequalities about wp for a general Dijkstra Structure $(p, \mathcal{T}, \dot{\mathcal{T}})$, and if it is Cartesian, these inequalities become equalities.

Conversely, let $(p, \mathcal{T}, \dot{\mathcal{T}})$ be a Dijkstra Structure, and assume

$$P = \mathrm{wp}(\eta_{pP}, P) = \eta_{pP}^*(\dot{\mathcal{T}}P), \tag{3}$$

$$g^*\dot{\mathcal{T}}(f^*\dot{\mathcal{T}}P) = \mathrm{wp}(g, \mathrm{wp}(f, P)) = \mathrm{wp}(f \bullet g, P) = f^*(Tg)^*\mu_{pP}^*\dot{\mathcal{T}}P. \tag{4}$$

Below let $X = pP$. (3) implies that $\dot{\eta}_P$ is Cartesian. By letting $g = \mathrm{id}_{TTP}$ and $f = \mathrm{id}_{TP}$ in (4), we obtain $\dot{\mathcal{T}}\dot{\mathcal{T}}P = \mu_P^*(\dot{\mathcal{T}}X)$, hence $\dot{\mu}_P$ is Cartesian. By letting $g = \mathrm{id}_{TX}$ and $f = \eta_X \circ h : Y \to X \to TX$ for some $h : Y \to X$ in (4), we obtain $\dot{\mathcal{T}}(h^*P) = \dot{\mathcal{T}}(h^*\eta_X^*\dot{\mathcal{T}}P) = (\mathcal{T}h)^*(\dot{\mathcal{T}}P)$, hence $\dot{\mathcal{T}}$ is fibered. □

Therefore in a Cartesian Dijkstra structure $(p : \mathbb{P} \to \mathbb{C}, \mathcal{T}, \dot{\mathcal{T}})$, the wppt becomes a functor of type $(\mathbb{C}_{\mathcal{T}})^{op} \to \mathbf{Pos}$. The corresponding fibration (by Grothendieck construction) coincides with the extension of p to Kleisli categories:

Corollary 3.5 *Let $(p, \mathcal{T}, \dot{\mathcal{T}})$ be a Cartesian Dijkstra Structure. We take Kleisli resolutions $L \dashv R : \mathbb{C}_{\mathcal{T}} \to \mathbb{C}$ of $\mathcal{T}$ and $\dot{L} \dashv \dot{R} : \mathbb{P}_{\dot{\mathcal{T}}} \to \mathbb{P}$ of $\dot{\mathcal{T}}$. We define the functor $p_{\mathcal{T},\dot{\mathcal{T}}} : \mathbb{P}_{\dot{\mathcal{T}}} \to \mathbb{C}_{\mathcal{T}}$ by $p_{\mathcal{T},\dot{\mathcal{T}}}P = pP$ and $p_{\mathcal{T},\dot{\mathcal{T}}}f = pf$. Then*

(i) *$p_{\mathcal{T},\dot{\mathcal{T}}}$ is a posetal fibration whose pullback coincides with the wppt: $f^*P = wp(f, P)$.*

(ii) *For each $X \in \mathbb{C}$, we have the isomorphism between fiber categories: $(\mathbb{P}_{\dot{\mathcal{T}}})_{LX} \cong \mathbb{P}_X$.*

(iii) *$(L, \dot{L})$ and $(R, \dot{R})$ form a fibered adjunction (see [14, Exercise 1.8.10]) between p and $p_{\mathcal{T},\dot{\mathcal{T}}}$.*

A recent work [2] presents the failure of the composability of the relational pre-expectation operator (rpeo for short). We associate their problem and our setting. Their rpeo coincides with our categorical wppt in the Dijkstra structure $(p, \mathcal{D}, \mathcal{K})$, where p : **EPMet** $\to$ **Set** is the forgetful functor from the category of extended pseudometric spaces (which is a partial order fibration), $\mathcal{D}$ is the finite probability distribution monad (Example 5.6), and $\mathcal{K}$ is the Kantorovich metric construction, known as a lifting of the finite probability distribution monad $\mathcal{D}$ along p. Since it fails to satisfy composability, we conclude that $\mathcal{K}$ is *not* Cartesian.

4 Dijkstra Structures on Lax Slice Categories

In the setting of an arbitrary fibration, we are unaware of any general way of constructing Cartesian liftings of monads. However, in the specific case of *domain fibrations* from *lax slice categories*, there is a recipe for constructing Cartesian liftings.

In this section, we will study this recipe abstractly, and we will then instantiate it for the examples in Sections 5-7, by constructing various Cartesian liftings of monads along domain fibrations.

With the goal of having fibers with structure of posets, we consider slices over ordered objects, defined as:

Definition 4.1 An *ordered object* in a category $\mathbb{C}$ is a pair of an object $\Omega \in \mathbb{C}$ and an assignment of a partial order $\leq_X$ to the homset $\mathbb{C}(X, \Omega)$ for each $X \in \mathbb{C}$. These partial orders should satisfy: for any $h : Y \to X$, $i \leq_X i'$ implies $i \circ h \leq_Y i' \circ h$.

Example 4.2 A typical way to give an ordered object in **Set** is to take a poset $(\Omega, \leq)$ and define the partial order $\leq_X$ on $\mathbf{Set}(X, \Omega)$ to be the pointwise order of functions. Examples in Section 5-7 all uses domain fibrations arising from this kind of ordered object.

From an ordered object $(\Omega, \leq)$ in $\mathbb{C}$, we define the *lax slice category* $\mathbb{C}/\Omega$. Its objects are $\mathbb{C}$-morphisms into Ω. A morphism from i to j is a morphism $h \in \mathbb{C}(X, Y)$ such that $i \leq_{\mathrm{dom}(i)} j \circ h$. The *domain functor* $d^{\mathbb{C},\Omega} : \mathbb{C}/\Omega \to \mathbb{C}$ defined by $d^{\mathbb{C},\Omega} i \triangleq \mathrm{dom}(i)$ and $d^{\mathbb{C},\Omega}(f) \triangleq f$ is a posetal fibration, and the fiber category $(\mathbb{C}/\Omega)_X$ at $X \in \mathbb{C}$ is the poset $(\mathbb{C}(X, \Omega), \leq_X)$. The pullback of $i : X \to \Omega$ along $h : Y \to X$ is given by $h^* i = i \circ h$.

For the remainder of this section, let $\mathbb{C}$ be a category and $(\Omega, \leq)$ an ordered object in $\mathbb{C}$.

We are particularly interested in fibered / Cartesian Dijkstra structures of the form $(d^{\mathbb{C},\Omega}, \mathcal{T}, \dot{\mathcal{T}})$. In fact, a pair $(\mathcal{T}, \dot{\mathcal{T}})$ making the triple a fibered / Cartesian Dijkstra structure is nothing but a *monoid object* in the strict monoidal category $(\mathbf{Pos\text{-}Fib}(d^{\mathbb{C},\Omega}, d^{\mathbb{C},\Omega}), \mathrm{Id}, \circ)$ / $(\mathbf{Pos\text{-}Fib}_c(d^{\mathbb{C},\Omega}, d^{\mathbb{C},\Omega}), \mathrm{Id}, \circ)$ respectively. To study these monoidal categories and monoids therein, we introduce the closely related concept of *monotone algebras* over Ω. In the coalgebra community, it has been used to determine predicate liftings of coalgebra functors [19,38,5].

Definition 4.3 Let $F : \mathbb{C} \to \mathbb{C}$ be a functor. A *monotone F-algebra over Ω* is an F-algebra $o : F\Omega \to \Omega$ such that the implication $i \leq_X i' \implies o \circ Fi \leq_{FX} o \circ Fi'$ holds for any $i, i' \in \mathbb{C}(X, \Omega)$.

From this definition, a monotone F-algebra $o : F\Omega \to \Omega$ determines a *lifting* $F_o : \mathbb{C}/\Omega \to \mathbb{C}/\Omega$ of the functor F along $d^{\mathbb{C},\Omega} : \mathbb{C}/\Omega \to \mathbb{C}$. It is given by $F_o(i) \triangleq o \circ Fi$. To understand the assignment $o \mapsto F_o$ better, we form the category $\mathbf{MAlg}^{\mathbb{C},\Omega}$ of monotone algebras over Ω as follows. An object is a pair (F, f) of a functor $F : \mathbb{C} \to \mathbb{C}$ and a monotone F-algebra o over Ω. A morphism from (F, o) to (G, o') is a natural transformation $\alpha : F \to G$ such that $o \leq_{F\Omega} o' \circ \alpha_\Omega$. The unit and tensor product are defined by $\mathbf{I} \triangleq (\mathrm{Id}_{\mathbb{C}}, \mathrm{id}_\Omega)$ and $(F, o) \otimes (G, o') \triangleq (F \circ G, o \circ Fo')$. We define $\mathbf{MAlg}_c^{\mathbb{C},\Omega}$ to be the wide subcategory of $\mathbf{MAlg}^{\mathbb{C},\Omega}$ where a morphism from (F, o) to (G, o') is a natural transformation such that $o = o' \circ \alpha_\Omega$. The following result lays out the relationship between fibered functors and monotone algebras.

