Dijkstra and Hoare Monads in Monadic Computation

Bart Jacobs

Institute for Computing and Information Sciences (iCIS),
Radboud University Nijmegen, The Netherlands.
Webaddress: www.cs.ru.nl/B.Jacobs

24 March 2015

Abstract

The Dijkstra and Hoare monads have been introduced recently for capturing weakest precondition computations and computations with pre- and post-conditions, within the context of program verification, supported by a theorem prover. Here we give a more general description of such monads in a categorical setting. We first elaborate the recently developed view on program semantics in terms of a triangle of computations, state transformers, and predicate transformers. Instantiating this triangle for different computational monads T shows how to define the Dijkstra monad associated with T, via the logic involved.

Subsequently we give abstract definitions of the Dijkstra and Hoare monad, parametrised by a computational monad. These definitions presuppose a suitable (categorical) predicate logic, defined on the Kleisli category of the underlying monad. When all this structure exists, we show that there are maps of monads (Hoare) $\Rightarrow$ (State) $\Rightarrow$ (Dijkstra), all parametrised by a monad T.

1 Introduction

A monad is a categorical concept that is surprisingly useful in the theory of computation. On the one hand it describes a form of computation (such as partial, non-deterministic, or probabilistic), and on the other hand it captures various algebraic structures. Technically, the computations are maps in the Kleisli category of the monad, whereas the algebraic structures are described via the category of so-called Eilenberg-Moore algebras. The Kleisli approach has become common in program semantics and functional programming (notably in the language Haskell), starting with the seminal paper [26]. The algebraic structure captured by the monad exists on these programs (as Kleisli

maps), technically because the Kleisli category is enriched over the category of algebras (for suitable monads).

Interestingly, the range of examples of monads has been extended recently from computation to program logic. So-called Hoare monads [27,31] and Dijkstra monads [30] have been defined in a systematic approach to program verification. Via these monads one describes not only a program but also the associated correctness assertions. These monads have been introduced in the language of a theorem prover, but have not been investigated systematically from a categorical perspective. Here we do so not only for the Dijkstra monad (like in [16]), but also for the Hoare monad. We generalise the original definition from [30,27,31] and show that 'Dijkstra' and 'Hoare' monads $\mathfrak{D}_T$ and $\mathfrak{H}_T$ can be associated with various well-known monads T that are used for modeling computations.

Since the Dijkstra and Hoare monads combine both semantics and logic of programs, we need to look at these two areas in a unified manner. From previous work [15] (see also [14]) a view on program semantics and logic emerged involving a triangle of the form:

$$\mathbf{Log}^{\mathrm{op}} = \left(\begin{array}{c}\text{predicate}\\ \text{transformers}\end{array}\right) \underset{\longleftarrow}{\overset{\longrightarrow}{\top}} \left(\begin{array}{c}\text{state}\\ \text{transformers}\end{array}\right) \qquad \begin{array}{ccc} & \overset{\mathit{Pred}}{\nwarrow} \quad \overset{\mathit{Stat}}{\nearrow} & \\ \big(\text{computations}\big) & & \end{array} \tag{1}$$

The three nodes in this diagram represent categories of which only the morphisms are described. The arrows between these nodes are functors, where the two arrows $\rightleftarrows$ at the top form an adjunction. The two triangles involved should commute. In the case where two up-going 'predicate' and 'state' functors *Pred* and *Stat* in (1) are full and faithful, we have three equivalent ways of describing computations. On morphisms, the predicate functor yields what is called substitution in categorical logic, but what amounts to a weakest precondition operation in program semantics. The upper category on the left is of the form $\mathbf{Log}^{\mathrm{op}}$, where **Log** is some category of logical structures. The opposite category $(-)^{\mathrm{op}}$ is needed because predicate transformers operate in the reverse direction, taking a postcondition to a precondition.

In a setting of quantum computation this translation back-and-forth $\rightleftarrows$ in (1) is associated with the different approaches of Heisenberg (logic-based, working backwards) and Schrödinger (state-based, working forwards), see *e.g.* [9]. In certain cases the adjunction $\rightleftarrows$ forms — or may be restricted to — an equivalence of categories, yielding a duality situation. It shows the importance of duality theory in program semantics and logic; this topic has a long history,

going back to [1].

Almost all of our examples of computations are given by maps in a Kleisli category of a monad. In this monadic setting, the right-hand-side of the diagram (1) is the full and faithful 'comparison' functor $\mathcal{K}\ell(T) \to \mathcal{EM}(T)$, for the monad T at hand. This functor embeds the Kleisli category $\mathcal{K}\ell(T)$ in the category $\mathcal{EM}(T)$ of (Eilenberg-Moore) algebras. The left-hand-side takes the form $\mathcal{K}\ell(T) \to \mathbf{Log}^{\mathrm{op}}$, and forms an indexed category (or, if you like, a fibration), and thus a categorical model of predicate logic. The monad T captures computations as maps in its Kleisli category. And via the predicate logic in (1) the 'Dijkstra' and 'Hoare' monads are defined.

One open problem in this area is how to obtain an appropriate categorical logic **Log** for a monad T, yielding a triangle (1). In Sections 6 and 7 we side-step this problem by axiomatising the required categorical logic that is needed for the Dijkstra and Hoare monads. Specifically, we describe the properties that a functor (indexed category) $\mathit{Pred}\colon \mathcal{K}\ell(T) \to \mathbf{Log}^{\mathrm{op}}$ should satisfy so that one can define the Dijkstra and Hoare monad, written as $\mathfrak{D}_T$ and $\mathfrak{H}_T$ respectively, associated with T. It turns out that the Dijkstra monad requires only mild logical properties, whereas the Hoare monad requires much stronger properties. Once they are satisfied we show that there maps of monads:

$$\begin{array}{ccccc} \left(\begin{array}{c}\text{Hoare}\\ \text{monad } \mathfrak{H}_T\end{array}\right) & \Longrightarrow & \left(\begin{array}{c}\text{state}\\ \text{monad } \mathfrak{S}_T\end{array}\right) & \Longrightarrow & \left(\begin{array}{c}\text{Dijkstra}\\ \text{monad } \mathfrak{D}_T\end{array}\right) \\ \{P\}h\{Q\} & \longmapsto & h & \longmapsto & wp(h,-) \end{array}$$

These maps describe some fundamental relations in the semantics of programs: a Hoare triple $\{P\}h\{Q\}$ is first sent to the program h, as element of the state monad $\mathfrak{S}_T = T(S \times -)^S$ associated with the monad T. In a second step this program h is sent to the weakest precondition operation $wp(h, -)$, mapping a postcondition to a precondition.

We assume that the reader is familiar with the basic concepts of category theory, especially with the theory of monads. The organisation of the paper is as follows: the first three sections 2 – 4 elaborate instances of the triangle (1) for non-deterministic, linear & probabilistic, and quantum computation. Subsequently, Section 5 shows how to obtain the Dijkstra monads for the different (concrete) monad examples, and proves in these cases that weakest precondition computation forms a map of monads. This approached is axiomatised in Section 6, starting from a suitable categorical predicate logic. In a similar, but more restricted, axiomatic setting the Hoare monad is defined in Section 7. Finally, Section 8 wraps up with some concluding remarks.

1.1 Notation for monads

We assume the reader is familiar with the notion of monad, see *e.g.* [25,2]. We shall restrict ourselves to monads $T = (T, \eta, \mu)$ on the category **Sets** of sets and functions. We write $\mathcal{K}\ell(T)$ for the Kleisli category of T, with sets X as objects, and 'Kleisli maps' $X \to Y$ given by functions $X \to T(Y)$. We write $J\colon \mathbf{Sets} \to \mathcal{K}\ell(T)$ for the functor given by $J(X) = X$ and $J(f) = \eta \circ f$. Kleisli maps of the form $J(f)$ are often called 'pure'. We use a fat dot $\bullet$ for composition in $\mathcal{K}\ell(T)$, to distinguish it from ordinary composition $\circ$. We recall that $g \bullet f = \mu \circ T(g) \circ f$, and $J(g) \bullet J(f) = J(g \circ f)$.

The Eilenberg-Moore category of the monad T is written as $\mathcal{EM}(T)$. Its objects are 'algebras' $a\colon T(X) \to X$ satisfying $a \circ \eta = \mathrm{id}$ and $a \circ T(a) = a \circ \mu$. There is a canonical functor $\mathit{Stat}\colon \mathcal{K}\ell(T) \to \mathcal{EM}(T)$, sending a set X to the free algebra $\mu\colon T^2(X) \to T(X)$ and a map $f\colon X \to T(Y)$ to the 'Kleisli extension' $f_* = \mu \circ T(f)\colon T(X) \to T(Y)$.

A monad T on **Sets** is automatically strong. There are 'strength' natural transformations $\mathsf{st}_1\colon T(X) \times Y \to T(X \times Y)$ and $\mathsf{st}_2\colon X \times T(Y) \to T(X \times Y)$ defined as:

$$\mathsf{st}_1(u, y) = T(\lambda x.\, \langle x, y\rangle)(u) \qquad \text{and} \qquad \mathsf{st}_2(x, v) = T(\lambda y.\, \langle x, y\rangle)(v).$$

These strength maps are related via swapping $X \times Y \cong Y \times X$, and satisfy $T(\pi_1) \circ \mathsf{st}_1 = \pi_1$ and $T(\pi_2) \circ \mathsf{st}_2 = \pi_2$. Moreover, they interact appropriately with the unit and multiplication of the monad, see *e.g.* [17].

2 Non-deterministic and partial computation

The powerset operation $\mathcal{P}(X) = \{U \mid U \subseteq X\}$ yields a monad $\mathcal{P}\colon \mathbf{Sets} \to \mathbf{Sets}$ with unit $\eta = \{-\}$ given by singletons and multiplication $\mu = \bigcup$ by union. The associated Kleisli category $\mathcal{K}\ell(\mathcal{P})$ is the category of sets and non-deterministic functions $X \to \mathcal{P}(Y)$, which may be identified with relations $R \subseteq X \times Y$. The category $\mathcal{EM}(\mathcal{P})$ of (Eilenberg-Moore) algebras is the category $\mathbf{CL}_\bigvee$ of complete lattices and join-preserving functions. In this situation diagram (1) takes the form:

$$\begin{array}{ccc} \left(\mathbf{CL}_\bigwedge\right)^{\mathrm{op}} & \overset{\cong}{\rightleftarrows} & \mathbf{CL}_\bigvee = \mathcal{EM}(\mathcal{P}) \\ & {\scriptstyle \mathit{Pred}} \nwarrow \quad \nearrow {\scriptstyle \mathit{Stat}} & \\ & \mathcal{K}\ell(\mathcal{P}) & \end{array} \tag{2}$$

where $\mathbf{CL}_{\bigwedge}$ is the category of complete lattices and meet-preserving maps. The isomorphism $\cong$ arises because each join-preserving map between complete lattices corresponds to a meet-preserving map in the other direction. The upgoing 'state' functor *Stat* on the right is the standard full and faithful functor from the Kleisli category of a monad to its category of algebras. The predicate functor $\mathit{Pred}\colon \mathcal{K}\ell(\mathcal{P}) \to (\mathbf{CL}_{\bigwedge})^{\mathrm{op}}$ on the left sends a set X to the powerset $\mathcal{P}(X)$ of predicates/subsets, as complete lattices; a Kleisli map $f\colon X \to \mathcal{P}(Y)$ yields a map:

$$\mathcal{P}(Y) \xrightarrow{f^* = \mathit{Pred}(f)} \mathcal{P}(X) \qquad \text{given by} \qquad (Q \subseteq Y) \longmapsto \{x \mid f(x) \subseteq Q\}. \tag{3}$$

In categorical logic, this map $\mathit{Pred}(f)$ is often written as f^*, and called a substitution functor. In modal logic one may write it as $\Box_f$. In the current context we also write it as $\mathit{wp}(f)$, since it forms the weakest precondition operation for f, see [5]. Clearly, it preserves arbitrary meets (intersections). It is not hard to see that the triangle (2) commutes.

Interestingly, the diagram (2) involves additional structure on homsets. If we have a collection of parallel maps f_i in $\mathcal{K}\ell(\mathcal{P})$, we can take their (pointwise) join $\bigvee_{i\in I} f_i$. Pre- and post-composition preserves such joins. This means that the Kleisli category $\mathcal{K}\ell(\mathcal{P})$ is enriched over the category $\mathbf{CL}_{\bigvee}$. The category $\mathbf{CL}_{\bigvee}$ is monoidal closed, and thus enriched over itself. Also the category $(\mathbf{CL}_{\bigwedge})^{\mathrm{op}}$ is enriched over $\mathbf{CL}_{\bigvee}$, with joins given by pointwise intersections. Further, the functors in (2) are enriched over $\mathbf{CL}_{\bigvee}$, which means that they preserve these joins on posets. In short, the triangle is a diagram in the category of categories enriched over $\mathbf{CL}_{\bigvee}$. In particular, the predicate functor is enriched, which amounts to the familiar law for non-deterministic choice in weakest precondition reasoning: $\mathit{wp}(\bigvee_i f_i) = \bigwedge_i \mathit{wp}(f_i)$.

