Dijkstra 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

February 26, 2014

Abstract. The Dijkstra monad has been introduced recently for capturing weakest precondition computations within the context of program verification, supported by a theorem prover. Here we give a more general description of such Dijkstra 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. Instantiations of this triangle for different monads T show how to define the Dijkstra monad associated with T, via the logic involved. Technically, we provide a morphism of monads from the state monad transformation applied to T, to the Dijkstra monad associated with T. This monad map is precisely the weakest precondition map in the triangle, given in categorical terms by substitution.

1 Introduction

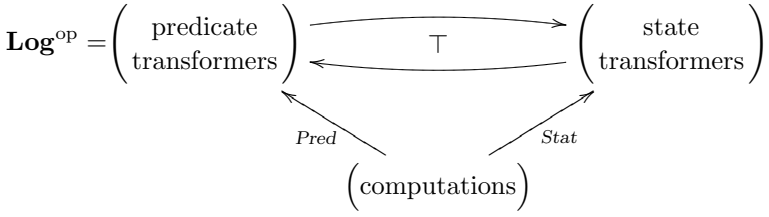
A monad is a categorical concept that is surprisingly useful in the theory of computation. On the one hand it describes a form of computation (such as partial, non-deterministic, or probabilistic), and on the other hand it captures various algebraic structures. Technically, the computations are maps 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 [23]. 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.

Interestingly, the range of examples of monads has been extended recently from computation to program logic. So-called Hoare monads [24,29] and Dijkstra monads [28] 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 for the Dijkstra monad. We generalise the

* To appear in the LNCS proceedings of CMCS 2014.

original definition from [28] and show that a "Dijkstra" monad can be associated with various well-known monads that are used for modelling computations. (The Hoare monad will be mentioned briefly towards the end.)

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 [13] (see also [12]) a view on program semantics and logic emerged involving a triangle of the form:



(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 post-condition to a precondition. In this paper we do not expand on the precise logical structure involved (which connectives, which quantifiers, *etc.* in **Log**) and simply claim that this 'indexed category' on the left is a model of some predicate logic. The reason is that at this stage we don't need more structure than 'substitution', which is provided by the functoriality of *Pred*.

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 in the category 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) an associated monad is defined (in Section 5) that captures predicate transfomers. Therefore, this new monad is called a "Dijkstra" monad, following [28].

We list the main points of this paper.

1. The paper explains the unified view on program semantics and logic as given by the above triangle (1) by presenting many examples, involving non-deterministic, partial, linear, probabilistic, and also quantum computation. This involves some new results, like the adjunction for partial computation in (5) in the next section.
2. Additionally, in many of these examples the enriched nature of these categories and functors is shown, capturing some essential compositional aspects of the weakest precondition operation. The role of these enrichments resembles the algebraic effects, see *e.g.* [25]; it goes beyond the topic of the current paper, but definitely deserves further investigation.
3. A necessary step towards understanding the Dijkstra monad is made, by simplifying previous accounts [28] and casting them in proper categorical language.
4. Using this combined view on computations and logic, for the different monad examples T in this paper, an associated "Dijkstra monad" $\mathfrak{D}_T$ is defined. This definition depends on the logic **Log** that is used to reason about T, since the monad is defined via a homset in this category **Log**. This logic-based approach goes well beyond the particular logic that is used in the original article [28], where the Dijkstra monad is introduced, since it now also applies to for instance probabilistic computation, in various forms.
5. Once we have the Dijkstra monad $\mathfrak{D}_T$ associated with T we define a "map of monads" $\mathfrak{S}_T \Rightarrow \mathfrak{D}_T$, where $\mathfrak{S}_T$ is the T-state monad, obtained by applying the state monad transformer to T. This map of monads is precisely the weakest precondition operation (categorically: substitution). This operation that is fundamental in the work of Dijkstra is thus captured neatly in categorical / monadic terms.
6. Finally, a general construction is presented that defines the Dijkstra monad $\mathfrak{D}_T$ for an arbitrary monad T on **Sets**. A deeper understanding of the construction requires a systematic account of how the categories "**Log**" in (1) arise in general. This is still beyond current levels of understanding.

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 that weakest precondition computation forms a map of monads. These examples are generalised in Section 6. Finally, Section 7 wraps up with some concluding remarks.

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:

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

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 $Pred\colon \mathcal{K}\ell(\mathcal{P}) \to (\mathbf{CL}_{\bigwedge})^{\mathrm{op}}$ 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:

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

In categorical logic, this $Pred(f)$ 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 $wp(f)$, since it forms the weakest precondition operation for f, see [4]. Clearly, it preserves arbitrary meets (intersections). It is not hard to see that the triangle (2) commutes.