Theorem 4.4 *There are monoidal isomorphisms of categories*

$$(\mathbf{MAlg}^{\mathbb{C},\Omega}, \mathbf{I}, \otimes) \simeq (\mathbf{Pos}\text{-}\mathbf{Fib}(d^{\mathbb{C},\Omega}, d^{\mathbb{C},\Omega}), \mathrm{Id}, \circ), \quad (\mathbf{MAlg}_c^{\mathbb{C},\Omega}, \mathbf{I}, \otimes) \simeq (\mathbf{Pos}\text{-}\mathbf{Fib}_c(d^{\mathbb{C},\Omega}, d^{\mathbb{C},\Omega}), \mathrm{Id}, \circ).$$

The object part of this isomorphisms is already shown as [5, Proposition 12]. By unfolding the definition, a monoid object in $\mathbf{MAlg}^{\mathbb{C},\Omega}$ consists of a monad $(\mathcal{T}, \eta, \mu)$ and a monotone $\mathcal{T}$-algebra $o : \mathcal{T}\Omega \to \Omega$ satisfying $\mathrm{id}_\Omega \leq o \circ \eta_\Omega$ and $o \circ \mathcal{T}o \leq o \circ \mu_\Omega$. We call o a *lax EM monotone $\mathcal{T}$-algebra* over Ω. A monoid object in $\mathbf{MAlg}_c^{\mathbb{C},\Omega}$ is exactly a pair of a monad $\mathcal{T}$ and an EM monotone $\mathcal{T}$-algebra over Ω. From Theorem 4.4, we obtain the following correspondence:

Corollary 4.5 *For any monad $\mathcal{T}$ on $\mathbb{C}$, there is a bijective correspondence between a (resp. lax) EM monotone $\mathcal{T}$-algebra over Ω and a Cartesian (resp. fibered) lifting of $\mathcal{T}$ along $d^{\mathbb{C},\Omega}$.*

We end this section by summarizing wppts of the Cartesian Dijkstra structures arising from EM monotone algebras.

Corollary 4.6 *Let $\mathcal{T}$ be a monad on $\mathbb{C}$, o be an EM monotone $\mathcal{T}$-algebra over Ω, and $\mathcal{T}_o$ be the corresponding Cartesian lifting of $\mathcal{T}$ along $d^{\mathbb{C},\Omega}$ by Corollary 4.5. The wppt in the Dijkstra structure $(d^{\mathbb{C},\Omega}, \mathcal{T}, \mathcal{T}_o)$ satisfies*

$$\mathrm{wp}(f, i) = o \circ Ti \circ f.$$

The same holds when o is a lax EM monotone $\mathcal{T}$-algebra and $\mathcal{T}_o$ is the corresponding fibered lifting of $\mathcal{T}$.

4.1 Further Restriction of Domain Fibrations

We consider imposing further conditions on the partial order given to the homset $\mathbb{C}(P, \Omega)$. Let $\mathbb{K}$ be a subcategory of **Pos**. We say that an ordered object $(\Omega, \leq)$ in $\mathbb{C}$ *belongs to* $\mathbb{K}$ if 1) the poset $(\mathbb{C}(X, \Omega), \leq_X)$ belongs to $\mathbb{K}$ for any $X \in \mathbb{C}$, and 2) for any $\mathbb{C}$-morphism $h : Y \to X$, the function $- \circ h : \mathbb{C}(X, \Omega) \to \mathbb{C}(Y, \Omega)$ is a morphism of type $(\mathbb{C}(X, \Omega), \leq_X) \to (\mathbb{C}(Y, \Omega), \leq_Y)$ in $\mathbb{K}$.

Lemma 4.7 *Let $\mathbb{K}$ be a subcategory of* **Pos**. *For any ordered $\mathbb{C}$-object $(\Omega, \leq)$ belonging to $\mathbb{K}$, the domain fibration $d^{\mathbb{C},\Omega} : \mathbb{C}/\Omega \to \mathbb{C}$ is a $\mathbb{K}$-fibration.*

Next, fix an ordered object $(\Omega, \leq)$ in $\mathbb{C}$ belonging to $\mathbb{K}$. For an endofunctor $F : \mathbb{C} \to \mathbb{C}$, we say that $o : F\Omega \to \Omega$ is a $\mathbb{K}$-*algebra* if for any $X \in \mathbb{C}$, the function $o \circ F- : \mathbb{C}(X, \Omega) \to \mathbb{C}(FX, \Omega)$ is a $\mathbb{K}$-morphism of type $(\mathbb{C}(X, \Omega), \leq_X) \to (\mathbb{C}(FX, \Omega), \leq_{FX})$. We write $\mathbb{K}\text{-}\mathbf{Alg}_c^{\mathbb{C},\Omega}$ for the full (monoidal) subcategory of $\mathbf{MAlg}_c^{\mathbb{C},\Omega}$ consisting of $\mathbb{K}$-algebras.

Theorem 4.8 *Let $\mathbb{K}$ be a subcategory of* **Pos** *and $(\Omega, \leq)$ be an ordered $\mathbb{C}$-object belonging to $\mathbb{K}$. There is a monoidal isomorphism of categories $(\mathbb{K}\text{-}\mathbf{Alg}_c^{\mathbb{C},\Omega}, \mathbf{I}, \otimes) \simeq (\mathbb{K}\text{-}\mathbf{Fib}_c(d^{\mathbb{C},\Omega}, d^{\mathbb{C},\Omega}), \mathrm{Id}, \circ)$.*

Example 4.9 We write $\mathbf{CLat}_\wedge$ for the category of complete lattices and monotone functions preserving all meets. This is a subcategory of $\mathbf{Pos}$. When we view the two-point poset $\mathbf{2} = \{\bot \leq \top\}$ as an ordered object in $\mathbf{Set}$ (as done in Example 4.2), it belongs to $\mathbf{CLat}_\wedge$. Therefore $d^{\mathbf{Set},\mathbf{2}} : \mathbf{Set}/\mathbf{2} \to \mathbf{Set}$ is a $\mathbf{CLat}_\wedge$-fibration. Furthermore, for any $(\mathcal{T}, \dot{\mathcal{T}}) \in \mathbf{CLat}_\wedge\text{-}\mathbf{Fib}_c(d^{\mathbf{Set},\mathbf{2}}, d^{\mathbf{Set},\mathbf{2}})$ the associated wppt preserves meets, i.e. $\mathrm{wp}(f, \bigwedge_i P_i) = \bigwedge_i \mathrm{wp}(f, P_i)$. In the context of program verification, such wppts are called *conjunctive*.

Strongest Postcondition Predicate Transformers

We turn our attention to *strongest postcondition predicate transformers*. In Hoare logic, the *strongest postcondition* with respect to a program f and a precondition ϕ is a formula ψ such that 1) $\phi\{f\}\psi$ holds, and 2) ψ entails ξ in the Hoare logic whenever $\phi\{f\}\xi$ holds. We formulate the concept of strongest postconditions in a Dijkstra structure $(p : \mathbb{P} \to \mathbb{C}, \mathcal{T}, \dot{\mathcal{T}})$. The strongest postcondition with respect to a morphism $f : X \to \dot{\mathcal{T}}Y$ in $\mathbb{C}$ and an object $P \in \mathbb{P}_X$ is an object $S \in \mathbb{P}_Y$ such that for any $Q \in \mathbb{P}_Y$, $S \leq Q$ if and only if $P\{f\}Q$. If such S exists, it is uniquely determined by f and P, hence we write it by $\mathrm{sp}(f, P)$. To summarize, we obtain the three-way equivalence:

$$\mathrm{sp}(f, P) \leq Q \iff P\{f\}Q \iff P \leq \mathrm{wp}(f, Q).$$

Therefore if $\mathrm{sp}(f, P)$ exists for any f, P, we obtain a function $\mathrm{sp} : \mathbb{C}(X, \mathcal{T}Y) \times \mathbb{P}_X \to \mathbb{P}_Y$ called the *strongest postcondition predicate transformer*, and $\mathrm{sp}(f, -)$ is characterized as a *left adjoint* of $\mathrm{wp}(f, -)$.

When programs are simply modeled as endofunctions over the set of memory configurations, sppts correspond to *direct images*. A categorical abstraction of this operation is *pushforward* [31, Example 6] in *op-fibrations*, which we briefly review below. A functor $p : \mathbb{P} \to \mathbb{C}$ is an op-fibration if $p^{op} : \mathbb{P}^{op} \to \mathbb{C}^{op}$ is a fibration. Given $P \in \mathbb{P}$, $Y \in \mathbb{C}$ and $f : pP \to Y$ in $\mathbb{C}$, the op-Cartesian lifting of f along the op-fibration is denoted by $\underline{f}P$, and its target by f_*P, which is called the *pushforward* of P along f. A functor that is both a fibration and an op-fibration is known as a *bifibration*. Posetal bifibrations can be characterized by the following lemma:

Lemma 4.10 ([14]) *Let $\mathbf{Pos}_r$ be the subcategory of $\mathbf{Pos}$ where morphisms are restricted to right adjoints between posets. A posetal fibration is a bifibration if and only if it is a $\mathbf{Pos}_r$-fibration.*

Example 4.11 Consider the case of $\mathbf{CLat}_\wedge$-fibrations. Since reindexing functors preserve arbitrary meets, they have left adjoints, which we can be computed as $f_*P = \bigwedge\{Q \in \mathbb{P}_{\mathrm{cod}f} \mid P \leq f^*Q\}$. Since $\mathbf{CLat}_\wedge$ is a (full) subcategory of $\mathbf{Pos}_r$, $\mathbf{CLat}_\wedge$-fibrations are posetal bifibrations.

Existence of left adjoints allows us to define sppt's as stated by the following result:

Proposition 4.12 *Let $(p : \mathbb{P} \to \mathbb{C}, \mathcal{T}, \dot{\mathcal{T}})$ be a Dijkstra structure where p is a posetal bifibration, and assume that the restriction $\dot{\mathcal{T}} : \mathbb{P}_X \to \mathbb{P}_{\mathcal{T}X}$ of $\dot{\mathcal{T}}$ to each*

fiber is a right adjoint. Then for any $f : X \to \mathcal{T}Y$ *in* $\mathbb{C}$, $\mathrm{wp}(f, -) : \mathbb{P}_Y \to \mathbb{P}_X$ *has a left adjoint* $\mathrm{sp}(f, -) : \mathbb{P}_X \to \mathbb{P}_Y$. *If the Dijkstra structure is Cartesian,* sp *is composable:* $\mathrm{sp}(\eta_{pP}, P) = P$ *and* $\mathrm{sp}(f, \mathrm{sp}(g, P)) = \mathrm{sp}(f \bullet g, P)$.