A less standard monad for non-determinism is the *ultrafilter* monad $\mathcal{U}\colon \mathbf{Sets} \to \mathbf{Sets}$. A convenient way to describe it, at least in the current setting, is:

$$\mathcal{U}(X) = \mathbf{BA}\Big(\mathcal{P}(X), 2\Big) = \{f\colon \mathcal{P}(X) \to 2 \mid f \text{ is a map of Boolean algebras }\}.$$

For a finite set X one has $X \xRightarrow{\cong} \mathcal{U}(X)$.

A famous result of [23] says that the category of algebras of $\mathcal{U}$ is the category $\mathbf{CH}$ of compact Hausdorff spaces (and continuous functions). It yields the following triangle.

$$\begin{array}{ccc} \mathbf{BA}^{\mathrm{op}} & \overset{\mathit{Spec}=\mathit{Hom}(-,2)}{\underset{\mathit{Clopen}}{\top}} & \mathbf{CH} = \mathcal{EM}(\mathcal{U}) \\ & {\scriptstyle \mathit{Pred}} \searrow \quad \nearrow {\scriptstyle \mathit{Stat}} & \\ & \mathcal{K}\ell(\mathcal{U}) & \end{array} \tag{4}$$

The predicate functor *Pred* sends a set X to the Boolean algebra $\mathcal{P}(X)$ of

subsets of X. For a map $f\colon X \to \mathcal{U}(Y)$ we get $f^*\colon \mathcal{P}(Y) \to \mathcal{P}(X)$ by $f^*(Q) = \{x \mid f(x)(Q) = 1\}$. This functor *Pred* is full and faithful, almost by construction.

The precise enrichment in this case is unclear. Enrichment over (compact Hausdorff) spaces, if present, is not so interesting because it does not provide algebraic structure on computations.

We briefly look at the *lift* (or 'maybe') monad $\mathcal{L}\colon \mathbf{Sets} \to \mathbf{Sets}$, given by $\mathcal{L}(X) = 1 + X$. Its Kleisli category $\mathcal{K}\ell(\mathcal{L})$ is the category of sets and partial functions. And its (equivalent) category of algebra $\mathcal{EM}(\mathcal{L})$ is the category $\mathbf{Sets}_\bullet$ of pointed sets, $(X, \bullet_X)$, where $\bullet_X \in X$ is a distinguished element; morphisms in $\mathbf{Sets}_\bullet$ are 'strict', in the sense that they preserve such points. There is then a situation:

$$\begin{array}{ccc} (\mathbf{ACL}_{\bigvee_\bullet,\wedge})^{\mathrm{op}} & \overset{\top}{\rightleftarrows} & \mathbf{Sets}_\bullet = \mathcal{EM}(\mathcal{L}) \\ & {\scriptstyle Pred}\nwarrow \quad \nearrow{\scriptstyle Stat} & \\ & \mathcal{K}\ell(\mathcal{L}) & \end{array} \tag{5}$$

We call a complete lattice *atomic* if (1) each element is the join of atoms below it, and (2) binary meets $\wedge$ distribute over arbitrary joins $\bigvee$. Recall that an atom a is a non-bottom element satisfying $x < a \Rightarrow x = \bot$. We write $At(L) \subseteq L$ for the subset of atoms. In such an atomic lattice atoms a are completely join-irreducible: for a non-empty index set I, if $a \leq \bigvee_{i\in I} x_i$ then $a \leq x_i$ for some $i \in I$.

The category $\mathbf{ACL}_{\bigvee_\bullet,\wedge}$ contains atomic complete lattices, with maps preserving non-empty joins (written as $\bigvee_\bullet$) and binary meets $\wedge$. Each Kleisli map $f\colon X \to \mathcal{L}(Y) = \{\bot\} \cup Y$ yields a substitution map $f^*\colon \mathcal{P}(Y) \to \mathcal{P}(X)$ by $f^*(Q) = \{x \mid \forall y.\, f(x) = y \Rightarrow Q(y)\}$. This f^* preserves $\wedge$ and non-empty joins $\bigvee_\bullet$. Notice that $f^*(\emptyset) = \{x \mid f(x) = \bot\}$, which need not be empty.

The adjunction $(\mathbf{ACL}_{\bigvee_\bullet,\wedge})^{\mathrm{op}} \rightleftarrows \mathbf{Sets}_\bullet$ amounts to a bijective correspondence:

$$\frac{L \xrightarrow{f} \mathcal{P}(X - \bullet)}{X \xrightarrow{g} \{\bot\} \cup At(L)} \qquad \begin{array}{l} \text{in } (\mathbf{ACL}_{\bigvee_\bullet,\wedge})^{\mathrm{op}} \\ \text{in } \mathbf{Sets}_\bullet \end{array}$$

This correspondence works as follows. Given $f\colon L \to \mathcal{P}(X - \bullet)$ notice that $X = f(\top) = f(\bigvee At(L)) = \bigcup_{a\in At(L)} f(a)$. Hence for each $x \in X$ there is an atom a with $x \in f(a)$. We define $\overline{f}\colon X \to \{\bot\} \cup At(L)$ as:

$$\overline{f}(x) = \begin{cases} a & \text{if } x \in f(a) - f(\bot) \\ \bot & \text{otherwise.} \end{cases}$$

This is well-defined: if x is both in $f(a) - f(\bot)$ and in $f(a') - f(\bot)$, for $a \neq a'$, then $x \in (f(a) \cap f(a')) - f(\bot) = f(a \wedge a') - f(\bot) = f(\bot) - f(\bot) = \emptyset$.

In the other direction, given $g\colon X \to \{\bot\} \cup At(L)$, define for $y \in L$,

$$\overline{g}(y) = \{x \in X \mid \exists a \in At(L).\, a \leq y \text{ and } g(x) = a\} \cup \{x \in X - \bullet \mid g(x) = \bot\}.$$

It is not hard to see that this yields a commuting triangle (5), and that the (upgoing) functors are full and faithful.

3 Linear and (sub)convex computation

We sketch two important sources for linear and (sub)convex structures.

(1) If A is a matrix, say over the real numbers $\mathbb{R}$, then the set of solution vectors v of the associated homogeneous equation $Av = 0$ forms a linear space: it is closed under finite additions and scalar multiplication. For a fixed vector $b \neq 0$, the solutions v of the non-homogeneous equation $Ax = b$ form a convex set: it is closed under convex combinations $\sum_i r_i v_i$ of solutions v_i and 'probability' scalars $r_i \in [0, 1]$ with $\sum_i r_i = 1$. Finally, for $b \geq 0$, the solutions v to the inequality $Av \leq b$ are closed under subconvex combinations $\sum_i r_i v_i$ with $\sum_i r_i \leq 1$. These examples typically occur in linear programming.

(2) If V is a vector space of some sort, we can consider the space of linear functions $f\colon V \to \mathbb{R}$ to the real (or complex) numbers. This space is linear again, via pointwise definitions. Now if V contains a unit 1, we can impose an additional requirement that such functions $f\colon V \to \mathbb{R}$ are 'unital', *i.e.* satisfy $f(1) = 1$. This yields a convex set of functions, where $\sum_i r_i f_i$ again preserves the unit, if $\sum_i r_i = 1$. If we require only $0 \leq f(1) \leq 1$, making f 'subunital', we get a subconvex set. These requirements typically occur in a setting of probability measures.

Taking (formal) linear and (sub)convex combinations over a set yields the structure of a monad. We start by recalling the definitions of these (three) monads, namely the multiset monad $\mathcal{M}_R$, the distribution monad $\mathcal{D}$, and the subdistribution monad $\mathcal{D}_{\leq 1}$, see [14] for more details. A semiring is given by a set R which carries a commutative monoid structure $(+, 0)$, and also another monoid structure $(\cdot, 1)$ which distributes over $(+, 0)$. As is well-known [13], each such semiring R gives rise to a *multiset* monad $\mathcal{M}_R\colon \mathbf{Sets} \to \mathbf{Sets}$, where:

$$\mathcal{M}_R(X) = \{\varphi\colon X \to R \mid \mathit{supp}(\varphi) \text{ is finite}\},$$

where $\mathit{supp}(\varphi) = \{x \in X \mid \varphi(x) \neq 0\}$ is the support of φ. Such $\varphi \in \mathcal{M}_R(X)$ may also be written as finite formal sum $\varphi = \sum_i s_i |x_i\rangle$ where $\mathit{supp}(\varphi) =$

$\{x_1, \ldots, x_n\}$ and $s_i = \varphi(x_i) \in R$ is the multiplicity of $x_i \in X$. The 'ket' notation $|x\rangle$ for $x \in X$ is just syntactic sugar. The unit of the monad is given by $\eta(x) = 1|x\rangle$ and its multiplication by $\mu(\sum_i s_i|\varphi_i\rangle) = \sum_x (\sum_i s_i \cdot \varphi_i(x))|x\rangle$.

The *distribution* monad $\mathcal{D}\colon \mathbf{Sets} \to \mathbf{Sets}$ is defined similarly. It maps a set X to the set of finite formal convex combinations over X, as in:

$$\begin{aligned}\mathcal{D}(X) &= \{\varphi\colon X \to [0,1] \mid \mathit{supp}(\varphi) \text{ is finite, and } \textstyle\sum_x \varphi(x) = 1\} \\ &= \{r_1|x_1\rangle + \cdots + r_n|x_n\rangle \mid x_i \in X, r_i \in [0,1] \text{ with } \textstyle\sum_i r_i = 1\}.\end{aligned}$$

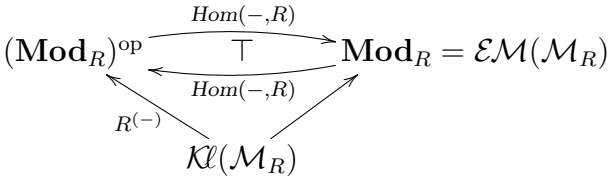
The unit η and multiplication μ for $\mathcal{D}$ are as for $\mathcal{M}_R$. We consider another variation, namely the *subdistribution* monad $\mathcal{D}_{\leq 1}$, where $\mathcal{D}_{\leq 1}(X)$ contains the formal *subconvex* combinations $\sum_i r_i|x_i\rangle$ where $\sum_i r_i \leq 1$. It has the same unit and multiplication as $\mathcal{D}$.

These three monads $\mathcal{M}_R, \mathcal{D}$ and $\mathcal{D}_{\leq 1}$ are used to capture different kinds of computation, in the style of [26]. Maps (coalgebras) of the form $c\colon X \to \mathcal{M}_R(X)$ capture 'multi-computations', which can be written in transition notation as $x \xrightarrow{r} x'$ if $c(x)(x') = r$. This label $r \in R$ can represent the time or cost of a transition. Similarly, the monads $\mathcal{D}$ and $\mathcal{D}_{\leq 1}$ capture probabilistic computation: for coalgebras $c\colon X \to \mathcal{D}(X)$ or $c\colon X \to \mathcal{D}_{\leq 1}(X)$ we can write $x \xrightarrow{r} x'$ where $c(x)(x') = r \in [0,1]$ describes the probability of the transition $x \to x'$.

The category $\mathcal{EM}(\mathcal{M}_R)$ of (Eilenberg-Moore) algebras of the multiset monad $\mathcal{M}_R$ contains the *modules* over the semiring R. Such a module is given by a commutative monoid $M = (M, +, 0)$ together with a scalar multiplication $R \times M \to M$ which preserves $(+, 0)$ in both arguments. More abstractly, if we write **CMon** for the category of commutative monoids, then the semiring R is a monoid in **CMon**, and the category $\mathbf{Mod}_R = \mathcal{EM}(\mathcal{M}_R)$ of modules over R is the category $Act_R(\mathbf{CMon})$ of R-actions $R \otimes M \to M$ in **CMon**, see also [25, VII§4]. For instance, for the semiring $R = \mathbb{N}$ of natural numbers we obtain $\mathbf{CMon} = \mathcal{EM}(\mathcal{M}_{\mathbb{N}})$ as associated category of algebras; for $R = \mathbb{R}$ or $R = \mathbb{C}$ we obtain the categories $\mathbf{Vect}_{\mathbb{R}}$ or $\mathbf{Vect}_{\mathbb{C}}$ of vector spaces over real or complex numbers; and for the Boolean semiring $R = 2 = \{0, 1\}$ we get the category **JSL** of join semi-lattices, since $\mathcal{M}_2$ is the finite powerset monad.