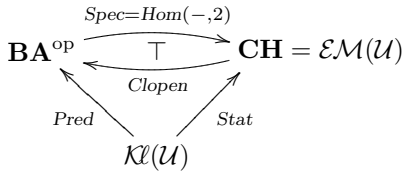
Interestingly, the diagram (2) involves additional structure on homsets. If we have a collection of parallel maps f_i in $\mathcal{K}\ell(\mathcal{P})$, we can take their (pointwise) join $\bigvee_{i \in I} f_i$. Pre- and post-composition preserves such joins. This means that the Kleisli category $\mathcal{K}\ell(\mathcal{P})$ 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: $wp(\bigvee_i f_i) = \bigwedge_i wp(f_i)$.

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 $X \cong \mathcal{U}(X)$.

A famous result of [19] says that the category of algebras of $\mathcal{U}$ is the category **CH** of compact Hausdorff spaces (and continuous functions). It yields the following triangle.

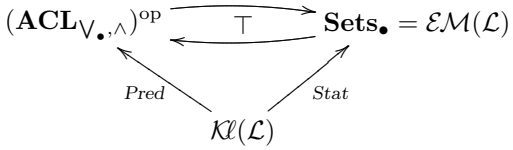


(4)

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:



(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 [12] 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 [11], 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 $supp(\varphi) = \{x \in X \mid \varphi(x) \neq 0\}$ 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 $supp(\varphi) = \{x_1, \ldots, x_n\}$ 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:

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

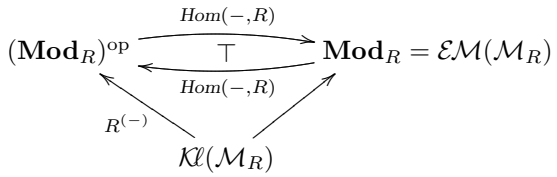
The unit η and multiplication μ for $\mathcal{D}$ are as for $\mathcal{M}_R$. We consider another variation, namely the *subdistribution* monad $\mathcal{D}_{\leq 1}$, where $\mathcal{D}_{\leq 1}(X)$ contains the formal *subconvex* combinations $\sum_i r_i|x_i\rangle$ where $\sum_i r_i \leq 1$. It has 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 [23]. 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 $x \xrightarrow{r} x'$ if $c(x)(x') = r \in [0,1]$ 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 $S \times M \to M$ 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 [21, 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 [27], see [10] 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 and computation the general diagram (1) takes the following form, where $\mathbf{Mod}_R = \mathcal{EM}(\mathcal{M}_R)$ and $\mathbf{Conv} = \mathcal{EM}(\mathcal{D})$.

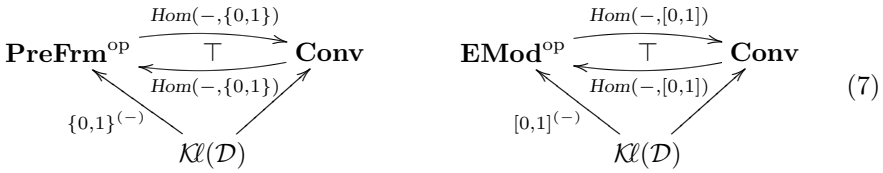


(6)

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 $f^* = R^f \colon R^Y \to R^X$ 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 post-condition 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.* [12]; 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 [12]. The one on the left is investigated further in [20]. It uses the category **PreFrm** of preframes: posets with directed joins and finite meets, distributing over these joins, see [16]. 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 $\owedge\colon M \times M \to M$ which is commutative and associative, in a suitable sense, and has a zero element $0 \in M$. One writes $x \perp y$ if $x \owedge y$ is defined. A morphism $f\colon M \to N$ of PCMs satisfies: $x \perp x'$ implies $f(x) \perp f(x')$, and then $f(x \owedge x') = f(x) \owedge f(x')$. This yields a category which we shall write as **PCMon**.

The unit interval $[0, 1]$ is clearly a PCM, with $r \owedge r'$ 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 [14] 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 $\owedge$ is to define $f \perp g$ as $dom(f) \cap dom(g) = \emptyset$; the sum $f \owedge g$ is defined on the union of the domains, via case distinction.

A second partial sum $f \owedge' g$ 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 $(f \owedge' g)(x) = f(x) + g(x)$, and elsewhere $f \owedge' g$ is f on $dom(f)$ and g on $dom(g)$.