4.2 Lifting of Composite Monads to the Lax Slice Category

In many situations, monads for modeling computational effects are composites of simpler monads through distributive laws. We extend our lifting theory of monads to composite monads.

Let $\mathcal{T}, \mathcal{S}$ be monads on $\mathbb{C}$ and $\alpha : \mathcal{S}\mathcal{T} \to \mathcal{T}\mathcal{S}$ be a distributive law (see Section 2). We are interested in EM monotone algebras of $\mathcal{T} \circ_\alpha \mathcal{S}$, which give Cartesian liftings of $\mathcal{T} \circ_\alpha \mathcal{S}$. Such EM algebras are studied in [4, Section 2], [27, Theorem 2.4.3]. Below we mildly extend these results.

Theorem 4.13 *Let* $\mathcal{T}, \mathcal{S}$ *be monads on* $\mathbb{C}$ *and* $\alpha : \mathcal{S}\mathcal{T} \to \mathcal{T}\mathcal{S}$ *be a distributive law. There is a bijective correspondence between each two of:*

(i) *An EM monotone* $\mathcal{T} \circ_\alpha \mathcal{S}$*-algebra.*

(ii) *A pair of EM monotone algebras* $t : \mathcal{T}\Omega \to \Omega$ *and* $s : \mathcal{S}\Omega \to \Omega$ *satisfying*

$$s \circ \mathcal{S}t = t \circ \mathcal{T}s \circ \alpha_\Omega; \tag{5}$$

(iii) *A triple of Cartesian liftings* $\dot{\mathcal{T}}$ *of* $\mathcal{T}$ *and* $\dot{\mathcal{S}}$ *of* $\mathcal{S}$ *along* $d^{\mathbb{C},\Omega}$ *and a distributive law* $\dot{\alpha} : \dot{\mathcal{S}}\dot{\mathcal{T}} \to \dot{\mathcal{T}}\dot{\mathcal{S}}$ *above* α *whose components are Cartesian.*

We note that the condition (5) appeared in [4, Section 2]. This theorem says that to identify liftings of $\mathcal{T} \circ_\alpha \mathcal{S}$, it suffices to identify all the pairs of EM monotone $\mathcal{T}$- and $\mathcal{S}$-algebras satisfying (5).

It is natural to seek for a lax version of the above correspondence. However, it cannot be fully generalized to the ordered setting. The problem comes from not being able to compare certain morphisms involving composition of algebras. Specifically, $o : \mathcal{T}\mathcal{S}\Omega \to \Omega$ and $o \circ \mathcal{T}(\eta^{\mathcal{S}}_\Omega \circ o \circ \eta^{\mathcal{T}}_{\mathcal{S}\Omega})$ are in general not comparable. But when o is the composite $t \circ \mathcal{T}s$ of lax EM monotone algebras $t : \mathcal{T}\Omega \to \Omega$ and $s : \mathcal{S}\Omega \to \Omega$, we do have $o \leq o \circ \mathcal{T}(\eta^{\mathcal{S}}_\Omega \circ o \circ \eta^{\mathcal{T}}_{\mathcal{S}\Omega})$. Therefore, we can find an isomorphism when we restrict it to these "well-behaved algebras".

Theorem 4.14 *Let* $\mathcal{T}, \mathcal{S}$ *be monads on* $\mathbb{C}$ *and* $\alpha : \mathcal{S}\mathcal{T} \to \mathcal{T}\mathcal{S}$ *a distributive law. Then a lax EM monotone* $\mathcal{T} \circ_\alpha \mathcal{S}$*-algebra* o *such that* $o \leq o \circ \mathcal{T}(\eta^{\mathcal{S}}_\Omega \circ o \circ \eta^{\mathcal{T}}_{\mathcal{S}\Omega})$ *bijectively corresponds to a pair of lax EM monotone algebras* $t : \mathcal{T}\Omega \to \Omega$ *and* $s : \mathcal{S}\Omega \to \Omega$ *such that* $s \circ \mathcal{S}t \leq t \circ \mathcal{T}s \circ \alpha_\Omega, t = t \circ \mathcal{T}s \circ \mathcal{T}\eta^{\mathcal{S}}_\Omega$ *and* $s = t \circ \mathcal{T}s \circ \eta^{\mathcal{T}}_{\mathcal{S}\Omega}$.

4.3 Algebraic Operations and Weakest Precondition Predicate Transformer

Various program constructs that cause computational effects can be naturally modeled by Plotkin and Power's *algebraic operations* [34]. They frequently occur in the monadic semantics of effectful programs. In this section, we give a wppt semantics of algebraic operations. We assume that $\mathbb{C}$ is Cartesian closed and T has a strength

$\theta_{X,Y} : X \times \mathcal{T}Y \to \mathcal{T}(X \times Y)$. We use notations $\Rightarrow, \lambda, \lambda^{-1}, ev$ for the exponential object, currying, uncurrying and evaluation.

Any morphism $g : A \to \mathcal{T}B$ (called *generic effect*) equips an EM $\mathcal{T}$-algebra $o : \mathcal{T}X \to X$ with the following operation $op(o, g) : A \times (B \Rightarrow X) \to X$:

$$op(o, g) \triangleq o \circ \mathcal{T}(ev) \circ \theta'_{B,B\Rightarrow X} \circ (g \times id_{B\Rightarrow X});$$

here $\theta'_{X,Y} : \mathcal{T}X \times Y \to \mathcal{T}(X \times Y)$ is the symmetric version of θ. Specifically, the family of operations on free algebras $\{\alpha(g)_X\}_{X\in\mathbb{C}}$ defined by $\alpha(g)_X = op(\mu_X, g)$ forms the *algebraic operation* corresponding to g. The following theorem helps us (in Section 6.2) to compute the wppt of programs involving algebraic operations:

Theorem 4.15 *Let $o : \mathcal{T}\Omega \to \Omega$ be an EM monotone $\mathcal{T}$-algebra and $\mathcal{T}_o$ be the Cartesian lifting of $\mathcal{T}$ along $d^{\mathbb{C},\Omega}$ by o. For any morphisms $g : A \to \mathcal{T}B, a : Y \to A$, $b : Y \to B \Rightarrow \mathcal{T}X$ and $i : X \to \Omega$, the following equality holds in the Dijkstra structure $(d^{\mathbb{C},\Omega}, \mathcal{T}, \mathcal{T}_o)$:*

$$\mathrm{wp}(\alpha(g)_X \circ \langle a, b\rangle, i) = op(o, g) \circ \langle a, \lambda(\mathrm{wp}(\lambda^{-1}(b), i))\rangle.$$

5 Dijkstra Structures on $d^{\mathbf{Set},\mathbf{2}}$

We apply the techniques developed in the previous sections to concrete settings. We first consider the lax slice category with the poset $\mathbf{2} = \{\bot \leq \top\}$, viewed as an ordered object in **Set** (Example 4.2). We identify an object $i : X \to \mathbf{2}$ in $\mathbf{Set}/\mathbf{2}$ as a pair (X, P) of sets such that $P \subseteq X$. The fibration $d^{\mathbf{Set},\mathbf{2}} : \mathbf{Set}/\mathbf{2} \to \mathbf{Set}$ is isomorphic to the subobject fibration (see e.g. [14, Chapter 0]) of **Set**.

Example 5.1 [11, Section 2.3] As a warm-up exercise, we consider Cartesian liftings of the *maybe monad* (a.k.a. partiality monad) $\mathcal{M}$ along $d^{\mathbf{Set},\mathbf{2}}$. Its functor part is defined by $MX \triangleq X + \{*\}$, where $\{*\}$ denotes the singleton set. There are exactly two EM monotone $\mathcal{M}$-algebras $\mathsf{tot}, \mathsf{par}$ over $\mathbf{2}$:

$$\mathsf{tot}(x) = \top \iff x = \top, \qquad \mathsf{par}(x) = \top \iff (x = \top \vee x = *).$$

They respectively induce two Cartesian liftings $\mathcal{M}_{\mathsf{tot}}, \mathcal{M}_{\mathsf{par}}$ of $\mathcal{M}$ along $d^{\mathbf{Set},\mathbf{2}}$. The Dijkstra structures associated to these Cartesian liftings offer the *total* and *partial* correctness interpretations of Hoare triples.

EM mono. $\mathcal{M}$-Alg o	Lifting $\mathcal{M}_o$, object part	$(X, P)\{f\}(Y, Q)$ in $(d^{\mathbf{Set},\mathbf{2}}, \mathcal{M}, \mathcal{M}_o)$
tot	$\mathcal{M}_{\mathsf{tot}}(X, P) = (\mathcal{M}X, P)$	$\forall x \in X \,.\, x \in P \implies f(x) \neq * \wedge f(x) \in Q$
par	$\mathcal{M}_{\mathsf{par}}(X, P) = (\mathcal{M}X, P \cup \{*\})$	$\forall x \in X \,.\, x \in P \wedge f(x) \neq * \implies f(x) \in Q$

Example 5.2 [13, Example 3.3] We next consider the Cartesian liftings of the *nonempty powerset monad* $\mathcal{P}^+$ along $d^{\mathbf{Set},\mathbf{2}}$. Its functor part is defined by $\mathcal{P}^+X \triangleq \{U \subseteq X \mid U \neq \emptyset\}$. There are exactly two monotone EM $\mathcal{P}^+$-algebras $\mathsf{may}, \mathsf{must}$ over $\mathbf{2}$:

$$\mathsf{may}(U) = \top \iff \top \in U, \qquad \mathsf{must}(U) = \top \iff \bot \notin U$$

They induce two Cartesian liftings $\mathcal{P}^+_{\mathsf{may}}, \mathcal{P}^+_{\mathsf{must}}$ of $\mathcal{P}^+$ along $d^{\mathbf{Set},\mathbf{2}}$. The Dijkstra structures with these Cartesian liftings of $\mathcal{P}^+$ along $d^{\mathbf{Set},\mathbf{2}}$ offer the *may* and *must* correctness interpretations of Hoare triples.