We shall write $\mathbf{Conv} = \mathcal{EM}(\mathcal{D})$ for the category of *convex* sets. These are sets X in which for each formal convex sum $\sum_i r_i|x_i\rangle$ there is an actual convex sum $\sum_i r_i x_i \in X$. Morphisms in **Conv** preserve such convex sums, and are often called affine functions. A convex set can be defined alternatively as a barycentric algebra [29], see [12] for the connection. Similarly, we write $\mathbf{Conv}_{\leq 1} = \mathcal{EM}(\mathcal{D}_{\leq 1})$ for the category of *subconvex* sets, in which subconvex sums exist.

For linear 'multi' computation the general diagram (1) takes the following form.

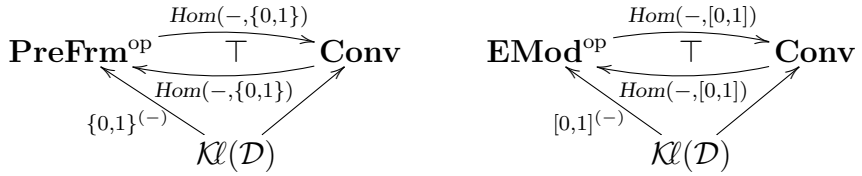


(6)

The adjunction $(\mathbf{Mod}_R)^{\mathrm{op}} \rightleftarrows \mathbf{Mod}_R$ is given by the correspondence between homomorphisms $M \to (N \multimap R)$ and $N \to (M \multimap R)$, where $\multimap$ is used for linear function space. The predicate functor $R^{(-)} \colon \mathcal{K}\ell(\mathcal{M}_R) \to (\mathbf{Mod}_R)^{\mathrm{op}}$ sends a set X to the module R^X of functions $X \to R$, with pointwise operations. A Kleisli map $f \colon X \to \mathcal{M}_R(Y)$ yields a map of modules $f^* \colon R^Y \to R^X$ by $f^*(q)(x) = \sum_y q(y) \cdot f(x)(y)$. Like before, this $f^*(q)$ may be understood as the weakest precondition of the postcondition q. In one direction the triangle commutes: $\mathit{Hom}(\mathcal{M}_R(X), R) \cong \mathbf{Sets}(X, R) = R^X$ since $\mathcal{M}_R(X)$ is the free module on X. Commutation in the other direction, that is $\mathit{Hom}(R^X, R) \cong \mathcal{M}_R(X)$ holds for finite sets X. Hence in order to get a commuting triangle we should restrict to the full subcategory $\mathcal{K}\ell_{\mathbb{N}}(\mathcal{M}_R) \hookrightarrow \mathcal{K}\ell(\mathcal{M}_R)$ with objects $n \in \mathbb{N}$, considered as n-element set.

Now let R be a *commutative* semiring. The triangle (6) is then a diagram enriched over $\mathbf{Mod}_R$: the categories, functors, and natural transformations involved are all enriched. Indeed, if the semiring R is commutative, then so is the monad $\mathcal{M}_R$, see *e.g.* [14]; this implies that $\mathbf{Mod}_R$ is monoidal closed, and in particular enriched over itself. Similarly, the Kleisli category $\mathcal{K}\ell(\mathcal{M}_R)$ is then enriched over $\mathbf{Mod}_R$.

In the probabilistic case one can choose to use a logic with classical predicates (subsets, or characteristic functions) $\{0,1\}^X$ or 'fuzzy predicates' $[0,1]^X$. These options are captured in the following two triangles.



(7)

The adjunctions both come from [14]. The one on the left is investigated further in [24]. It uses the category **PreFrm** of preframes: posets with directed joins and finite meets, distributing over these joins, see [20]. Indeed, for a Kleisli map $f \colon X \to \mathcal{D}(Y)$ we have a substitution functor $f^* \colon \mathcal{P}(Y) \to \mathcal{P}(X)$ in **PreFrm** given by $f^*(Q) = \mathit{wp}(f)(Q) = \{x \in X \mid \mathit{supp}(f(x)) \subseteq Q\}$. This f^* preserves directed joins because the support of $f(x) \in \mathcal{D}(Y)$ is finite.

The homsets $\mathbf{PreFrm}(X, Y)$ of preframe maps $X \to Y$ have finite meets $\wedge, \top$, which can be defined pointwise. As a result, these homsets are convex sets, in a trivial manner: a sum $\sum_i r_i h_i$ is interpreted as $\bigwedge_i h_i$, where we implicitly

assume that $r_i > 0$ for each i. With this in mind one can check that the triangle on the left in (7) is enriched over **Conv**. It yields the rule $wp(\sum_i r_i f_i)(Q) = \bigcap_i wp(f_i)(Q)$.

The situation on the right in (7) requires more explanation. We sketch the essentials. A *partial commutative monoid* (PCM) is a given by a set M with a partial binary operation $\oslash\colon M \times M \to M$ which is commutative and associative, in a suitable sense, and has a zero element $0 \in M$. One writes $x \perp y$ if $x \oslash y$ is defined. A morphism $f\colon M \to N$ of PCMs satisfies: $x \perp x'$ implies $f(x) \perp f(x')$, and then $f(x \oslash x') = f(x) \oslash f(x')$. This yields a category which we shall write as **PCMon**.

The unit interval $[0, 1]$ is clearly a PCM, with $r \oslash r'$ defined and equal to $r + r'$ if $r + r' \leq 1$. With its multiplication operation this $[0, 1]$ is a monoid in the category **PCMon**, see [18] for details. We define a category $\mathbf{PCMod} = Act_{[0,1]}(\mathbf{PCMon})$ of *partial commutative modules*; its objects are PCMs M with an action $[0, 1] \times M \to M$, forming a homomorphism of PCMs in both coordinates. These partial commutative modules are thus like vector spaces, except that their addition is partial and their scalars are probabilities in $[0, 1]$.

Example 1 *Consider the set of* partial *functions from a set X to the unit interval $[0, 1]$. Thus, for such a $f\colon X \rightharpoonup [0, 1]$ there is an output value $f(x) \in [0, 1]$ only for $x \in X$ which are in the domain $dom(f) \subseteq X$. Obviously, one can define scalar multiplication $r \bullet f$, pointwise, without change of domain. We take the empty function — nowhere defined, with empty domain — as zero element. Consider the following two partial sums that turn these partial functions into a partial commutative module.*

One way to define a partial sum $\oslash$ is to define $f \perp g$ as $dom(f) \cap dom(g) = \emptyset$; the sum $f \oslash g$ is defined on the union of the domains, via case distinction.

A second partial sum $f \oslash' g$ is defined if for each $x \in dom(f) \cap dom(g)$ one has $f(x) + g(x) \leq 1$. For those x in the overlap of domains, we define $(f \oslash' g)(x) = f(x) + g(x)$, and elsewhere $f \oslash' g$ is f on $dom(f)$ and g on $dom(g)$.

An *effect algebra* (see [7,6]) is a PCM with for each element x a unique complement $x^\perp$ satisfying $x \oslash x^\perp = 1 = 0^\perp$, together with the requirement $1 \perp x \Rightarrow x = 0$. In the unit interval $[0, 1]$ we have $r^\perp = 1 - r$. In Example 1 for both the partial sums $\oslash$ and $\oslash'$ one does *not* get an effect algebra: in the first case there is not always an $f^\perp$ with $f \oslash f^\perp = 1$, where 1 is the function that is everywhere defined and equal to 1. For $\oslash'$ there is $f^\perp$ with $f \oslash' f^\perp$, but $f^\perp$ need not be unique. E.g. the function 1 has both the empty function and the everywhere 0 function as complement. We can adapt this example to an effect algebra by considering only partial functions $X \rightharpoonup (0, 1]$, excluding 0 as outcome.

A map of effect algebras f is a map of PCMs satisfying $f(1) = 1$. This yields a subcategory $\mathbf{EA} \hookrightarrow \mathbf{PCMon}$. An *effect module* is at the same time an effect algebra and a partial commutative module. We get a subcategory $\mathbf{EMod} \hookrightarrow \mathbf{PCMod}$. By 'homming into $[0,1]$' one obtains an adjunction $\mathbf{EMod}^{\mathrm{op}} \rightleftarrows \mathbf{Conv}$, see [14] for details. The resulting triangle on the right in (7) commutes in one direction, since $\mathbf{Conv}(\mathcal{D}(X), [0,1]) \cong [0,1]^X$. In the other direction one has $\mathbf{EMod}([0,1]^X, [0,1]) \cong \mathcal{D}(X)$ for finite sets X.

In [28] it is shown that each effect module is a convex set. The proof is simple, but makes essential use of the existence of orthocomplements $(-)^\perp$. In fact, the category $\mathbf{EMod}$ is enriched over $\mathbf{Conv}$. Even stronger, the triangle on the right in (7) is enriched over $\mathbf{Conv}$. This yields $wp(\sum_i r_i f_i) = \sum_i r_i wp(f_i)$.

There are two variations on the distribution monad $\mathcal{D}$ that are worth pointing out. The first one is the expectation monad $\mathcal{E}(X) = \mathbf{EMod}([0,1]^X, [0,1])$ introduced in [19] (and used for instance in [3] for probabilistic program semantics). It can be seen as a probabilistic version of the ultrafilter monad from the previous section. For a finite set one has $\mathcal{E}(X) \cong \mathcal{D}(X)$. The category of algebras $\mathcal{EM}(\mathcal{E})$ contains the convex compact Hausdorff spaces, see [19]. This monad $\mathcal{E}$ gives rise to a triangle as on the left below, see [19] for details.

$$\begin{array}{ccc} \mathbf{EMod}^{\mathrm{op}} & \overset{Hom(-,[0,1])}{\underset{Hom(-,[0,1])}{\top}} & \mathcal{EM}(\mathcal{E}) \\ & {\scriptstyle [0,1]^{(-)}} \searrow \quad \nearrow {\scriptstyle Hom(-,[0,1])} & \\ & \mathcal{K}\ell(\mathcal{E}) & \end{array} \qquad \begin{array}{ccc} \sigma\mathbf{EMod}^{\mathrm{op}} & \overset{Hom(-,[0,1])}{\underset{Hom(-,[0,1])}{\top}} & \mathcal{EM}(\mathcal{G}) \\ & {\scriptstyle \mathbf{Meas}(-,[0,1])} \searrow \quad \nearrow {\scriptstyle Hom(-,[0,1])} & \\ & \mathcal{K}\ell(\mathcal{G}) & \end{array} \tag{8}$$

The triangle on the right captures continuous probabilistic computation, via the Giry monad $\mathcal{G}$ on the category $\mathbf{Meas}$ of measurable spaces. This is elaborated in [15]. The category $\sigma\mathbf{EMod}$ contains effect modules in which countable ascending chains have a join. Both these triangles commute, and are enriched over convex sets.

We continue with the category $\mathbf{Conv}_{\leq 1} = \mathcal{EM}(\mathcal{D}_{\leq 1})$ of subconvex sets. We now get a triangle of the form:

$$\begin{array}{ccc} \mathbf{GEMod}^{\mathrm{op}} & \overset{Hom(-,[0,1])}{\underset{Hom(-,[0,1])}{\top}} & \mathbf{Conv}_{\leq 1} = \mathcal{EM}(\mathcal{D}_{\leq 1}) \\ & {\scriptstyle [0,1]^{(-)}} \searrow \quad \nearrow {\scriptstyle Hom(-,[0,1])} & \\ & \mathcal{K}\ell(\mathcal{D}_{\leq 1}) & \end{array} \tag{9}$$

We need to describe the category $\mathbf{GEMod}$ of generalised effect modules. First, a generalised effect algebra, according to [6], is a partial commutative monoid (PCM) in which $x \oslash y = 0 \Rightarrow x = y = 0$ and $x \oslash z = y \oslash z \Rightarrow x = y$ hold. In that case one can define a partial order $\leq$ in the usual way. We obtain a full subcategory $\mathbf{GEA} \hookrightarrow \mathbf{PCMon}$. In fact we have $\mathbf{EA} \hookrightarrow \mathbf{GEA} \hookrightarrow \mathbf{PCMon}$,

since a generalised effect algebra is not an effect algebra, but a more general 'topless' structure: a generalized effect algebra with a top element 1 is an effect algebra.

One can now add multiplication with scalars from $[0,1]$ to generalised effect algebras, like for partial commutative modules. But we require more, namely the existence of subconvex sums $r_1 x_1 \circledcirc \cdots \circledcirc r_n x_n$, for $r_i \in [0,1]$ with $\sum_i r_i \leq 1$. As noted before, such sums exist automatically in effect modules, but this is not the case in generalised effect algebra with scalar multiplication, as the first structure in Example 1 illustrates. Thus we define a full subcategory $\mathbf{GEMod} \hookrightarrow \mathbf{PCMod}$, where objects of $\mathbf{GEMod}$ are at the same time partial commutative modules and generalised effect algebras, with the additional requirement that all subconvex sums exist. Summarising, we have the following diagram of 'effect' structures, where the bottom row involves scalar multiplication.

$$\begin{array}{ccccc} \mathbf{EA} & \hookrightarrow & \mathbf{GEA} & \hookrightarrow & \mathbf{PCMon} \\ \uparrow & & \uparrow & & \uparrow \\ \mathbf{EMod} & \hookrightarrow & \mathbf{GEMod} & \hookrightarrow & \mathbf{PCMod} \end{array}$$

Once we know what generalized effect modules are, it is easy to see that 'homming into $[0,1]$' yields the adjunction in (9). Moreover, this diagram (9) is enriched over $\mathbf{Conv}_{\leq 1}$, so that weakest precondition wp preserves subconvex sums of Kleisli maps (programs).

4 Quantum computation, briefly

In this section we wish to point out that the triangle (1) applies beyond the monadic setting. For instance, quantum computation, modeled via the category $\mathbf{Cstar}_{\mathrm{PU}}$ of C^*-algebras (with unit) and positive, unital maps, one obtains a triangle:

$$\begin{array}{ccc} & \overset{Hom(-,[0,1])}{\underset{Hom(-,[0,1])}{\top}} & \\ \mathbf{EMod}^{\mathrm{op}} & & \mathbf{Conv} \\ & {\scriptstyle Pred} \searrow \quad \nearrow {\scriptstyle Stat} & \\ & (\mathbf{Cstar}_{\mathrm{PU}})^{\mathrm{op}} & \end{array} \tag{10}$$

The predicate functor sends a C^*-algebra A to the unit interval $[0,1]_A \subseteq A$ of 'effects' in A, where $[0,1]_A = \{a \in A \mid 0 \leq a \leq 1\}$. This functor is full and faithful, see [8]. On the other side, the state functor sends a C^*-algebra A to the (convex) set of its states, given by the homomorphisms $A \to \mathbb{C}$. This diagram is enriched over convex sets. A similar setting of states and effects, for Hilbert spaces instead of C^*-algebras, is used in [4] for a quantum precondition calculus.

In [8] it was shown that *commutative* C^*-algebras, capturing the probabilistic, non-quantum case, can be described as a Kleisli category. It is unclear if the non-commutative, proper quantum, case can also be described via a monad.

5 Dijkstra monad examples

In [30] the 'Dijkstra' monad is introduced, as a variant of the 'Hoare' monad from [27]. It captures weakest precondition computations for the state monad $X \mapsto (S \times X)^S$, where S is a fixed collection of states (the heap). Here we wish to give a precise description of the Dijkstra monad, for various concrete computational monads T.

For the powerset monad $\mathcal{P}$, a first version of the Dijkstra monad, following the description in [30], yields $\mathfrak{D}_{\mathcal{P}} \colon \mathbf{Sets} \to \mathbf{Sets}$ defined as:

$$\mathfrak{D}_{\mathcal{P}}(X) = \mathcal{P}(S)^{\mathcal{P}(S \times X)}, \tag{11}$$

where S is again a fixed set of states. Thus, an element $w \in \mathfrak{D}_{\mathcal{P}}(X)$ is a function $w \colon \mathcal{P}(S \times X) \to \mathcal{P}(S)$ that transforms a postcondition $Q \in \mathcal{P}(X \times S)$ into a precondition $w(Q) \in \mathcal{P}(S)$. The postcondition is a binary predicate, on both an output value from X and a state from S; the precondition is a unary predicate, only on states.

In this first version (11) we simply take *all* functions $\mathcal{P}(S \times X) \to \mathcal{P}(S)$. But in the triangle (2) we see that predicate transformers are maps in $\mathbf{CL}_{\bigwedge}$, *i.e.* are meet-preserving maps between complete lattices. Hence we now properly (re)define $\mathfrak{D}_{\mathcal{P}}$ as the set of meet-preserving functions:

$$\begin{aligned} \mathfrak{D}_{\mathcal{P}}(X) &\stackrel{\text{def}}{=} \mathbf{CL}_{\bigwedge}\Big(\mathcal{P}(S \times X), \mathcal{P}(S)\Big) \\ &= \Big(\mathbf{CL}_{\bigwedge}\Big)^{\text{op}}\Big(\mathit{Pred}(S), \mathit{Pred}(S \times X)\Big) \end{aligned} \tag{12}$$

This is indeed a monad, following [30], with unit and multiplication:

$$\eta(x) = \lambda Q.\, \{s \mid (s, x) \in Q\} \qquad \mu(H) = \lambda Q.\, H\Big(\{(s, h) \mid s \in h(Q)\}\Big).$$

We introduce some notation ($\mathfrak{S}$, *i.e.* fraktur S) for the result of applying the state transformer monad to an arbitrary monad (see *e.g.* [22]).

Definition 2 *For a monad $T \colon \mathbf{Sets} \to \mathbf{Sets}$ and for a fixed set (of 'states') S, the* T-state monad $\mathfrak{S}_T$ *is defined as:*

$$\mathfrak{S}_T(X) = T(S \times X)^S = \mathcal{K}\ell(T)\Big(S, S \times X\Big).$$

For the record, its unit and multiplication are given by:

$$x \longmapsto \lambda s \in S.\, \eta(s, x) \quad \text{and} \quad H \longmapsto \mu \circ T(\lambda(s, h).\, h(s)) \circ H,$$

where η, μ are the unit and multiplication of T.

Proposition 3 *There is a map of monads $\mathfrak{S}_{\mathcal{P}} \Rightarrow \mathfrak{D}_{\mathcal{P}}$ from the $\mathcal{P}$-state monad to the $\mathcal{P}$-Dijkstra monad* (12)*, with components:*

$$\mathfrak{S}_{\mathcal{P}}(X) = \mathcal{K}\ell(\mathcal{P})\Big(S, S \times X\Big) \xrightarrow{\sigma_X} \Big(\mathbf{CL}_{\bigwedge}\Big)^{op}\Big(Pred(S), Pred(S \times X)\Big) = \mathfrak{D}_{\mathcal{P}}(X)$$

given by substitution / weakest precondition:

$$\sigma_X(f) = Pred(f) = f^* = wp(f) = \lambda Q \in \mathcal{P}(S \times X).\, \{s \mid f(s) \subseteq Q\},$$

following the description from (3)*.*

Proof. We have to check that substitution is natural in X and commutes with the units and multiplications. This is easy; for instance:

$$\begin{aligned}\Big(\sigma \circ \eta^{\mathfrak{S}}\Big)(x)(Q) \;=\; \Big(\eta^{\mathfrak{S}}(x)\Big)^*(Q) &= \{s \mid \eta^{\mathfrak{S}}(x)(s) \subseteq Q\} \\ &= \{s \mid \eta^{\mathcal{P}}(s, x) \subseteq Q\} \\ &= \{s \mid \{(s, x)\} \subseteq Q\} \\ &= \{s \mid (s, x) \in Q\} \;=\; \eta^{\mathfrak{D}}(x)(Q). \qquad \square\end{aligned}$$

At this stage the generalisation of the Dijkstra monad for other monads — with an associated logic as in (1) — should be clear. For instance, for the multiset $\mathcal{M}_R$ and (sub)distribution monad $\mathcal{D}, \mathcal{D}_{\leq 1}$ we use the triangles in (6), (7) and (9) to define associated Dijkstra monads:

$$\begin{aligned}\mathfrak{D}_{\mathcal{M}_R}(X) &= \mathbf{Mod}_R\Big(Pred(S \times X), Pred(S)\Big) \;=\; \mathbf{Mod}_R\Big(R^{S \times X}, R^S\Big) \\ \mathfrak{D}_{\mathcal{D}}(X) &= \mathbf{EMod}\Big(Pred(S \times X), Pred(S)\Big) \;=\; \mathbf{EMod}\Big([0,1]^{S \times X}, [0,1]^S\Big) \\ \mathfrak{D}_{\mathcal{D}_{\leq 1}}(X) &= \mathbf{GEMod}\Big(Pred(S \times X), Pred(S)\Big) \;=\; \mathbf{GEMod}\Big([0,1]^{S \times X}, [0,1]^S\Big)\end{aligned} \tag{13}$$

Then there is the following result, analogously to Proposition 3. The proofs involve extensive calculations but are essentially straightforward.

Proposition 4 *For the multiset, distribution, and subdistribution monads $\mathcal{M}_R$, $\mathcal{D}$, and $\mathcal{D}_{\leq 1}$ there are maps of monads given by substitution:*

$$\mathfrak{S}_{\mathcal{M}_R} \xRightarrow{(-)^*} \mathfrak{D}_{\mathcal{M}_R} \qquad \mathfrak{S}_{\mathcal{D}} \xRightarrow{(-)^*} \mathfrak{D}_{\mathcal{D}} \qquad \mathfrak{S}_{\mathcal{D}_{\leq 1}} \xRightarrow{(-)^*} \mathfrak{D}_{\mathcal{D}_{\leq 1}}$$

from the associated state monads to the associated Dijkstra monads (13). □

The Dijkstra monad associated with the expectation monad $\mathcal{E}$ is the same as for the distribution monad $\mathcal{D}$. Hence one gets a map of monads $\mathfrak{S}_{\mathcal{E}} \Rightarrow \mathfrak{D}_{\mathcal{D}}$, with substitution components:

$$\begin{array}{rcl} \mathfrak{S}_{\mathcal{E}}(X) = \mathcal{E}(S \times X)^S = \mathbf{EMod}\Big([0,1]^{S\times X}, [0,1]\Big)^S & & \\ & \Big\downarrow{\scriptstyle (-)^*} & \\ & \mathbf{EMod}\Big([0,1]^{S\times X}, [0,1]^S\Big) = \mathfrak{D}_{\mathcal{D}}(X) & \end{array}$$

where $f^*(q)(s) = f(s)(q)$. Details are left to the reader.

6 Dijkstra's monad, beyond examples

We have seen several instances of state-and-effect triangles. It would be highly desirable to have a general method to construct for an arbitrary monad T (on **Sets**) an associated 'logical category' **Log**, leading to a triangle of the form:

$$\begin{array}{ccc} \mathbf{Log}^{\mathrm{op}} & \overset{\top}{\rightleftarrows} & \mathcal{EM}(T) \\ & {\scriptstyle Pred}\nwarrow \quad \nearrow{\scriptstyle Stat} & \\ & \mathcal{K}\ell(T) & \end{array}$$

In the previous version [16] of this paper a general adjunction $\mathbf{Sets}^{\mathrm{op}} \rightleftarrows \mathcal{EM}(T)$ was described, induced by a fixed algebra $T(\Omega) \to \Omega$. This suggests some way towards a category **Log** by suitably restricting this adjunction. But how to do that in general remains unclear.

Here we take a different route: we simply assume that we have a functor $Pred\colon \mathcal{K}\ell(T) \to \mathbf{Log}^{\mathrm{op}}$ and investigate what properties it should satisfy in order to define a 'Dijkstra' monad — and also a Hoare monad in the next section. These requirements turn out to be rather light. In the next section we ask the same question for the Hoare monad. In that case we need much stronger properties.

Theorem 5 *Let T be an arbitrary monad on* **Sets**, *and let S be an arbitrary set (of states), for which we form the associated state monad $\mathfrak{S}_T = (S \times -)^S$. Let $Pred\colon \mathcal{K}\ell(T) \to \mathbf{Log}^{op}$ be a functor to some (unspecified) category* **Log**. *If Pred preserves copowers, then the definition*

$$\mathfrak{D}_T(X) = \mathbf{Log}\Big(Pred(S \times X), Pred(S)\Big)$$

yields a monad on **Sets**. *It comes equipped with a map of monads* $\mathfrak{S}_T \Rightarrow \mathfrak{D}_T$.

As before we simply write $f^* = \mathit{Pred}(f) \colon \mathit{Pred}(Y) \to \mathit{Pred}(X)$, for a Kleisli map $f \colon X \to T(Y)$, and call this f^* a substitution morphism.

For an arbitrary index set I, a copower is an I-fold coproduct $\coprod_{i \in I} X$ without any dependence on i. It is often written as $I \cdot X$. In **Sets** such a copower is simply the cartesian product $I \times X$. Since colimits in a Kleisli category are inherited from the underlying category, this $I \times X$ is also a copower in $\mathcal{K}\ell(T)$.