EM mono. $\mathcal{P}^+$-Alg o	Lifting $\mathcal{P}^+_o$, object part	$(X,P)\{f\}(Y,Q)$ in $(d^{\mathbf{Set},\mathbf{2}}, \mathcal{P}^+, \mathcal{P}^+_o)$
may	$\mathcal{P}^+_{\mathsf{may}}(X,P) = (\mathcal{P}^+X, \{U \mid U \cap P \neq \emptyset\})$	$\forall x \in X \,.\, x \in P \implies \exists u \in f(x) \,.\, u \in Q$
must	$\mathcal{P}^+_{\mathsf{must}}(X,P) = (\mathcal{P}^+X, \{U \mid U \subseteq P\})$	$\forall x \in X \,.\, x \in P \implies \forall u \in f(x) \,.\, u \in Q$

Example 5.3 We consider again the $\mathcal{P}^+$ monad of the previous example, but now we regard $d^{\mathbf{Set},\mathbf{2}}$ as a $\mathbf{CLat}_\wedge$-fibration. Then must above is the only EM $\mathbf{CLat}_\wedge$-algebra over $\mathbf{2}$. Therefore *must* correctness interpretation of the Hoare triple has the strongest postcondition predicate transformer.

Example 5.4 We consider the state monad $\mathcal{S}$, whose functor part is defined by $\mathcal{S}\,X = S \Rightarrow X \times S$; here S is the set of states. However, the two-point set $\mathbf{2}$ is too small to remember stored values when $|S| \geq 2$. Indeed,

Theorem 5.5 *When $|S| \geq 2$, there is no EM monotone $\mathcal{S}$-algebra over $\mathbf{2}$.*

The proof idea is simple, every computation needs to be associated to an element of 2 in such a way that they satisfy the algebraic theory of lookup and update studied by Plotkin and Power in [35]. If $|S| \geq 2$ then there exist $s_1 \neq s_2$ in S and we cannot distinguish between looking up the state, storing s_1 and storing s_2, so a contradiction can be derived.

When we replace the slicing object $\mathbf{2}$ with $S \Rightarrow \mathbf{2}$ (with the pointwise order), we find that it carries an EM monotone $\mathcal{S}$-algebra $o(k) \triangleq \lambda s \,.\, \pi_1(ks)(\pi_2(ks))$; see also [25, Section 4.4]. The corresponding Cartesian lifting of $\mathcal{S}$ along $d^{\mathbf{Set},(S\Rightarrow\mathbf{2})} : \mathbf{Set}/(S \Rightarrow \mathbf{2}) \to \mathbf{Set}$ and the derived wppt satisfies

$$\mathcal{S}_o(i)(c) = \lambda s \,.\, i(\pi_1(cs))(\pi_2(cs)), \quad \mathrm{wp}(f,i)(x) = \lambda s \,.\, i(\pi_1(f(x)(s)))(\pi_2(f(x)(s))).$$

By using the uncurrying i' and f' of i and f respectively, wppt can be simplified: $\mathrm{wp}(f,i)(x) = \lambda s \,.\, i'(f'(x,s))$.

Example 5.6 Finally, we consider the finite probability distribution monad $\mathcal{D}$. A *finite probability distribution* on a set X is a function $\mu : X \to [0,1]$ such that μ is non-zero at finitely many elements in X, and $\sum_{i \in X} \mu(i) = 1$. The functor part of the monad $\mathcal{D}$ is given by $\mathcal{D}X = \{\mu \text{ is a finite probability distribution on } X\}$.

Theorem 5.7 *There are exactly two EM monotone $\mathcal{D}$-algebras over $\mathbf{2}$, given by*

$$\mathsf{pmay}(\mu) = \top \iff \mu(\top) > 0, \qquad \mathsf{pmust}(\mu) = \top \iff \mu(\top) = 1$$

They induce two Cartesian liftings $\mathcal{D}_{\mathsf{pmay}}, \mathcal{D}_{\mathsf{pmust}}$ of $\mathcal{D}$ along $d^{\mathbf{Set},\mathbf{2}}$. The predicate transformers associated to them generalize the *may* and *must* correctness of nondeterministic programs to the probabilistic setting.

EM mono. $\mathcal{D}$-Alg o	Lifting $\mathcal{D}_o$, object part	$(X,P)\{f\}(Y,Q)$ in $(d^{\mathbf{Set},\mathbf{2}}, \mathcal{D}, \mathcal{D}_o)$
pmay	$\mathcal{D}_{\mathsf{pmay}}(X,P) = (\mathcal{D}X, \{\mu \mid \exists x \in P \,.\, \mu(x) > 0\})$,	$\forall x \in X \,.\, x \in P$ $\implies \mathsf{Pr}_{y \sim f(x)}[y \in Q] > 0$
pmust	$\mathcal{D}_{\mathsf{pmust}}(X,P) = (\mathcal{D}X, \{\mu \mid \forall x \in P \,.\, \mu(x) > 0\})$	$\forall x \in X \,.\, x \in P$ $\implies \mathsf{Pr}_{y \sim f(x)}[y \in Q] = 1$

Here the notation $\mathsf{Pr}_{y\sim f(x)}[y \in Q]$ denotes the probability of $y \in Q$ when sampling y from $f(x)$.

We note that Cartesian liftings are rather rare, compared to arbitrary liftings. We look at the case of the powerset monad $\mathcal{P}$ along $d^{\mathbf{Set},\mathbf{2}}$. For each regular cardinal λ, $\mathcal{P}_\lambda(P \subseteq X) = \{U \subseteq P \mid |U| < \lambda\}$ gives a lifting of the powerset monad $\mathcal{P}$ along $d^{\mathbf{Set},\mathbf{2}}$. On the other hand, (a modification of) Example 5.2 shows that there are only two Cartesian liftings of $\mathcal{P}$ along $d^{\mathbf{Set},\mathbf{2}}$. Example 5.4 also shows that there is *no* Cartesian lifting of the (nontrivial) state monad along $d^{\mathbf{Set},\mathbf{2}}$.

6 Dijkstra Structures on $d^{\mathbf{Set},[0,\infty]}$

We next replace $\mathbf{2}$ with the poset $[0,\infty]$ of non-negative extended reals, with the usual numerical order. It is a completely distributive lattice, and the domain fibration $d^{\mathbf{Set},[0,\infty]} : \mathbf{Set}/[0,\infty] \to \mathbf{Set}$ has a rich structure (in fact it is a tripos [33, Example 2.2]). An object of $\mathbf{Set}/[0,\infty]$ is a function $i : X \to [0,\infty]$, which can be seen in many different ways: *random variables*, assignments of *cost*, *robustness*, *fuzzy predicates*, *reward functions*, etc.

Example 6.1 We consider the *counting monad* $\mathcal{C}$ over $\mathbf{Set}$, whose functor part is given by $\mathcal{C}X \triangleq \mathbb{N} \times X$. This is identical to the writer monad with a single character ($\mathbb{N} \simeq 1^*$). Each monotone function $f : [0,\infty] \to [0,\infty]$ (bijectively) determines an EM monotone $\mathcal{C}$-algebra $\overline{f}(n,x) \triangleq f^n(x)$, and the corresponding Cartesian lifting $\mathcal{C}_{\overline{f}}$ of $\mathcal{C}$ along $d^{\mathbf{Set},[0,\infty]}$ is given by $\mathcal{C}_{\overline{f}}(i)(n,x) = f^n(i(x))$. The standard choice of $f : [0,\infty] \to [0,\infty]$ is the successor function $s(x) = x + 1$. In this case, the lifting becomes $\mathcal{C}_{\overline{s}}i = \lambda(n,x)\ .\ n + i(x)$.

Example 6.2 We consider Cartesian liftings of the nonempty powerset monad $\mathcal{P}^+$ along $d^{\mathbf{Set},[0,\infty]}$. There are at least two EM monotone $\mathcal{P}^+$-algebras $\mathsf{sup}, \mathsf{inf} : \mathcal{P}^+[0,\infty] \to [0,\infty]$. They induce two Cartesian liftings:

$$\mathcal{P}^+_{\mathsf{sup}}(i) = \lambda U\ .\ \sup_{x\in U} i(x), \quad \mathcal{P}^+_{\mathsf{inf}}(i) = \lambda U\ .\ \inf_{x\in U} i(x) \quad (i : X \to [0,\infty])$$

Example 6.3 We can view objects in $\mathbf{Set}/[0,\infty]$ as random variables. The expected value of a random variable $i : X \to [0,\infty]$ over a probability distribution $\mu \in \mathcal{D}X$, denoted by $E_{x\sim\mu}[i(x)]$, is computed as $\mathsf{E} \circ \mathcal{D}i(\mu)$, where $\mathsf{E} : \mathcal{D}[0,\infty] \to [0,\infty]$ is the expected value function on $[0,\infty]$. It is easy to check that E is an EM monotone $\mathcal{D}$-algebra over $[0,\infty]$. Therefore the expected value *is* a Cartesian lifting of $\mathcal{D}$ along $d^{\mathbf{Set},[0,\infty]}$:

$$\mathcal{D}_{\mathsf{E}}(i) \triangleq \lambda\mu\ .\ E_{x\sim\mu}[i(x)] = \lambda\mu\ .\ \sum i(x)\cdot\mu(x). \quad (i : X \to [0,\infty])$$

The wppt of the Cartesian Dijkstra structure $(d^{\mathbf{Set},[0,\infty]}, \mathcal{D}, \mathcal{D}_{\mathsf{E}})$ computes the *weakest pre-expectation* [29]:

$$\mathrm{wp}(f,i) = \lambda j\ .\ E_{x\sim f(j)}[i(x)]. \quad (f : Y \to \mathcal{D}X, i : X \to [0,\infty])$$

Example 6.4 When modeling a programming language with both nondeterministic choice and probabilistic choice, we might want to combine $\mathcal{P}^+$ and $\mathcal{D}$ via a distributive law $\mathcal{D} \circ \mathcal{P}^+ \to \mathcal{P}^+ \circ \mathcal{D}$. However, Plotkin showed that there is *no* such distributive law [40]. To remedy this, Varacca and Winskel introduced the *indexed valuation monad* [40], which does have a distributive law with $\mathcal{P}^+$.

In this paper we employ its probabilistic variant called *finite indexed distribution monad* [37]. A *finite indexed distribution* over a set X is an unordered list of pairs (p, x) where $p \in (0, 1]$ and $x \in X$, and moreover the sum of ps in the list is 1. We define $\mathcal{I}(X) \triangleq \{\mu : \text{finite indexed distribution over } X\}$. The probabilistic sum of finite indexed distributions is defined as follows: (below @ denotes list concatenation):

$$[\cdots(p_i, x_i)\cdots] \oplus_p [\cdots(q_j, y_j)\cdots] \triangleq \begin{cases} [\cdots(q_j, y_j)\cdots] & (p = 0) \\ [\cdots(p \cdot p_i, x_i)\cdots] @ [\cdots((1-p)\cdot q_j, y_j)\cdots] & (0 < p < 1) \\ [\cdots(p_i, x_i)\cdots] & (p = 1) \end{cases}$$

Every finite probability distribution may be regarded as a finite indexed distribution; we write the inclusion map by $\iota_X : \mathcal{D}(X) \to \mathcal{I}(X)$. We can extend $\mathcal{I}$ to a monad on **Set** called the *finite indexed distribution monad*. The concept of expectation can also be extended to finite indexed distributions. First, the function $\mathsf{IE} : \mathcal{I}([0,\infty]) \to [0,\infty]$ defined below gives an EM monotone $\mathcal{I}$-algebra over $[0,\infty]$:

$$\mathsf{IE}[(p_1, x_1), \cdots, (p_n, x_n)] \triangleq \sum_{i=1}^{n} p_i \cdot x_i.$$

We write $IE_{x\sim\nu}[i(x)]$ to mean $\mathsf{IE} \circ (\mathcal{I}(i)(\nu))$. We note that $\mathsf{E} = \mathsf{IE} \circ \iota$ holds. The corresponding Cartesian lifting of $\mathcal{I}$ along $d^{\mathbf{Set},[0,\infty]}$, and the wppt within the derived Cartesian Dijkstra structure is

$$\mathcal{I}_{\mathsf{IE}}(i)(\nu) \triangleq IE_{x\sim\nu}[i(x)] = \sum_{(p,x)\in\nu} p \cdot i(x), \quad \mathrm{wp}(f, i)(x) = \sum_{(p,y)\in f(x)} p \cdot i(y).$$

6.1 Dijkstra Structures for Composite Monads

Various intensional quantities of program executions can be measured using the counting monad in Example 6.1. The distributive law of any **Set**-monad $\mathcal{T}$ over the counting monad $\mathcal{C}$ exists, and is provided by the *unique strength* $\theta_{X,Y} : X \times \mathcal{T}Y \to \mathcal{T}(X \times Y)$ of $\mathcal{T}$ (see [32]). Here we discuss giving a Cartesian lifting of $\mathcal{T} \circ_\theta \mathcal{C}$.

Theorem 6.5 *Let $o : \mathcal{T}[0,\infty] \to [0,\infty]$ be an EM monotone $\mathcal{T}$-algebra and $f : [0,\infty] \to [0,\infty]$ be a monotone function. Then o and $\overline{f}$ (the EM monotone $\mathcal{C}$-algebra associated to f; Example 6.1) satisfy (5) if and only if $o \circ \mathcal{T} = f \circ o$, that is, f is a $\mathcal{T}$-algebra endomorphism over o.*

Example 6.6 We consider the instance of Theorem 6.5 where $\mathcal{T}$ is the finite probability distribution monad $\mathcal{D}$. The combined monad $\mathcal{D} \circ_\theta \mathcal{C}$ is suitable for modeling the programs that report cost counting during their executions. We will see this in Section 6.2. The EM monotone $\mathcal{D}$-algebra $\mathsf{E} : \mathcal{D}[0,\infty] \to [0,\infty]$ and the successor

function $s : [0,\infty] \to [0,\infty]$ evidently satisfies the equality condition of Theorem 6.5:

$$\mathsf{E} \circ \mathcal{D}s = \lambda\mu\ .\ E_{x\sim\mu}[1+x] = \lambda\mu\ .\ 1 + E_{x\sim\mu}[x] = s \circ \mathsf{E}.$$

Therefore E and $\overline{s}$ satisfy (5) by Theorem 6.5. The composite $\mathcal{D}_{\mathsf{E}} \circ \mathcal{C}_{\overline{s}}$ is a Cartesian lifting of $\mathcal{D} \circ_\theta \mathcal{C}$ along $d^{\mathbf{Set},[0,\infty]}$, and is computed as $\mathcal{D}_{\mathsf{E}} \circ \mathcal{C}_{\overline{s}}(i) = \lambda\mu\ .\ E_{(c,x)\sim\mu}[c + i(x)]$. The wppt in the Dijkstra structure $(d^{\mathbf{Set},[0,\infty]}, \mathcal{D} \circ_\theta \mathcal{C}, \mathcal{D}_{\mathsf{E}} \circ \mathcal{C}_{\overline{s}})$ is

$$\mathrm{wp}(f,j)(x) = E_{(c,y)\sim f(x)}[c + j(y)] \quad (f : X \to \mathcal{D} \circ_\theta \mathcal{C}(Y), j : Y \to [0,\infty]).$$

Example 6.7 The nonempty powerset monad $\mathcal{P}^+$ distributes over the finite indexed distribution monad $\mathcal{I}$. The distributive law $\gamma : \mathcal{I} \circ \mathcal{P}^+ \to \mathcal{P}^+ \circ \mathcal{I}$ is given by

$$\gamma_P[(p_1,U_1),\cdots,(p_n,U_n)] \triangleq \{[(p_1,x_1),\cdots,(p_n,x_n)] \mid x_1 \in U_1,\cdots,x_n \in U_n\}.$$

It is easy to check that the expectation function $\mathsf{IE} : \mathcal{I}([0,\infty]) \to [0,\infty]$ (Example 6.4) distributes over $\mathsf{sup} : \mathcal{P}^+[0,\infty] \to [0,\infty]$ (Example 6.2). Therefore the wppt in the Cartesian Dijkstra structure $(d^{\mathbf{Set},[0,\infty]}, \mathcal{P}^+ \circ_\gamma \mathcal{I}, \mathcal{P}^+_{\mathsf{sup}} \circ \mathcal{I}_{\mathsf{IE}})$ is

$$\mathrm{wp}(f,i)(j) = \sup_{\nu \in f(j)} IE_{x\sim\nu}[i(x)]. \quad (f : X \to \mathcal{P}^+ \circ \mathcal{I}(Y))$$

This wppt performs *angelic choice*. We can also derive the one performing *demonic choice* by replacing sup with inf in this argument. Finally, we compose it with the counting monad $\mathcal{C}$. This yields the Cartesian Dijkstra structure $(d^{\mathbf{Set},[0,\infty]}, (\mathcal{P}^+ \circ_\gamma \mathcal{I}) \circ_\theta \mathcal{C}, \mathcal{P}^+_{\mathsf{sup}} \circ \mathcal{I}_{\mathsf{IE}} \circ \mathcal{C}_{\overline{s}})$, whose wppt is:

$$\mathrm{wp}(f,i)(j) = \sup_{\nu \in f(j)} IE_{(c,x)\sim\nu}[c + i(x)].$$

6.2 Expected Runtime as Weakest Precondition

Kaminski et al. showed that the expected runtime of probabilistic programs can be computed using a predicate transformer-like operator called the *expected runtime transformer* (ert for short) [16]. We apply our theory of Dijkstra structures to derive their ert operator.

The language we consider here is a loop-free fragment of Kaminski et al.'s language. Fix sets Var, Exp, Bool of variables, probabilistic expressions and probabilistic boolean expressions, respectively. The syntax of the language is defined by the following BNF grammar (below $x \in \mathrm{Var}, e \in \mathrm{Exp}, b \in \mathrm{Bool}$):

$$\mathrm{Prog} \ni C ::= \mathrm{empty} \mid C;C \mid x :\sim e \mid \mathrm{tick} \mid \{C\}\square\{C\} \mid \mathrm{if}(b)\{C\}\{C\}.$$

The command $x :\sim e$ samples a value from a probabilistic expression e. The box operator $\{C\}\square\{C'\}$ non-deterministically chooses C or C' and executes the chosen program. The conditional expression $\mathrm{if}(b)\{C\}\{C'\}$, samples a bit from a Bernoulli distribution with bias given by an expression b, then runs C if the sample is 1;

otherwise runs C'. This conditional expression thus combines both probabilistic choice and deterministic conditionals.