A power is a 'constant' product $\prod_{i \in I} X$, written as X^I. Since the functor $\mathit{Pred} \colon \mathcal{K}\ell(T) \to \mathbf{Log}^{\mathrm{op}}$ goes to an opposite, preservation of copowers means that it sends copowers in $\mathcal{K}\ell(T)$ to powers in **Log**. Thus, there are (canonical) isomorphisms:

$$\mathit{Pred}(X)^I \xrightarrow[\cong]{\mathsf{cpp}} \mathit{Pred}(X \times I) \tag{14}$$

in the category **Log**. These maps are called cpp, for 'copower preservation'. They are canonical in the sense that they are inverses to the maps $\mathit{Pred}(X \times I) \to \mathit{Pred}(X)^I$ obtained as tuple $\langle J(\kappa_i)^* \rangle_{i \in I}$ for the coprojection functions $\kappa_i \colon X \to X \times I$, given by $\kappa_i(x) = \langle x, i \rangle$ in **Sets**. In $\mathcal{K}\ell(T)$ these coprojections are $J(\kappa_i) = \eta^T \circ \kappa_i = \eta^{\mathfrak{S}}(i)$, where $\eta^{\mathfrak{S}}$ is the unit of the state monad $\mathfrak{S}_T = T(S \times -)^S$. This preservation holds for Boolean and fuzzy predicates, since $\mathcal{P}(X)^I \cong \mathcal{P}(X \times I)$ and $([0,1]^X)^I \cong [0,1]^{X \times I}$. The maps cpp in (14) are natural in X, in the sense that for $f \colon X \to T(Y)$ and $g \colon I \to J$ the following diagram commutes.

$$\begin{array}{ccc} \mathit{Pred}(X)^I & \xrightarrow[\cong]{\mathsf{cpp}} & \mathit{Pred}(X \times I) \\ {\scriptstyle f^* \circ (-) \circ g = (f^*)^g}\uparrow & & \uparrow{\scriptstyle (\mathsf{st}_1 \circ (f \times g))^*} \\ \mathit{Pred}(Y)^J & \xrightarrow[\cong]{\mathsf{cpp}} & \mathit{Pred}(Y \times J) \end{array} \tag{15}$$

The map on the right uses the strength map $\mathsf{st}_1 \colon T(Y) \times J \to T(Y \times J)$ from Subsection 1.1.

Proof. (Of Theorem 5) For $f \colon X \to Y$ in **Sets** we have $\mathfrak{D}_T(f) \colon \mathfrak{D}_T(X) \to \mathfrak{D}_T(Y)$ given by $\mathfrak{D}_T(f)(h) = h \circ J(\mathrm{id}_S \times f)^* \colon \mathit{Pred}(S \times Y) \to \mathit{Pred}(S \times X) \to \mathit{Pred}(S)$. The unit $\eta^{\mathfrak{D}} \colon X \to \mathfrak{D}_T(X)$ is given by $\eta^{\mathfrak{D}}(x) = \eta^{\mathfrak{S}}(x)^*$, where $\eta^{\mathfrak{S}}(x) = \lambda s.\, (s, x) \colon S \to T(S \times X)$ as in Definition 2. It is not hard to see that $\eta^{\mathfrak{D}}$ is a natural transformation.

For the definition of the multiplication $\mu^{\mathfrak{D}} \colon \mathfrak{D}_T(\mathfrak{D}_T(X)) \to \mathfrak{D}_T(X)$ we use that the predicate functor preserves copowers. For a map $H \colon \mathit{Pred}(S \times \mathfrak{D}_T(X)) \to \mathit{Pred}(S)$ we have to define $\mu^{\mathfrak{D}}(H) \colon \mathit{Pred}(S \times X) \to \mathit{Pred}(S)$. We take the following composite.

$$\mathit{Pred}(S \times X) \xrightarrow{\langle h \rangle_{h \in \mathfrak{D}_T(X)}} \mathit{Pred}(S)^{\mathfrak{D}_T(X)} \xrightarrow[\cong]{\mathsf{cpp}} \mathit{Pred}(S \times \mathfrak{D}_T(X)) \xrightarrow{H} \mathit{Pred}(S)$$

We illustrate the verification of one of the monad laws. For $h \in \mathfrak{D}_T(X)$,

$$\begin{aligned}
\Big(\mu^{\mathfrak{D}}_X \circ \eta^{\mathfrak{D}}_{\mathfrak{D}_T(X)}\Big)(h) &= \mu^{\mathfrak{D}}_X\Big(\eta^{\mathfrak{S}}(h)^*\Big) \\
&= \eta^{\mathfrak{S}}(h)^* \circ \mathsf{cpp} \circ \langle k \rangle_{k \in \mathfrak{D}_T(X)} \\
&\overset{(*)}{=} (\lambda f.\, f(h)) \circ \langle k \rangle_{k \in \mathfrak{D}_T(X)} \\
&= \lambda p.\, (\lambda f.\, f(h))\Big(\langle k \rangle_{k \in \mathfrak{D}_T(X)}(p)\Big) \\
&= \lambda p.\, (\lambda f.\, f(h))\Big(\lambda k.\, k(p)\Big) \\
&= \lambda p.\, h(p) \\
&= h.
\end{aligned}$$

The marked equation $\overset{(*)}{=}$ holds since $(\lambda f.\, f(h)) \circ \mathsf{cpp}^{-1} = (\eta^{\mathfrak{S}}(h))^*$, as shown below.

$$\begin{aligned}
\Big((\lambda f.\, f(h)) \circ \mathsf{cpp}^{-1}\Big)(p) &= \mathsf{cpp}^{-1}(p)(h) \\
&= (\eta^T \circ \kappa_h)^*(p) \\
&= (\lambda s.\, \eta^T(\kappa_h(s)))^*(p) \\
&= (\lambda s.\, \eta^T(s, h))^*(p) \\
&= (\lambda s.\, \eta^{\mathfrak{S}}(h)(s))^*(p) \\
&= (\eta^{\mathfrak{S}}(h))^*(p).
\end{aligned}$$

Like before, substitution forms a map of monads $\mathfrak{S}_T \Rightarrow \mathfrak{D}_T$, by sending a function $f \in \mathfrak{S}_T(X) = T(S \times X)^S$ to $f^* \in \mathbf{Log}\Big(\mathit{Pred}(S \times X), \mathit{Pred}(S)\Big) = \mathfrak{D}_T(X)$. □

7 The Hoare monad

The Dijkstra monad captures functions from postconditions to preconditions. In Theorem 5 in the previous section we have seen that very little logical structure is needed to define the Dijkstra monad for a monad T. We proceed with the Hoare monad, and will see below that it requires — in contrast — non-trivial logical structure. This Hoare monad contains triples (P, h, Q) of precondition, program, and postcondition, corresponding to a Hoare triple $\{P\}h\{Q\}$, with meaning: if condition P holds in state s in which the program h is executed, then postcondition Q holds for all states and outputs produced by running h in state s.

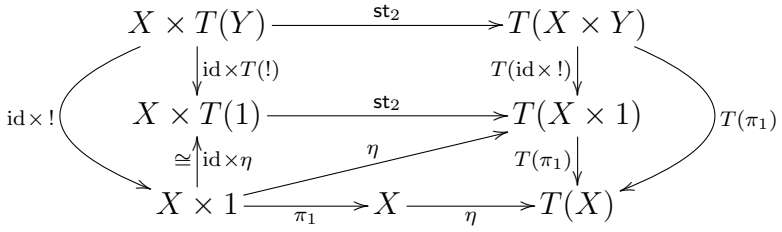
This section contains one result, namely Theorem 7. It involves a bit of explanation. First, a monad T is called *affine* (see [10]) if it preserves the terminal object 1, or, more specifically, if the unit map $\eta\colon 1 \to T(1)$ is an isomorphism. The powerset monad is not affine, but the nonempty-powerset monad is affine.

The distribution monad $\mathcal{D}$ is also affine, and of course, the identity monad is affine. The following result is an adaptation of [10, Lemma 4.2].

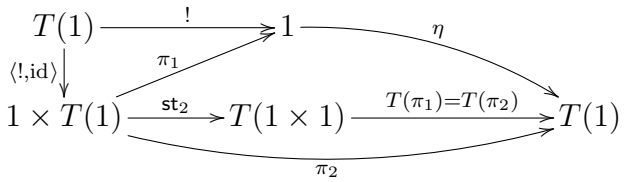
Lemma 6 *For a monad T on* **Sets**, *the following statements are equivalent.*

(1) T is affine, i.e. $T(1) \cong 1$;
(2) $T(\pi_1) \circ \mathsf{st}_2 = \eta \circ \pi_1 \colon X \times T(Y) \to T(X)$;
(3) $T(\pi_2) \circ \mathsf{st}_1 = \eta \circ \pi_2 \colon T(X) \times Y \to T(Y)$.

Proof. The equivalence of points (2) and (3) is easy and left to the reader, so we concentrate on (1) $\Leftrightarrow$ (2). First assume that the unit $\eta \colon 1 \to T(1)$ is an isomorphism. The equation $T(\pi_1) \circ \mathsf{st}_2 = \eta \circ \pi_1$ follows from the following diagram chase.



In the other direction we use the equality of projections $\pi_1 = \pi_2 \colon 1 \times 1 \to 1$ to prove $\eta \circ ! = \mathrm{id}$ in:



□

Theorem 7 *Let T be an affine monad on* **Sets** *and S be a set of states. Let Pred: $\mathcal{K}\ell(T) \to \mathbf{MSL}^{op}$ be a functor to the category* **MSL** *of meet semi-lattices. This functor Pred satisfies the following additional properties.*

- *It preserves copowers, as described after Definition 5.*
- *There are left and right adjoints $\coprod_f \dashv J(f)^* \dashv \prod_f$ to substitution maps $J(f)^* \colon \mathrm{Pred}(Y) \to \mathrm{Pred}(X)$, for each $f \colon X \to Y$ in* **Sets**. *The left adjoints satisfy the Frobenius property:*

$$\coprod_f \big(J(f)^*(P) \wedge Q \big) = P \wedge \coprod_f (Q).$$

  We also require the Beck-Chevalley condition, as described below.
- *Each semi-lattice $\mathrm{Pred}(X)$ is actually a Heyting algebra. Hence there is an implication operation $P \supset Q$ in each $\mathrm{Pred}(X)$. Moreover, substitution $J(f)^*$ along a pure map $J(f) = \eta^T \circ f$ preserves $\supset$.*

Sending a set X to the set of Hoare triples:

$$\mathfrak{H}_T(X) = \Big\{(P,h,Q) \,\Big|\, P \in \mathit{Pred}(S), h \in \mathfrak{S}_T(X), Q \in \mathit{Pred}(S \times (S \times X)) \\ \text{with } P \leq (\mathsf{st}_2 \circ \langle \mathrm{id}, h\rangle)^*(Q)\Big\}.$$

then yields a monad $\mathfrak{H}_T$ on **Sets**, *which is appropriately called the* Hoare monad *for T, following in [27,31]. Forgetting the pre- and postconditions gives a map of monads $\mathfrak{H}_T \Rightarrow \mathfrak{S}_T$ to the state monad $\mathfrak{S}_T = T(S \times -)^S$.*

The inequality $P \leq (\mathsf{st}_2 \circ \langle \mathrm{id}, h\rangle)^*(Q)$ in the definition of $\mathfrak{H}_T(X)$ may be read informally as: if $P(s)$, then $Q(s, h(s))$. The postcondition Q is thus a relation between a 'pre-state' s and the result $h(s)$ involving a post-state in S and an outcome in X. Having such a pre-state in postconditions is convenient in specification, for instance for expressing properties like $\mathsf{val} = \mathsf{old}(\mathsf{val})+1$, where $\mathsf{old}(\mathsf{val})$ refers to the value of the variable val in the pre-state. Such old-syntax occurs for instance in the specification language JML for Java, see [21].

Notice that it is implicit in the type of the functor $\mathit{Pred}\colon \mathcal{K}\ell(T) \to \mathbf{MSL}^{\mathrm{op}}$ that substitution maps $g^* = \mathit{Pred}(g)$ preserve finite meets $(\wedge, 1)$. However, we do not require that all substitution g^* also preserves implication $\supset$, only for pure maps $g = J(f) = \eta^T \circ f$. But since each g^* preserves $\wedge$ we do get an inequality in one direction, namely: $g^*(P \supset Q) \leq g^*(P) \supset g^*(Q)$. This will be used below.

We do require what is called a Beck-Chevalley condition (see [11] for more information) for the adjoints $\coprod_f, \prod_f$. It takes the following special form. Given a pullback in **Sets** of the form:

$$\begin{array}{ccc} X & \xrightarrow{\;f\;} & Y \\ {\scriptstyle h}\downarrow & \lrcorner & \downarrow{\scriptstyle k} \\ A & \xrightarrow[\;g\;]{} & B \end{array}$$

then there are equations:

$$\coprod_f \circ J(h)^* = J(k)^* \circ \coprod_g \qquad J(k)^* \circ \prod_g = \prod_f \circ J(h)^* \tag{16}$$

Given this Beck-Chevalley property for pure maps, one can prove that the cpp maps not only commute with substitution $(-)^*$, as in (15), but also with $\coprod$, $\prod$ and $\supset$.