Before introducing the expected runtime transformer, we prepare some interpretations of primitives. We fix a set V of values and let $M = V^{\mathrm{Var}}$ be the set of memory configurations. We give interpretations of $e \in \mathrm{Exp}$ and $b \in \mathrm{Bool}$ by functions $[\![e]\!] : M \to \mathcal{D}V$ and $[\![b]\!] : M \to [0,1]$, respectively. We also assume that the memory configuration update function $\mathrm{upd}_x : M \times V \to M$ is suitably defined for each variable $x \in \mathrm{Var}$. The expected runtime transformer $\mathrm{ert} : \mathrm{Prog} \times (\mathbf{Set}/[0,\infty])_M \to (\mathbf{Set}/[0,\infty])_M$ is inductively defined as follows:

$$\begin{array}{ll} \mathrm{ert}(\mathrm{empty}, i)\rho = i(\rho) & \mathrm{ert}(\mathrm{tick}, i)\rho = 1 + i(\rho) \\ \mathrm{ert}(C_1; C_2, i)\rho = \mathrm{ert}(C_1, \mathrm{ert}(C_2, i))\rho & \mathrm{ert}(\{C_1\}\square\{C_2\}, i)\rho = \max(\mathrm{ert}(C_1, i)\rho, \mathrm{ert}(C_2, i)\rho) \\ \mathrm{ert}(x :\sim e, i)\rho = 1 + E_{v\sim[\![e]\!]\rho}[i(\mathrm{upd}_x(\rho, v))] & \mathrm{ert}(\mathrm{if}(b)\{C_1\}\{C_2\}, i)\rho = 1 + \mathrm{ert}(C_1, i)\rho \oplus_{[\![b]\!]\rho} \mathrm{ert}(C_2, i)\rho \end{array}$$

To derive the ert in our framework, we first give a monadic semantics of the language using the composite monad $\mathcal{T} \triangleq (\mathcal{P}^+ \circ_\gamma \mathcal{I}) \circ_\theta \mathcal{C}$ in Example 6.7. We interpret effectful commands and conditional using the following generic operations and associated $\mathcal{T}$-algebraic operations (c.f. Section 4.3):

$$\begin{array}{ll} t(*) = \{[(1,(1,*))]\} & \alpha(t)(U) = \{\mathcal{I}(\lambda x\ .\ x+1)(\nu) | \nu \in U\} \\ u(*) = \{[(1,(0,\top))], [(1,(0,\bot))]\} & \alpha(u)(U,V) = U \cup V \\ c(p) = \{[(p,(0,\top)), (1-p,(0,\bot))]\} & \alpha(c)(p,U,V) = \{\nu \oplus_p \mu | \nu \in U, \mu \in V\} \end{array}$$

From these algebraic operations we define our denotational semantics:

$$\begin{array}{ll} [\![\mathrm{empty}]\!] = \eta^{\mathcal{T}}_M & [\![\mathrm{tick}]\!] = \alpha(t)_M \circ \eta^{\mathcal{T}}_M \\ [\![C_1; C_2]\!] = [\![C_2]\!] \bullet [\![C_1]\!] & [\![\{C_1\}\square\{C_2\}]\!] = \alpha(u)_M \circ \langle [\![C_1]\!], [\![C_2]\!] \rangle \\ [\![x :\sim e]\!] = \mathcal{T}(\mathrm{upd}_x) \circ \theta_{M,V} \circ \langle \mathrm{id}_M, \iota \circ [\![e]\!] \rangle & [\![\mathrm{if}(b)\{C_1\}\{C_2\}]\!] = \alpha(c)_M \circ \langle [\![b]\!], [\![C_1]\!], [\![C_2]\!] \rangle \end{array}$$

Recall that in Example 6.7, we obtained a Cartesian lifting $\dot{\mathcal{T}} \triangleq \mathcal{P}^+_{\mathsf{sup}} \circ \mathcal{I}_{\mathsf{IE}} \circ \mathcal{C}_{\overline{s}}$ of $\mathcal{T}$ along $d^{\mathbf{Set},[0,\infty]}$. In the Dijkstra structure $(d^{\mathbf{Set},[0,\infty]}, \mathcal{T}, \dot{\mathcal{T}})$, $\mathrm{wp}([\![C]\!], i)$ satisfies

$$\begin{array}{ll} \mathrm{wp}([\![\mathrm{empty}]\!], i)\rho = i(\rho) & \mathrm{wp}([\![\mathrm{tick}]\!], i)\rho = 1 + i(\rho) \\ \mathrm{wp}([\![C_1; C_2]\!], i)\rho = \mathrm{wp}([\![C_1]\!], \mathrm{wp}([\![C_2]\!], i))\rho & \mathrm{wp}([\![\{C_1\}\square\{C_2\}]\!], i)\rho = \max(\mathrm{wp}([\![C_1]\!], i)\rho, \mathrm{wp}([\![C_2]\!], i)\rho) \\ \mathrm{wp}([\![x :\sim e]\!], i)\rho = E_{v\sim[\![e]\!]\rho}[i(\mathrm{upd}_x(\rho, v))] & \mathrm{wp}([\![\mathrm{if}(b)\{C_1\}\{C_2\}]\!], i)\rho = \mathrm{wp}([\![C_1]\!], i)\rho \oplus_{[\![b]\!]\rho} \mathrm{wp}([\![C_2]\!], i)\rho \end{array}$$

This is already very close to Kaminski et al's expected runtime transformer. The major difference between wp and ert is that the latter adds 1 to reflect the time consumption by probabilistic assignments and probabilistic conditionals. This difference can be resolved by an *accounting transformation* A defined below. It inserts a tick command before time-consuming instructions:

$$\begin{array}{lll} A(\mathrm{empty}) = \mathrm{empty}, & A(x :\sim e) = \mathrm{tick}; x :\sim e & A(\{C\}\square\{C'\}) = \{A(C)\}\square\{A(C')\} \\ A(C;C) = A(C); A(C') & A(\mathrm{tick}) = \mathrm{tick} & A(\mathrm{if}(b)\{C\}\{C'\}) = \mathrm{tick}; \mathrm{if}(b)\{C\}\{C'\} \end{array}$$

Theorem 6.8 *Under the Cartesian Dijkstra structure* $(d^{\mathbf{Set},[0,\infty]}, \mathcal{T}, \dot{\mathcal{T}})$*, we have* $\mathrm{ert}(C, i) = \mathrm{wp}([\![A(C)]\!], i)$.

7 Dijkstra Structures on $d^{\mathbf{Set},[0,\infty]^n}$

As our final example we consider predicates that take values over $[0,\infty]^n$. Kura et al. [21] study the problem of estimating the value of higher moments of the runtime of a program, with the goal of computing bounds on tail probabilities. That is, their techniques allow us to estimate expected values of the form $E[T^n]$, where T is the runtime of an execution of a program. These higher-order moments are computed with a wppt-like operator that acts on the set of configurations of a control-flow graph and that propagates functions from configurations to $[0,\infty]^n$ backwards through the graph.

We derive this operator as a wppt in a suitable Cartesian Dijkstra structure on a domain fibration (unlike [21] here we do not consider the nondeterministic computations). Consider again the composite $\mathcal{D} \circ_\theta \mathcal{C}$ of the counter monad and the finite probability distribution monad considered in Example 6.6. Let U_n denote the upper-triangular $n \times n$ matrix such that $(U_n)_{i,j} = \binom{j}{i}$ whenever $i \leq j$ and 0 otherwise. Then we define a family of operation called $\mathrm{tick}_n \colon [0,\infty]^n \to [0,\infty]^n$ to be the affine map: $\mathrm{tick}_n(r_1,\ldots,r_n) = (r_1,\ldots,r_n) \cdot U_n + (1,\ldots,1)$. This coincides with the *elapse function* defined in [21]. From this, we derive a family of EM monotone $\mathcal{C}$-algebras $\overline{\mathrm{tick}_n} \colon \mathcal{C}[0,\infty]^n \to [0,\infty]^n$. On the other hand, we take the product of n-many copies of the EM monotone $\mathcal{D}$-algebra of $\mathsf{E} : \mathcal{D}[0,\infty] \to [0,\infty]$ in Example 6.3. We name the resulting monotone algebra $\mathsf{E}^n \colon \mathcal{D}([0,\infty]^n) \to [0,\infty]^n$. It follows from affinity of the tick_n operator and linearity of the expected value that tick_n is a $\mathcal{D}$-algebra endomorphism over E^n. From Theorem 6.5, we obtain the Cartesian lifting $\mathcal{D}_{\mathsf{E}^n} \circ \mathcal{C}_{\overline{\mathrm{tick}_n}}$ of $\mathcal{D} \circ_\theta \mathcal{C}$ along $d^{\mathbf{Set},[0,\infty]^n} : \mathbf{Set}/[0,\infty]^n \to \mathbf{Set}$. In the Dijkstra structure $(d^{\mathbf{Set},[0,\infty]^n}, \mathcal{D} \circ_\theta \mathcal{C}, \mathcal{D}_{\mathsf{E}^n} \circ \mathcal{C}_{\overline{\mathrm{tick}_n}})$, the wppt satisfies the definition below, which coincides with the F_n operator from [21]:

$$\mathrm{wp}(f,p)(j) \triangleq (\mathrm{wp}_1(f,p)(j),\ldots,\mathrm{wp}_n(f,p)(j)), \quad \mathrm{wp}_k(f,p)(j) \triangleq \sum_{c\in[0,\infty],i\in X} f(j)(c,i) \cdot \sum_{l=0}^{k} \binom{k}{l} c^{l-k} p_l(i).$$

8 Dijkstra Structures on Relational Fibrations

So far we have seen examples of fibrations from lax slice categories, whose objects represent Ω-valued predicates as morphisms of type $X \to \Omega$. In this section we consider Ω-valued *relations* represented as morphisms of type $X \otimes Y \to \Omega$, and consider Cartesian liftings of *monoidal monads* to Ω-valued relations.