Remark 8 (ii) below explains why the requirement that the monad T be affine makes sense in the presence of such quantification. But first we present the proof of Theorem 7. It is a long exercise in categorical logic. A simplified, logical description is given in Remark 8 (i). It may be helpful.

Proof. To start we define three projection maps $\mathsf{pre}, \mathsf{prog}, \mathsf{post}$, namely:

$$\mathsf{pre}(P,h,Q) = P \qquad \mathsf{prog}(P,h,Q) = h \qquad \mathsf{post}(P,h,Q) = Q.$$

These maps have types $\mathsf{pre}\colon \mathfrak{H}_T(X) \to \mathit{Pred}(S)$, $\mathsf{prog}\colon \mathfrak{H}_T(X) \to \mathfrak{S}_T(X)$, and $\mathsf{post}\colon \mathfrak{H}_T(X) \to \mathit{Pred}(S\times(S\times X))$. The last claim of the theorem says that prog is a natural transformation $\mathfrak{D}_T \Rightarrow \mathfrak{S}_T$. This will be the case by construction, since for the middle, program part of $\mathfrak{D}_T$ we copy $\mathfrak{S}_T$.

We first have to show that $\mathfrak{H}_T$ is a functor on **Sets**. For a function $f\colon X \to Y$ one obtains $\mathfrak{H}_T(f)\colon \mathfrak{H}_T(X) \to \mathfrak{H}_T(Y)$ by:

$$\mathfrak{H}_T(f)\big(P,h,Q\big) = \big(P,\, \mathfrak{S}_T(f)(h),\, \textstyle\coprod_{\mathrm{id}\times(\mathrm{id}\times f)}(Q)\big),$$

where $\mathfrak{S}_T(f)(h) = T(\mathrm{id}\times f)\circ h$, see Definition 2. This map $\mathfrak{H}_T(f)$ is well-defined since, if we assume $(\mathsf{st}_2\circ\langle \mathrm{id}, h\rangle)^*(Q) \geq P$, then:

$$\begin{aligned}
&\Big(\mathsf{st}_2\circ\langle \mathrm{id}, \mathfrak{S}_T(f)(h)\rangle\Big)^*\Big(\textstyle\coprod_{\mathrm{id}\times(\mathrm{id}\times f)}(Q)\Big)\\
&= \Big(\mathsf{st}_2\circ\langle \mathrm{id}, T(\mathrm{id}\times f)\circ h\rangle\Big)^*\Big(\textstyle\coprod_{\mathrm{id}\times(\mathrm{id}\times f)}(Q)\Big)\\
&= \Big(\mathsf{st}_2\circ(\mathrm{id}\times T(\mathrm{id}\times f))\circ\langle \mathrm{id}, h\rangle\Big)^*\Big(\textstyle\coprod_{\mathrm{id}\times(\mathrm{id}\times f)}(Q)\Big)\\
&= \Big(T(\mathrm{id}\times(\mathrm{id}\times f))\circ\mathsf{st}_2\circ\langle \mathrm{id}, h\rangle\Big)^*\Big(\textstyle\coprod_{\mathrm{id}\times(\mathrm{id}\times f)}(Q)\Big)\\
&= \Big(\mu\circ T(\eta^T\circ(\mathrm{id}\times(\mathrm{id}\times f)))\circ\mathsf{st}_2\circ\langle \mathrm{id}, h\rangle\Big)^*\Big(\textstyle\coprod_{\mathrm{id}\times(\mathrm{id}\times f)}(Q)\Big)\\
&= \Big(J(\mathrm{id}\times(\mathrm{id}\times f))\bullet(\mathsf{st}_2\circ\langle \mathrm{id}, h\rangle)\Big)^*\Big(\textstyle\coprod_{\mathrm{id}\times(\mathrm{id}\times f)}(Q)\Big)\\
&= (\mathsf{st}_2\circ\langle \mathrm{id}, h\rangle)^* J(\mathrm{id}\times(\mathrm{id}\times f)^*\Big(\textstyle\coprod_{\mathrm{id}\times(\mathrm{id}\times f)}(Q)\Big)\\
&\geq \Big(\mathsf{st}_2\circ\langle \mathrm{id}, h\rangle\Big)^*(Q) \qquad \text{via the unit of } \textstyle\coprod_{(-)} \dashv J(-)^*\\
&\geq P.
\end{aligned}$$

In order to define the unit map $\eta^{\mathfrak{H}}\colon X \to \mathfrak{H}_T(X)$ we use that each set of predicates $\mathit{Pred}(X)$ has a greatest element 1_X, and that substitution maps preserves these top elements. Then:

$$\eta^{\mathfrak{H}}(x) = \Big(1_S, \eta^{\mathfrak{S}}(x), \textstyle\coprod_{\langle \mathrm{id},\kappa_x\rangle}(1_S)\Big)$$

where $\kappa_x = \lambda s \in S.\,(s,x)\colon S \to S\times X$ and $\coprod_{\kappa_x} \dashv (\eta^T\circ\kappa_x)^* = \eta^{\mathfrak{S}}(x)^* = \eta^{\mathfrak{D}}(x)$. We leave it to the reader to verify that $\eta^{\mathfrak{H}}$ is a natural transformation.

The definition of the multiplication map $\mu\colon (\mathfrak{H}_T)^2(X) \to \mathfrak{H}_T(X)$ involves more

work:

$$\begin{aligned}\mu^{\mathfrak{H}}(P,H,Q) &= (P',H',Q') \qquad \text{where}\\ P' &= P \wedge \prod\nolimits_{\pi_1}\Big(Q \supset J(\pi_2)^*(\mathsf{cpp}(\mathsf{pre}))\Big)\\ H' &= \mu^T \circ T\Big(\lambda(s,h).\,\mathsf{prog}(h)(s)\Big) \circ H\\ Q' &= \coprod\nolimits_{\pi_1}\Big(J(\pi_1 \times \mathrm{id})^*(Q) \wedge \\ &\qquad\qquad J(\langle\langle \pi_1 \circ \pi_2, \pi_2 \circ \pi_1\rangle, \pi_2 \circ \pi_2\rangle)^*(\mathsf{cpp}(\mathsf{post}))\Big).\end{aligned} \tag{17}$$

We briefly explain the three parts P', H', Q' of this definition.

- Applying the copower preservation map $\mathsf{cpp}\colon \mathit{Pred}(S)^{\mathfrak{H}_T(X)} \xrightarrow{\cong} \mathit{Pred}(S \times \mathfrak{H}_T(X))$ to the first projection $\mathsf{pre}\colon \mathfrak{H}_T(X) \to \mathit{Pred}(S)$ yields a predicate $\mathsf{cpp}(\mathsf{pre}) \in \mathit{Pred}(S \times \mathfrak{H}_T(X))$. Via the projection $\pi_2\colon S \times (S \times \mathfrak{H}_T(X)) \to S \times \mathfrak{H}_T(X)$ we get $J(\pi_2)^*(\mathsf{cpp}(\mathsf{pre})) \in \mathit{Pred}(S \times (S \times \mathfrak{H}_T(X)))$. Since Q is a predicate on the same set $S \times (S \times \mathfrak{H}_T(X))$, we can form the implication $Q \supset J(\pi_2)^*(\mathsf{cpp}(\mathsf{pre}))$. The projection $\pi_1\colon S \times (S \times \mathfrak{H}_T(X)) \to S$, yields a substitution map $J(\pi_1)^*\colon \mathit{Pred}(S) \to \mathit{Pred}(S \times (S \times \mathfrak{H}_T(X)))$, which has $\prod_{\pi_1}$ as right adjoint. Thus, the product $\prod_{\pi_1}\Big(Q \supset J(\pi_2)^*(\mathsf{cpp}(\mathsf{pre}))\Big)$ is a predicate on S.
- The map H is by definition in $\mathfrak{S}_T(\mathfrak{H}_T(X))$, and thus has type $H\colon S \to T(S \times \mathfrak{H}_T(X))$. For $(s,h) \in S \times \mathfrak{H}_T(X)$ we have $\mathsf{prog}(h)\colon S \to T(S \times X)$. Thus the composite in the second part in (17) is of the appropriate form:

  $$H' = \Big(S \xrightarrow{H} T(S \times \mathfrak{H}_T(X)) \xrightarrow{T(\mathsf{prog}')} T^2(S \times X) \xrightarrow{\mu^T} T(S \times X)\Big)$$

  where $\mathsf{prog}'\colon S \times \mathfrak{H}_T(X) \to T(S \times X)$ is $\mathsf{prog}'(s,h) = \mathsf{prog}(h)(s)$. This allows us to write H' as Kleisli composition $H' = \mathsf{prog}' \bullet H$.
- The postcondition $Q' \in \mathit{Pred}(S \times (S \times X))$ requires most work. A crucial role is played by the projection:

  $$\Big(S \times (S \times X)\Big) \times \Big(S \times \mathfrak{H}_T(X)\Big) \xrightarrow{\pi_1} S \times (S \times X)$$

  We move the predicates $Q \in \mathit{Pred}(S \times (S \times \mathfrak{H}_T(X)))$ and $\mathsf{cpp}(\mathsf{post}) \in \mathit{Pred}((S \times (S \times X)) \times \mathfrak{H}_T(X))$ to the domain of this projection. We abbreviate:

  $$\begin{aligned}Q_1 &= J(\pi_1 \times \mathrm{id})^*(Q)\\ Q_2 &= J(\langle\langle \pi_1 \circ \pi_2, \pi_2 \circ \pi_1\rangle, \pi_2 \circ \pi_2\rangle)^*(\mathsf{cpp}(\mathsf{post})),\end{aligned}$$

  and define $Q' = \coprod_{\pi_1}(Q_1 \wedge Q_2)$ as in (17).

We will show that the multiplication $\mu^{\mathfrak{H}}$ in (17) is well-defined, *i.e.* that $P' \leq (\mathsf{st}_2 \circ \langle \mathrm{id}, H'\rangle)^*(Q')$. This involves some work.

First, the triple $(P, H, Q) \in \mathfrak{H}_T(\mathfrak{H}_T(X))$ satisfies $P \leq (\mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle)^*(Q)$. The counit of the adjunction $J(-)^* \dashv \prod_{(-)}$ yields in $\mathit{Pred}(S \times \mathfrak{H}_T(X))$,

$$J(\pi_1)^* \prod_{\pi_1} \Big(Q \supset J(\pi_2)^*(\mathsf{cpp}(\mathsf{pre})) \Big) \leq Q \supset J(\pi_2)^*(\mathsf{cpp}(\mathsf{pre})).$$

By applying $(\mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle)^*$ to both sides we get:

$$\begin{aligned}
&(\mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle)^* J(\pi_1)^* \prod_{\pi_1} \Big(Q \supset (\eta \circ \pi_2)^*(\mathsf{cpp}(\mathsf{pre})) \Big) \\
&\leq (\mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle)^* (Q \supset J(\pi_2)^*(\mathsf{cpp}(\mathsf{pre}))) \\
&\leq (\mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle)^*(Q) \supset (\mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle)^* J(\pi_2)^*(\mathsf{cpp}(\mathsf{pre})).
\end{aligned}$$

On the left-hand-side we use that the monad T is affine to prove that the composition of substitution functors $(\mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle)^* \circ J(\pi_1)^*$ is the identity $\mathrm{id} = (\eta^T)^*$, in:

$$\begin{aligned}
J(\pi_1) \bullet (\mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle) &= \mu^T \circ T(\eta^T \circ \pi_1) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle \\
&= T(\pi_1) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle \\
&= \eta^T \circ \pi_1 \circ \langle \mathrm{id}, H\rangle \qquad \text{by Lemma 6} \\
&= \eta^T.
\end{aligned}$$

And on the right-hand-side we see that:

$$J(\pi_2) \bullet (\mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle) = T(\pi_2) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle = \pi_2 \circ \langle \mathrm{id}, H\rangle = H.$$

Altogether we have $P' \leq P \leq (\mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle)^*(Q)$, and also:

$$P' \leq \prod_{\pi_1} \Big(Q \supset J(\pi_2)^*(\mathsf{cpp}(\mathsf{pre})) \Big) \leq (\mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle)^*(Q) \supset H^*(\mathsf{cpp}(\mathsf{pre})).$$

Hence we obtain $P' \leq H^*(\mathsf{cpp}(\mathsf{pre}))$ by modus ponens.

At this stage we need a 'dependent' version of the naturality diagram (15) for cpp. It goes as follows. For a function $f\colon Y \times I \to T(Z)$ we have:

$$\begin{array}{ccc}
\mathit{Pred}(Y)^I & \xrightarrow[\cong]{\mathsf{cpp}} & \mathit{Pred}(Y \times I) \\
{\scriptstyle f^\#}\uparrow & & \uparrow{\scriptstyle (\mathsf{st}_1 \circ \langle f, \pi_2\rangle)^*} \\
\mathit{Pred}(Z)^I & \xrightarrow[\cong]{\mathsf{cpp}} & \mathit{Pred}(Z \times I)
\end{array} \tag{18}$$

where the function $f^\#$ is defined by $f^\#(Q)(i) = f(-, i)^*(Q(i))$. We leave it to the reader to verify that this diagram commutes, using the inverse of cpp.

We apply this diagram with $f = \mathsf{st}_2 \circ \langle \pi_1, \mathsf{prog}'\rangle\colon S \times \mathfrak{H}_T(X) \to T(S \times (S \times X))$, where $\mathsf{prog}'(s, h) = \mathsf{prog}(h)(s)$. This function prog has type $\mathfrak{H}_T(X) \to T(S \times$

$X)^S$ and satisfies, by definition of $\mathfrak{H}_T(X)$, for each $h \in \mathfrak{H}_T(X)$, the following inequality.

$$\mathsf{pre}(h) \leq (\mathsf{st}_2 \circ \langle \mathrm{id}, \mathsf{prog}(h)\rangle)^*\Big(\mathsf{post}(h)\Big) = f^{\#}(\mathsf{post})(h).$$

Using this last equality we get:

$$\begin{aligned} \mathsf{cpp}(\mathsf{pre}) &\leq \mathsf{cpp}((\mathsf{st}_2 \circ \langle \pi_1, \mathsf{prog}'\rangle)^{\#}(\mathsf{post})) \\ &\overset{(18)}{=} (\mathsf{st}_1 \circ \langle \mathsf{st}_2 \circ \langle \pi_1, \mathsf{prog}'\rangle, \pi_2\rangle)^*(\mathsf{cpp}(\mathsf{post})). \end{aligned}$$

Recall that our aim is to prove:

$$P' \leq (\mathsf{st}_2 \circ \langle \mathrm{id}, H'\rangle)^*(P') = (\mathsf{st}_2 \circ \langle \mathrm{id}, H'\rangle)^*(\coprod_{\pi_1}(Q_1 \wedge Q_2)).$$

The unit of the adjunction $\coprod_{(-)} \dashv J(-)^*$ gives an inequality $Q_1 \wedge Q_2 \leq J(\pi_1)^*(\coprod_{\pi_1}(Q_1 \wedge Q_2))$. We apply on both sides the map $f = \mu^T \circ T(\mathsf{st}_1 \circ \langle \mathsf{st}_2 \circ (\mathrm{id} \times \mathsf{prog}'), \pi_2\rangle) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle \colon S \to T((S \times (S \times X)) \times (S \times \mathfrak{H}_T(X)))$. It yields:

$$f^*(Q_1) \wedge f^*(Q_2) = f^*(Q_1 \wedge Q_2) \leq f^*J(\pi_1)^*(\coprod_{\pi_1}(Q_1 \wedge Q_2)). \tag{19}$$

Our strategy is to prove the following three equations.

(a) $f^*J(\pi_1)^* = (\mathsf{st}_2 \circ \langle \mathrm{id}, H'\rangle)^*$, so that the right-hand-side in (19) equals $(\mathsf{st}_2 \circ \langle \mathrm{id}, H'\rangle)^*\coprod_{\pi_1}(Q_1 \wedge Q_2) = (\mathsf{st}_2 \circ \langle \mathrm{id}, H'\rangle)^*(Q')$;
(b) $f^*(Q_1) = (\mathsf{st}_2 \circ \langle \mathrm{id}, H\rangle)^*(Q) \geq P'$;
(c) $f^*(Q_2) = H^*(\mathsf{st}_1 \circ \langle \mathsf{st}_2 \circ \langle \pi_1, \mathsf{prog}'\rangle, \pi_2\rangle)^*(\mathsf{cpp}(\mathsf{post})) \geq H^*(\mathsf{cpp}(\mathsf{pre})) \geq P'$.

Using these inequalities, that we proved before, we are done showing that $P' \leq (\mathsf{st}_2 \circ \langle \mathrm{id}, H'\rangle)^*(Q')$, and thus that the multiplication map $\mu^{\mathfrak{H}}$ in (17) is well-defined.

For point (a) we calculate:

$$
\begin{aligned}
&J(\pi_1) \bullet f \\
&= \mu^T \circ T(\eta^T \circ \pi_1) \circ \mu^T \circ T(\mathsf{st}_1 \circ \langle \mathsf{st}_2 \circ (\mathrm{id} \times \mathsf{prog}'), \pi_2 \rangle) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= T(\pi_1) \circ \mu^T \circ T(\mathsf{st}_1 \circ \langle \mathsf{st}_2 \circ (\mathrm{id} \times \mathsf{prog}'), \pi_2 \rangle) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= \mu^T \circ T(T(\pi_1) \circ \mathsf{st}_1 \circ \langle \mathsf{st}_2 \circ (\mathrm{id} \times \mathsf{prog}'), \pi_2 \rangle) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= \mu^T \circ T(\pi_1 \circ \langle \mathsf{st}_2 \circ (\mathrm{id} \times \mathsf{prog}'), \pi_2 \rangle) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= \mu^T \circ T(\mathsf{st}_2 \circ (\mathrm{id} \times \mathsf{prog}')) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= \mu^T \circ T(\mathsf{st}_2) \circ \mathsf{st}_2 \circ (\mathrm{id} \times T(\mathsf{prog}')) \circ \langle \mathrm{id}, H \rangle \\
&= \mathsf{st}_2 \circ (\mathrm{id} \times \mu^T) \circ \langle \mathrm{id}, T(\mathsf{prog}') \circ H \rangle \\
&= \mathsf{st}_2 \circ \langle \mathrm{id}, H' \rangle.
\end{aligned}
$$

Similarly, point (b) is obtained by using that $Q_1 = J(\pi \times \mathrm{id})^*(Q)$ and:

$$
\begin{aligned}
&J(\pi_1 \times \mathrm{id}) \bullet f \\
&= \mu^T \circ T(\eta^T \circ (\pi_1 \times \mathrm{id})) \circ \mu^T \circ T(\mathsf{st}_1 \circ \langle \mathsf{st}_2 \circ (\mathrm{id} \times \mathsf{prog}'), \pi_2 \rangle) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= T(\pi_1 \times \mathrm{id}) \circ \mu^T \circ T(\mathsf{st}_1 \circ \langle \mathsf{st}_2 \circ (\mathrm{id} \times \mathsf{prog}'), \pi_2 \rangle) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= \mu^T \circ T(T(\pi_1 \times \mathrm{id}) \circ \mathsf{st}_1 \circ \langle \mathsf{st}_2 \circ (\mathrm{id} \times \mathsf{prog}'), \pi_2 \rangle) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= \mu^T \circ T(\mathsf{st}_1 \circ (T(\pi_1) \times \mathrm{id}) \circ \langle \mathsf{st}_2 \circ (\mathrm{id} \times \mathsf{prog}'), \pi_2 \rangle) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= \mu^T \circ T(\mathsf{st}_1 \circ \langle \eta^T \circ \pi_1 \circ (\mathrm{id} \times \mathsf{prog}'), \pi_2 \rangle) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \qquad \text{by Lemma 6} \\
&= \mu^T \circ T(\eta^T \circ \langle \pi_1, \pi_2 \rangle) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle.
\end{aligned}
$$

Finally, for point (c) we use $Q_2 = J(\langle\langle \pi_1 \circ \pi_2, \pi_2 \circ \pi_1 \rangle, \pi_2 \circ \pi_2 \rangle)^*(\mathsf{cpp}(\mathsf{post}))$ and:

$$
\begin{aligned}
&J(\langle\langle \pi_1 \circ \pi_2, \pi_2 \circ \pi_1 \rangle, \pi_2 \circ \pi_2 \rangle) \bullet f \\
&= T\langle\langle \pi_1 \circ \pi_2, \pi_2 \circ \pi_1 \rangle, \pi_2 \circ \pi_2 \rangle) \circ \mu^T \circ T(\mathsf{st}_1 \circ \langle \mathsf{st}_2 \circ (\mathrm{id} \times \mathsf{prog}'), \pi_2 \rangle) \\
&\qquad \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= \mu^T \circ T(T\langle\langle \pi_1 \circ \pi_2, \pi_2 \circ \pi_1 \rangle, \pi_2 \circ \pi_2 \rangle) \circ \mathsf{st}_1 \circ (\mathsf{st}_2 \times \mathrm{id}) \circ \langle \mathrm{id} \times \mathsf{prog}', \pi_2 \rangle) \\
&\qquad \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&\overset{(*)}{=} \mu^T \circ T(\mathsf{st}_1 \circ (\mathsf{st}_2 \times \mathrm{id}) \circ \langle\langle \pi_1 \circ \pi_2, \pi_2 \circ \pi_1 \rangle, \pi_2 \circ \pi_2 \rangle \circ \langle \mathrm{id} \times \mathsf{prog}', \pi_2 \rangle) \\
&\qquad \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= \mu^T \circ T(\mathsf{st}_1 \circ (\mathsf{st}_2 \times \mathrm{id}) \circ \langle\langle \pi_1 \circ \pi_2, \mathsf{prog}' \circ \pi_2 \rangle, \pi_2 \circ \pi_2 \rangle) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= \mu^T \circ T(\mathsf{st}_1 \circ (\mathsf{st}_2 \times \mathrm{id}) \circ \langle\langle \pi_1, \mathsf{prog}' \rangle, \pi_2 \rangle) \circ T(\pi_2) \circ \mathsf{st}_2 \circ \langle \mathrm{id}, H \rangle \\
&= \mu^T \circ T(\mathsf{st}_1 \circ \langle \mathsf{st}_2 \circ \langle \pi_1, \mathsf{prog}' \rangle, \pi_2 \rangle) \circ H.
\end{aligned}
$$

The marked equation $\overset{(*)}{=}$ may be obtained by unpacking the definitions of st_1 and st_2.

We thus have well-defined unit and multiplication maps. Verifying that they satisfy the appropriate equations to make the functor $\mathfrak{H}_T$ into a monad is still a lot of work, involving Frobenius, Beck-Chevalley, and various preservation properties of the 'copower preservation maps' cpp. It proceeds along the lines given above, and will be left to the meticulous reader. $\square$

An easy instance of Theorem 7 is obtained by taking the identity monad $T = \mathrm{id}$ on **Sets**, with standard classical logic $Pred\colon \mathbf{Sets} \to \mathbf{MSL}^{\mathrm{op}}$ given by $Pred(X) = \mathcal{P}(X)$ and substitution $f^* = Pred(f) = f^{-1}$ given by inverse image. The associated monad $\mathfrak{H}_{\mathrm{id}}$ then captures Hoare triples for deterministic computations with state, as described by the state monad $\mathfrak{S}_{\mathrm{id}}(X) = (S \times X)^S$.

Another, proper, example involves the non-empty powerset monad $\mathcal{P}_\bullet\colon \mathbf{Sets} \to \mathbf{Sets}$. Notice that it is affine, since $\mathcal{P}_\bullet(1) \cong 1$, using that the singleton set $1 = \{0\}$ has precisely one non-empty subset, namely $1 \subseteq 1$. The associated predicate functor $Pred\colon \mathcal{K}\ell(\mathcal{P}_\bullet) \to \mathbf{MSL}^{\mathrm{op}}$ is given on objects again by $Pred(X) = \mathcal{P}(X)$, and on arrows now by $Pred(g)(Q) = g^*(Q) = \{x \mid g(x) \subseteq Q\}$, for $g\colon X \to \mathcal{P}(Y)$, like in (3). Each function (or pure map) $f\colon X \to Y$ in **Sets** yields $J(f)^* = (\{-\} \circ f)^*\colon Pred(Y) \to Pred(X)$ by $J(f)^*(Q) = \{x \mid \{f(x)\} \subseteq Q\} = f^{-1}(Q)$. These inverse image maps f^{-1} have left and right adjoints $Pred(Y) \to Pred(X)$ in a standard way, namely:

$$\coprod_f(P) = \{y \mid \exists x.\, f(x) = y \wedge P(x)\} = \{f(x) \mid P(x)\}$$
$$\prod_f(P) = \{y \mid \forall x.\, f(x) = y \Rightarrow P(x)\}.$$

We now obtain a Hoare monad $\mathfrak{H}_{\mathcal{P}_\bullet}$ for non-deterministic computations with state, in $\mathcal{P}_\bullet(S \times X)^S$.