Let $(\mathbb{C}, \mathbf{I}, \otimes)$ be a monoidal category and $(\Omega, \leq)$ be an ordered object in $\mathbb{C}$. We take the pullback of $d^{\mathbb{C},\Omega} : \mathbb{C}/\Omega \to \mathbb{C}$ along the tensor product functor $\otimes : \mathbb{C}^2 \to \mathbb{C}$ in **CAT**:

$$\begin{array}{ccc} \mathbf{BRel}(\mathbb{C},\Omega) & \longrightarrow & \mathbb{C}/\Omega \\ {\scriptstyle r^{\mathbb{C},\Omega}}\downarrow & \lrcorner & \downarrow{\scriptstyle d^{\mathbb{C},\Omega}} \\ \mathbb{C}^2 & \xrightarrow[\otimes]{} & \mathbb{C} \end{array}$$

This is a typical way to derive fibrations for binary relations [14,5]. The resulting projection functor, which we name $r^{\mathbb{C},\Omega} : \mathbf{BRel}(\mathbb{C},\Omega) \to \mathbb{C}^2$, is a fibration. The

concrete description of $\mathbf{BRel}(\mathbb{C}, \Omega)$ is the following: objects are triples $(X, Y, i : X \otimes Y \to \Omega)$ of two objects and a morphism in $\mathbb{C}$, and a morphism from (X, Y, i) to (X', Y', i') is a pair of $\mathbb{C}$-morphisms $f : X \to X'$ and $g : Y \to Y'$ such that $i \leq_{X \otimes Y} i' \circ (f \otimes g)$. Intuitively, $\mathbf{BRel}(\mathbb{C}, \Omega)$ is the category of Ω-valued binary relations between $\mathbb{C}$-objects, represented as morphisms in $\mathbb{C}$. In the fibration $r^{\mathbb{C},\Omega}$, the pullback is given by $(f, g)^*(X, Y, i) = i \circ (f \otimes g)$.

Theorem 8.1 *Let $\mathbb{C}$ be a monoidal category, $(\mathcal{T}, m_{\mathbf{I}}, m_{X,Y}, \eta, \mu)$ be a monoidal monad on it, and o be an EM monotone $\mathcal{T}$-algebra over Ω. Then the mapping $\mathcal{T}_o$ defined by $\mathcal{T}_o(X, Y, i) = (\mathcal{T}X, \mathcal{T}Y, o \circ \mathcal{T}i \circ m_{X,Y})$ extends to a Cartesian lifting of the monad $\mathcal{T}^2$ on $\mathbb{C}^2$ along $r^{\mathbb{C},\Omega} : \mathbf{BRel}(\mathbb{C}, \Omega) \to \mathbb{C}^2$.*

We first set $\mathbb{C} = \mathbf{Set}, \Omega = \mathbf{2}$, and instantiate Theorem 8.1 with $\mathcal{T} = \mathcal{P}^+$ and $o \in \{\mathsf{may}, \mathsf{must}\}$ (Example 5.2); notice that the monad $\mathcal{P}^+$ is commutative, hence carries a symmetric monoidal structure with respect to the Cartesian monoidal structure of **Set** [20]. We identify an object in $\mathbf{BRel}(\mathbf{Set}, \mathbf{2})$ and a triple of sets (X, Y, R) where $R \subseteq X \times Y$. The theorem derives the following Cartesian liftings of $(\mathcal{P}^+)^2$ along $r^{\mathbf{Set},\mathbf{2}} : \mathbf{BRel}(\mathbf{Set}, \mathbf{2}) \to \mathbf{Set}^2$:

$$\mathcal{T}_{\mathsf{may}}(X, Y, R) = (\mathcal{P}^+X, \mathcal{P}^+Y, \{(U, V) \mid \exists u \in U \,.\, \exists v \in V \,.\, (u, v) \in R\})$$
$$\mathcal{T}_{\mathsf{must}}(X, Y, R) = (\mathcal{P}^+X, \mathcal{P}^+Y, \{(U, V) \mid \forall u \in U \,.\, \forall v \in V \,.\, (u, v) \in R\}).$$

We next set $\mathbb{C} = \mathbf{Set}, \Omega = [0, \infty]$ and instantiate Theorem 8.1 with $\mathcal{T} = \mathcal{D}$, which is commutative, and $o = \mathsf{E}$ (Example 6.3). The theorem derives the following Cartesian lifting of D^2 along $r^{\mathbf{Set},[0,\infty]} : \mathbf{BRel}(\mathbf{Set}, [0, \infty]) \to \mathbf{Set}^2$; below $\mu \otimes \nu$ is the product distribution:

$$D_{\mathsf{E}}(X, Y, i) = (DX, DY, \lambda(\mu, \nu) \,.\, E_{(x,y)\sim\mu\otimes\nu}[i(x, y)]).$$

9 Related Work

The interpretation of Hoare logic has been pursued using other structures than fibrations. Abramsky et al. introduce *specification structures* as a loose framework for general Hoare logic [1]. It is easy to see that a Dijkstra structure in our setting determines a specification structure. Specification structures themselves do not provide wppts, and computational effects are not explicitly modeled.

Martin et al. [28] give a categorical framework for Hoare logic using certain functors of type $H : S \to \mathbf{PreOrd}$. They correspond to *opfibrations*, hence they offers *strongest postconditions predicate transformers* instead. Computational effects are not explicitly modeled. One unique feature of [28] is that their Hoare logic supports *trace operators* in the sense of Joyal et al [15]. Supporting (possibly diverging) loop in our framework is missing, and an interesting future work.

In [9], Goncharov et al. constructed semantics of Hoare logic from a pair $(\mathcal{P}, \mathcal{T})$ of an order-enriched monad $\mathcal{T}$ and an *innocent submonad* $\mathcal{P}$ of $\mathcal{T}$. They showed that $\mathcal{P}1$ is a frame, and used this fact to interpret Hoare logic predicates by morphisms of type $X \to \mathcal{P}1$. They introduced the wppt using the join of $\mathcal{P}1$ in [9], and called its

composability *sequential compatibility*. Later they showed the wppt induced from the pair $(\mathcal{P}, \mathcal{T})$ is sequentially compatible if and only if the canonical morphism $\mathcal{T}\mathcal{P}1 \to \mathcal{P}1$ is an EM-algebra [36, Remark 11]. This fact bridges the work [9,36] and the following approach studied by Hasuo.

In [11], Hasuo introduced *PT situations* to construct composable wppts. A PT situation is a tuple of an order-enriched monad $(\mathcal{T}, \leq)$ (in a different sense from [9]), an object $\Omega \in \mathbb{C}$ and an EM $\mathcal{T}$-algebra τ on $\mathcal{T}\Omega$, satisfying certain conditions. Each PT situation induces a functor $\mathbb{P}^{Kl}(\tau) : (\mathbb{C}_{\mathcal{T}})^{op} \to \mathbf{Pos}$ embodying a wppt. The construction of $\mathbb{P}^{Kl}$ can be accounted for by the recipe given in Section 4. Let $(\mathcal{T}, \leq, \Omega, \tau)$ be a PT situation. It determines an ordered object $(\mathcal{T}\Omega, \leqslant)$ in $\mathbb{C}$, and τ is monotone [11, Definition 2.4]. Using the recipe in Section 4, it determines a Cartesian lifting $\mathcal{T}_\tau$ of $\mathcal{T}$ along $d^{\mathbb{C},\mathcal{T}\Omega} : \mathbb{C}/\mathcal{T}\Omega \to \mathbb{C}$. Then the wppt wp in the Cartesian Dijkstra structure $(d^{\mathbb{C},\mathcal{T}\Omega}, \mathcal{T}, \mathcal{T}_\tau)$ coincides with $\mathbb{P}^{Kl}(\tau)$. This demonstrates that the recipe in Section 4 generalizes the construction of $\mathbb{P}^{Kl}(\tau)$ given in [11]. Especially, our recipe does not demand order enrichment on $\mathcal{T}$, and allows us to take arbitrary ordered object (not limited to the form $\mathcal{T}\Omega$) as the basis of the lax slice construction. Examples 5.2, 5.4, 5.6 are benefited from this generalization.

In [13], Hino et al. introduced another construction of wppts using *relative algebras*. A relative algebra is a monad morphism of type $\mathcal{T} \to \mathbb{D}(\Omega^-, \Omega)$, where $\mathcal{T}$ is a **Set**-monad and $\mathbb{D}$ is a category with powers Ω^I [24, Section X.4 (4)] of a fixed $\mathbb{D}$-object Ω. They construct wppts by composing the comparison functor $\mathbf{Set}_{\mathcal{T}} \to \mathbf{Set}^{\mathcal{T}}$ [24, Theorem VI.3] and the functor $\mathbf{Set}^{\mathcal{T}} \to \mathbb{D}^{\mathrm{op}}$ induced by an relative algebra [13, Theorem 4.8]. When $\mathbb{D} = \mathbf{Pos}$, their wppts coincide with those of Cartesian liftings of $\mathcal{T}$ along the posetal fibration $d^{\mathbf{Set},\Omega}$ in Section 4. Therefore examples in Section 5-7 are also covered by the relative-algebra based wppts. On the other hand, materials in Section 4.2 and 4.3 are new compared to [13]. Particularly, the interaction between generic operations and wppts is the key in the semantic analysis of Kaminski's ert in Section 6.2.

The concept of monotone EM algebra appears in Voorneveld's program logic for a call-by-push-value language [41]. The logic can have multiple modalities to make assertions about effectful programs, and these modalities are interpreted by endofunctor algebras of the monad T_Σ of possibly infinite and partial Σ-terms. He introduces two conditions on such endofunctor algebras: *leaf-monotonicity* [41, Definition 4.5] and *sequentiality* [41, Definition 4.10]. They are respectively equivalent to monotonicity (Definition 4.3) and Eilenberg-Moore algebra axiom.