Remark 8 *(1) The monad definitions used in Theorem 7 and its proof are based on the ones in [27]. There, they are formulated in the language of the theorem prover Coq. We deviate in two points, one small, and one more significant.*

- *We use categorical language, instead of type-theoretic language. With sufficient fluency in categorical logic, one sees that the constructions are essentially the same.*
- *We extend the construction by weaving in an affine monad T. Thus, the work in [27] is a special case of our construction, when T is the identity monad.*

As background information to the categorical formulations that we use we give the corresponding logical descriptions of the unit and multiplication of the Hoare monad $\mathfrak{H} = \mathfrak{H}_{\mathrm{id}}$, for the identity monad $T = \mathrm{id}$. For an

element $x \in X$, and triple $(P, H, Q) \in \mathfrak{H}^2(X)$,

$$\begin{aligned}\eta(x) &= \big(1_S,\ \lambda s.\,(s,x),\ \{(s,s,x) \mid s \in S\}\big)\\ \mu(P,H,Q) &= \big(\{s \mid P(s) \wedge \forall s', h.\, Q(s,s',h) \Rightarrow \mathsf{pre}(h)(s')\},\\ &\qquad \lambda s.\, \mathsf{prog}(\pi_2 H(s))(\pi_1 H(s)),\\ &\qquad \{(s,t,x) \mid \exists s', h.\, Q(s,s',h) \wedge \mathsf{post}(h)(s',t,x)\}\big)\end{aligned}$$

(2) The requirement that the monad T be affine is relevant in the context of quantification. We shall explain this for the powerset monad $\mathcal{P}$ and the non-empty powerset monad $\mathcal{P}_\bullet$. Notice that $\mathcal{P}_\bullet$ is affine, since $\mathcal{P}_\bullet(1) \cong 1 = \{0\}$, but $\mathcal{P}$ is not, since $\mathcal{P}(1) \cong 2 = \{0,1\}$. We consider both $\mathcal{K}\ell(\mathcal{P})$ and $\mathcal{K}\ell(\mathcal{P}_\bullet)$ with the standard logic $Pred(X) = \mathcal{P}(X)$.

Now consider the coproduct $\coprod_{\pi_2}$ for the special case of a projection $\pi_2 \colon X \times Y \to Y$. Then $\coprod_{\pi_2}(P) = \{y \mid \exists x.\, P(x,y)\}$. If we have a Kleisli map $g \colon Y \to X$, then we expect to be able to prove, intuitively that from $P(g(y), y)$ we can deduce $\exists x.\, P(x,y)$. But in setting with monads, this is a bit more subtle. First, the predicate $P(g(y), y)$ obtained by substitution, is really the result:

$$(\mathsf{st}_1 \circ \langle g, \mathrm{id}\rangle)^*(P) \quad \textit{where} \quad \mathsf{st}_1 \circ \langle g, \mathrm{id}\rangle \colon Y \to \mathcal{P}(X) \times Y \to \mathcal{P}(X \times Y).$$

Hence, this substitution is:

$$(\mathsf{st}_1 \circ \langle g, \mathrm{id}\rangle)^*(P) = \{y \mid \forall x \in g(y).\, P(x,y)\}.$$

From this we obtain $\exists x.\, P(x,y)$ if $g(y) \neq \emptyset$, that is, if $g \colon Y \to X$ is a map in $\mathcal{K}\ell(\mathcal{P}_\bullet)$ instead of in $\mathcal{K}\ell(\mathcal{P})$.

Categorically, for an affine monad T, this works as follows. There is the unit of the adjunction $P \leq J(\pi_2)^* \coprod_{\pi_2}(P)$. Hence by applying $(\mathsf{st}_1 \circ \langle g, \mathrm{id}\rangle)^*$ substitution on both sides we get:

$$(\mathsf{st}_1 \circ \langle g, \mathrm{id}\rangle)^*(P) \leq (\mathsf{st}_1 \circ \langle g, \mathrm{id}\rangle)^*(J(\pi_2)^*(\textstyle\coprod_{\pi_2}(P)) \overset{(*)}{=} \coprod_{\pi_2}(P).$$

For this second, marked equation $\overset{(*)}{=}$ we use that the monad T is affine, via Lemma 6:

$$\begin{aligned}J(\pi_2) \bullet (\mathsf{st}_1 \circ \langle g, \mathrm{id}\rangle) &= T(\pi_2) \circ \mathsf{st}_1 \circ \langle g, \mathrm{id}\rangle\\ &= \eta \circ \pi_2 \circ \langle g, \mathrm{id}\rangle\\ &= \eta.\end{aligned}$$

(3) An important example of an affine functor is the distribution monad $\mathcal{D} \colon \mathbf{Sets} \to \mathbf{Sets}$, introduced in Section 3. It comes with a predicate logic $Pred \colon \mathcal{K}\ell(\mathcal{D}) \to \mathbf{EMod}^{op}$, where $Pred(X) = [0,1]^X$ is a meet semi-lattice, via the standard order on the unit interval $[0,1]$, used pointwise. However, it does not satisfy the requirements of Theorem 7 since substitution

functors f^* do not preserve finite meets. We include a simple counter example.

Consider the Kleisli map $f\colon 1 \to \mathcal{D}(2)$ given by $f = \frac{1}{4}|0\rangle + \frac{3}{4}|1\rangle$, with predicates $q_1, q_2 \in [0,1]^2$ given by $q_1(0) = \frac{1}{5}$, $q_1(1) = \frac{1}{3}$, $q_2(0) = \frac{1}{6}$, $q_2(1) = \frac{1}{2}$. We obtain $f^*(q_i) \in [0,1]^1 \cong [0,1]$ and their meet via:

$$\begin{aligned} f^*(q_1) &= \textstyle\sum_i f(i) \cdot q_1(i) = \frac{1}{4} \cdot \frac{1}{5} + \frac{3}{4} \cdot \frac{1}{3} = \frac{1}{20} + \frac{1}{4} = \frac{3}{10} \\ f^*(q_2) &= \frac{1}{4} \cdot \frac{1}{6} + \frac{3}{4} \cdot \frac{1}{2} = \frac{1}{24} + \frac{3}{8} = \frac{5}{12} \\ f^*(q_1) \wedge f^*(q_2) &= \frac{3}{10} \wedge \frac{5}{12} = \frac{3}{10}. \end{aligned}$$

But:

$$f^*(q_1 \wedge q_2) = \tfrac{1}{4} \cdot (\tfrac{1}{5} \wedge \tfrac{1}{6}) + \tfrac{3}{4} \cdot (\tfrac{1}{3} \wedge \tfrac{1}{2}) = \tfrac{1}{4} \cdot \tfrac{1}{6} + \tfrac{3}{4} \cdot \tfrac{1}{3} = \tfrac{1}{24} + \tfrac{1}{4} = \tfrac{7}{24}.$$

8 Concluding remarks

The triangle-based semantics and logic that was presented via many examples forms the basis for (a) several versions of the Dijkstra monad, associated with different monads T, and (b) a description of the weakest precondition operation as a map of monads. The categorical predicate logic for a monad T on **Sets** is axiomatised subsequently in such a way that the Dijkstra and Hoare monads (for T) can be defined. In the end, it leads to maps of monads (Hoare) $\Rightarrow$ (State) $\Rightarrow$ (Dijkstra), by combining Theorems 5 and 7.

Acknowledgments

Thanks to Sam Staton, Mathys Rennela, and Bas Westerbaan for their input & feedback.

References

[1] S. Abramsky. Domain theory in logical form. *Ann. Pure & Appl. Logic*, 51(1/2):1–77, 1991.

[2] S. Awodey. *Category Theory.* Oxford Logic Guides. Oxford Univ. Press, 2006.

[3] G. Barthe, B. Grégoire, and S. Zanella Béguelin. Formal certification of code-based cryptographic proofs. In *Principles of Programming Languages*, pages 90–101. ACM Press, 2009.

[4] E. D'Hondt and P. Panangaden. Quantum weakest preconditions. *Math. Struct. in Comp. Sci.*, 16(3):429–451, 2006.

[5] E. Dijkstra and C. Scholten. *Predicate Calculus and Program Semantics*. Springer, Berlin, 1990.

[6] A. Dvurečenskij and S. Pulmannová. *New Trends in Quantum Structures*. Kluwer Acad. Publ., Dordrecht, 2000.

[7] D. J. Foulis and M.K. Bennett. Effect algebras and unsharp quantum logics. *Found. Physics*, 24(10):1331–1352, 1994.

[8] R. Furber and B. Jacobs. From Kleisli categories to commutative C^*-algebras: Probabilistic Gelfand duality. In R. Heckel and S. Milius, editors, *Conference on Algebra and Coalgebra in Computer Science (CALCO 2013)*, number 8089 in Lect. Notes Comp. Sci., pages 141–157. Springer, Berlin, 2013.

[9] T. Heinosaari and M. Ziman. *The Mathematical Language of Quantum Theory. From Uncertainty to Entanglement*. Cambridge Univ. Press, 2012.

[10] B. Jacobs. Semantics of weakening and contraction. *Ann. Pure & Appl. Logic*, 69(1):73–106, 1994.

[11] B. Jacobs. *Categorical Logic and Type Theory*. North Holland, Amsterdam, 1999.

[12] B. Jacobs. Convexity, duality, and effects. In C. Calude and V. Sassone, editors, *IFIP Theoretical Computer Science 2010*, number 82(1) in IFIP Adv. in Inf. and Comm. Techn., pages 1–19. Springer, Boston, 2010.

[13] B. Jacobs. Introduction to coalgebra. Towards mathematics of states and observations. Book, in preparation, version 2, 2012.

[14] B. Jacobs. New directions in categorical logic, for classical, probabilistic and quantum logic. See arxiv.org/abs/1205.3940, 2012. To appear in *Logical Methods in Computer Science*.

[15] B. Jacobs. Measurable spaces and their effect logic. In *Logic in Computer Science*. IEEE, Computer Science Press, 2013.

[16] B. Jacobs. Dijkstra monads in monadic computation. In M. Bonsangue, editor, *Coalgebraic Methods in Computer Science (CMCS 2014)*, number 8446 in Lect. Notes Comp. Sci., pages 135–150. Springer, Berlin, 2014.

[17] B. Jacobs. Introduction to coalgebra. Towards mathematics of states and observations. 2014.

[18] B. Jacobs and J. Mandemaker. Coreflections in algebraic quantum logic. *Found. of Physics*, 42(7):932–958, 2012.

[19] B. Jacobs and J. Mandemaker. The expectation monad in quantum foundations. In B. Jacobs, P. Selinger, and B. Spitters, editors, *Quantum Physics and Logic (QPL) 2011*, volume 95 of *Elect. Proc. in Theor. Comp. Sci.*, pages 143–182, 2012.

[20] P. Johnstone and S. Vickers. Preframe presentations present. In A. Carboni, M.C. Pedicchio, and G. Rosolini, editors, *Como Conference on Category Theory*, number 1488 in Lect. Notes Math., pages 193–212. Springer, Berlin, 1991.

[21] G. Leavens, A. Baker, and C. Ruby. JML: A notation for detailed design. In H. Kilov and B. Rumpe, editors, *Behavioral Specifications of Business and Systems*, pages 175–188. Kluwer, 1999.

[22] S. Liang, P. Hudak, and M. Jones. Monad transformers and modular interpreters. In *Principles of Programming Languages*, pages 333–343. ACM Press, 1995.

[23] E. Manes. A triple-theoretic construction of compact algebras. In B. Eckman, editor, *Seminar on Triples and Categorical Homolgy Theory*, number 80 in Lect. Notes Math., pages 91–118. Springer, Berlin, 1969.

[24] Y. Maruyama. Categorical duality theory: With applications to domains, convexity, and the distribution monad. In S. Ronchi Della Rocca, editor, *Computer Science Logic*, pages 500–520. Leibniz Int. Proc. in Informatics, 2013.

[25] S. Mac Lane. *Categories for the Working Mathematician.* Springer, Berlin, 1971.

[26] E. Moggi. Notions of computation and monads. *Inf. & Comp.*, 93(1):55–92, 1991.

[27] A. Nanevski, G. Morrisett, A. Shinnar, P. Govereau, and L. Birkedal. Ynot: dependent types for imperative programs. In *International Conference on Functional Programming (ICFP)*, pages 229–240. ACM SIGPLAN Notices, 2008.

[28] S. Pulmannová and S. Gudder. Representation theorem for convex effect algebras. *Commentationes Mathematicae Universitatis Carolinae*, 39(4):645–659, 1998.

[29] M. Stone. Postulates for the barycentric calculus. *Ann. Math.*, 29:25–30, 1949.

[30] N. Swamy, J. Weinberger, C. Schlesinger, J. Chen, and B. Livshits. Verifying higher-order programs with the Dijkstra monad. In *Proc. of the 34th ACM SIGPLAN conf. on Programming language design and implementation (PLDI)*, pages 387–398. ACM, 2013.

[31] W. Swierstra. A Hoare logic for the state monad. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *Theorem Proving in Higher Order Logics*, number 5674 in Lect. Notes Comp. Sci., pages 440–451. Springer, Berlin, 2009.