Our categorical framework is designed to provide an underlying semantic structure for the Hoare logic where formulas can only examine memory configurations. On the other hand, several extensions of program logics are introduced for the languages with exception [22,39,25]. In these extensions, formulas in the logic can also examine exceptions raised by programs — for instance, in [39], the Hoare logic is extended so that we can specify two postconditions for normal and exceptional termination. It is an interesting challenge to understand these extended Hoare logic within our fibrational framework.

Turning our eyes to functional programming languages, a series of recent papers [3,25,26] develop *Dijkstra monads* for the verification of effectful programs in dependent type theories. The main task of a Dijkstra monad is to take a type A and a specification $w \in \mathcal{W}A$ written in the language of a *specification monad* $\mathcal{W}$, and collect computations that satisfy w. In [25, Section 5], they establish an equivalence between Dijkstra monads and monad morphisms into ordered monads. Dijkstra monads intersect with our categorical wppts when the former arise from monad morphisms of type $\mathcal{T} \to \mathcal{W}^{\mathrm{pure}}$, where $\mathcal{W}^{\mathrm{pure}}$ is an ordered continuation monad (see [25, Section 4.1] for detail). In this situation, such a monad morphism bijectively corresponds to an EM monotone $\mathcal{T}$-algebra o over the return type of $\mathcal{W}^{\mathrm{pure}}$ [25, Section 4.4]. The corresponding Dijkstra monad then collects all the computations $c \in \mathcal{T}A$ such that the predicate transformer $\mathrm{wp}(-, c)$ induced from o is below the specification $w \in \mathcal{W}^{\mathrm{pure}}A$. This is the only relationship we know between Dijkstra monads and our fibrational theory of wppts, and we leave exploring further relationships between them as future work.

10 Conclusion

We have presented a fibrational and monadic semantics of Hoare triples and weakest precondition predicate transformers. The key fact is that the composability of wppts is equivalent to the Cartesian-ness of the monad lifting. We next studied the Cartesian liftings of monads along domain fibrations; they bijectively correspond to EM monotone algebras, giving a fibrational view of [11]. Despite examples presented here being within the framework of [11,13], we studied new examples, in which we have revealed the monads behind wppt-like operators used to compute expected run-time [16] and higher moments [21].

References

[1] Samson Abramsky, Simon Gay, and Rajagopal Nagarajan. *Specification Structures and propositions-as-types for concurrency*, pages 5–40. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

[2] Alejandro Aguirre, Gilles Barthe, Justin Hsu, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. Kantorovich continuity of probabilistic programs. *CoRR*, abs/1901.06540, 2019.

[3] Danel Ahman, Catalin Hritcu, Kenji Maillard, Guido Martínez, Gordon D. Plotkin, Jonathan Protzenko, Aseem Rastogi, and Nikhil Swamy. Dijkstra monads for free. In Giuseppe Castagna and Andrew D. Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pages 515–529. ACM, 2017.

[4] Jon Beck. Distributive laws. In B. Eckmann, editor, *Seminar on Triples and Categorical Homology Theory*, pages 119–140, Berlin, Heidelberg, 1969. Springer Berlin Heidelberg.

[5] Filippo Bonchi, Barbara König, and Daniela Petrisan. Up-to techniques for behavioural metrics via fibrations. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency Theory, CONCUR 2018, September 4-7, 2018, Beijing, China*, volume 118 of *LIPIcs*, pages 17:1–17:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[6] Edsger W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Commun. ACM*, 18(8):453–457, 1975.

[7] Andrzej Filinski. *Controlling Effects*. PhD thesis, Carnegie Mellon University, 1996.

[8] Andrzej Filinski. On the relations between monadic semantics. *Theor. Comput. Sci.*, 375(1-3):41–75, 2007.

[9] Sergey Goncharov and Lutz Schröder. A relatively complete generic hoare logic for order-enriched effects. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 273–282. IEEE Computer Society, 2013.

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

[11] Ichiro Hasuo. Generic weakest precondition semantics from monads enriched with order. *Theoretical Computer Science*, 604:2 – 29, 2015. Coalgebraic Methods in Computer Science.

[12] Claudio Hermida. *Fibrations, Logical Predicates and Indeterminants*. PhD thesis, University of Edinburgh, 1993.

[13] Wataru Hino, Hiroki Kobayashi, Ichiro Hasuo, and Bart Jacobs. Healthiness from duality. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 682–691. ACM, 2016.

[14] Bart Jacobs. *Categorical Logic and Type Theory*. Number 141 in Studies in Logic and the Foundations of Mathematics. North Holland, Amsterdam, 1999.

[15] André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3):447–468, 1996.

[16] Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Federico Olmedo. Weakest precondition reasoning for expected run-times of probabilistic programs. In Peter Thiemann, editor, *Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9632 of *Lecture Notes in Computer Science*, pages 364–389. Springer, 2016.

[17] Shin-ya Katsumata. A semantic formulation of $\top\top$-lifting and logical predicates for computational metalanguage. In *Proc. CSL '05*, volume 3634 of *LNCS*, pages 87–102. Springer, 2005.

[18] Shin-ya Katsumata, Tetsuya Sato, and Tarmo Uustalu. Codensity lifting of monads and its dual. *Logical Methods in Computer Science*, Volume 14, Issue 4, October 2018.

[19] Bartek Klin. A coalgebraic approach to process equivalence and a coinduction principle for traces. *Electronic Notes in Theoretical Computer Science*, 106:201 – 218, 2004. Proceedings of the Workshop on Coalgebraic Methods in Computer Science (CMCS).

[20] Anders Kock. Strong functors and monoidal monads. *Archiv der Mathematik*, 23:113–120, 1970.

[21] Satoshi Kura, Natsuki Urabe, and Ichiro Hasuo. Tail probabilities for randomized program runtimes via martingales for higher moments. In Tomáš Vojnar and Lijun Zhang, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 135–153, Cham, 2019. Springer International Publishing.

[22] K. Rustan M. Leino and Jan L. A. van de Snepscheut. Semantics of exceptions. In Ernst-Rüdiger Olderog, editor, *Programming Concepts, Methods and Calculi, Proceedings of the IFIP TC2/WG2.1/WG2.2/WG2.3 Working Conference on Programming Concepts, Methods and Calculi (PROCOMET '94) San Miniato, Italy, 6-10 June, 1994*, volume A-56 of *IFIP Transactions*, pages 447–466. North-Holland, 1994.

[23] Tom Leinster. *Higher Operads, Higher Categories*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2004.

[24] Saunders MacLane. *Categories for the Working Mathematician (Second Edition)*, volume 5 of *Graduate Texts in Mathematics*. Springer, 1998.

[25] Kenji Maillard, Danel Ahman, Robert Atkey, Guido Martínez, Catalin Hritcu, Exequiel Rivas, and Éric Tanter. Dijkstra monads for all. *Proc. ACM Program. Lang.*, 3(ICFP):104:1–104:29, 2019.

[26] Kenji Maillard, Catalin Hritcu, Exequiel Rivas, and Antoine Van Muylder. The next 700 relational program logics. *Proc. ACM Program. Lang.*, 4(POPL):4:1–4:33, 2020.

[27] Ernest Manes and Philip Mulry. Monad compositions I: general constructions and recursive distributive laws. *Theory and Applications of Categories*, 18:172–208, 04 2007.

[28] Ursula Martin, Erik A. Mathiesen, and Paulo Oliva. Hoare logic in the abstract. In Zoltán Ésik, editor, *Computer Science Logic*, pages 501–515, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[29] Annabelle McIver and Carroll Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer, New York, NY, 2005.

[30] 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 SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pages 3–16. ACM, 2015.

[31] Paul-André Melliès and Noam Zeilberger. Isbell duality for refinement types. *CoRR*, abs/1501.05115, 2015.

[32] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.

[33] Andrew M. Pitts. Tripos theory in retrospect. *Electronic Notes in Theoretical Computer Science*, 23(1):111 – 127, 1999. Tutorial Workshop on Realizability Semantics and Applications (associated to FLoC'99, the 1999 Federated Logic Conference).

[34] Gordon Plotkin and John Power. Semantics for algebraic operations, 2001.

[35] Gordon Plotkin and John Power. Notions of computation determine monads. In Mogens Nielsen and Uffe Engberg, editors, *Foundations of Software Science and Computation Structures*, pages 342–356, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[36] Christoph Rauch, Sergey Goncharov, and Lutz Schröder. Generic hoare logic for order-enriched effects with exceptions. In Phillip James and Markus Roggenbach, editors, *Recent Trends in Algebraic Development Techniques - 23rd IFIP WG 1.3 International Workshop, WADT 2016, Gregynog, UK, September 21-24, 2016, Revised Selected Papers*, volume 10644 of *Lecture Notes in Computer Science*, pages 208–222. Springer, 2016.

[37] Tetsuya Sato. A probabilistic monad with nondeterminism for coalgebraic trace semantics. tetsuya sato. In *CALCO Young Researchers Workshop CALCO-jnr 2011 selected papers*. University of Southampton, 2011.

[38] Lutz Schröder. Expressivity of coalgebraic modal logic: The limits and beyond. *Theoretical Computer Science*, 390(2):230 – 247, 2008. Foundations of Software Science and Computational Structures.

[39] Emil Sekerinski. *Dependability and Computer Engineering: Concepts for Software-Intensive Systems*, chapter Exceptions for Dependability, pages 11–35. IGI Global, 2012.

[40] Daniele Varacca and Glynn Winskel. Distributing probability over non-determinism. *Mathematical Structures in Computer Science*, 16(1):87–113, 2006.

[41] Niels F. W. Voorneveld. Quantitative logics for equivalence of effectful programs. In Barbara König, editor, *Proceedings of the Thirty-Fifth Conference on the Mathematical Foundations of Programming Semantics, MFPS 2019, London, UK, June 4-7, 2019*, volume 347 of *Electronic Notes in Theoretical Computer Science*, pages 281–301. Elsevier, 2019.

[42] Whiskering. https://ncatlab.org/nlab/show/whiskering.