An *effect algebra* (see [7,5]) is a PCM with for each element x a unique complement $x^\perp$ satisfying $x \owedge x^\perp = 1 = 0^\perp$, 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 $\owedge$ and $\owedge'$ one does *not* get an effect algebra: in the first case there is not always an $f^\perp$ with $f \owedge f^\perp = 1$, where 1 is the function that is everywhere defined and equal to 1. For $\owedge'$ there is $f^\perp$ with $f \owedge' f^\perp$, 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 [12] 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 [26] 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 **EMod** is enriched over **Conv**. Even stronger, the triangle on the right in (7) is enriched over **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 [15] (and used for instance in [2] 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 [15]. This monad $\mathcal{E}$ gives rise to a triangle as on the left below, see [15] for details.

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

The triangle on the right captures continuous probabilistic computation, via the Giry monad $\mathcal{G}$ on the category **Meas** of measurable spaces. This is elaborated in [13]. The category σ**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:

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

We need to describe the category **GEMod** of generalised effect modules. First, a generalised effect algebra, according to [5], is a partial commutative monoid (PCM) in which $x \owedge y = 0 \Rightarrow x = y = 0$ and $x \owedge z = y \owedge z \Rightarrow x = y$ 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 $r_1 x_1 \owedge \cdots \owedge r_n x_n$, for $r_i \in [0,1]$ with $\sum_i r_i \leq 1$. As noted before, such sums exist automatically in effect algebras, 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.

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

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, modelled via the category $\mathbf{Cstar}_{\mathrm{PU}}$ of C^*-algebras (with unit) and positive, unital maps, one obtains a triangle:

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

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 [3] 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 [28] the "Dijkstra" monad is introduced, as a variant of the "Hoare" monad from [24]. 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 monads T.

For the powerset monad $\mathcal{P}$, a first version of the Dijkstra monad, following the description in [28], 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 post-condition 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:

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

This is indeed a monad, following [28], 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.* [18]).

Definition 1. *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 1. *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:*

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

given by substitution / weakest precondition:

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

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:

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

Then there is the following result, analogously to Proposition 1. The proofs involve extensive calculations but are essentially straightforward.

Proposition 2. *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). $\square$

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:

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

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

6 Dijkstra's monad, beyond examples

In the end it remains a bit unsatisfactory to see only particular instances of what we called a Dijkstra monad $\mathfrak{D}_T$. Below we offer a more general description,

even though it is not the definitive story. For convenience we restrict ourselves to monads on **Sets**.

So let $T\colon \mathbf{Sets} \to \mathbf{Sets}$ be an arbitrary monad. As observed in (an exercise in) [11], each (fixed) Eilenberg-Moore algebra $\omega\colon T(\Omega) \to \Omega$ determines an adjunction $\mathbf{Sets}^{\mathrm{op}} \rightleftarrows \mathcal{EM}(T)$, via functors $\Omega^{(-)}\colon \mathbf{Sets}^{\mathrm{op}} \to \mathcal{EM}(T)$ and $\mathit{Hom}(-,\omega)\colon \mathcal{EM}(T) \to \mathbf{Sets}^{\mathrm{op}}$. It makes sense to require that the algebra ω is a cogenerator in $\mathcal{EM}(T)$, making the unit of the adjunction injective, but this is not needed in general. The adjunction can be generalised to strong monads on monoidal categories with equalisers, but that is not so relevant at this stage.

With this adjunction we can form a triangle of the form:

$$\begin{array}{ccc} & \overset{\mathit{Hom}(-,\Omega)}{\underset{\mathit{Hom}(-,\omega)}{\mathbf{Sets}^{\mathrm{op}} \rightleftarrows \mathcal{EM}(T)} \;\top & \\ {\scriptstyle \mathit{Pred}=\mathit{Hom}(K-,\omega)\cong\Omega^{(-)}}\nwarrow & & \nearrow{\scriptstyle K} \\ & \mathcal{K}\ell(T) & \end{array} \tag{14}$$

The induced predicate functor *Pred* is defined on a Kleisli map $f\colon X \to T(Y)$ as:

$$\Omega^Y \ni q \longmapsto \Big(X \xrightarrow{f} T(Y) \xrightarrow{T(q)} T(\Omega) \xrightarrow{\omega} \Omega\Big).$$

Appropriate restrictions of this adjunction may give rise to more suitable triangles, like in (2) and (4) – (9). How to do this restriction in a systematic manner is unclear at this stage.

But what we can do is define for a fixed set of states S, a Dijkstra monad, namely:

$$\mathfrak{D}_T(X) = \mathbf{Sets}^{\mathrm{op}}\big(\mathit{Pred}(S), \mathit{Pred}(S \times X)\big) = \mathbf{Sets}\big(\Omega^{S\times X}, \Omega^S\big). \tag{15}$$

There is a unit $\eta_X\colon X \to \mathfrak{D}_T(X)$, namely $\eta_X(x)(q)(s) = q(s,x)$, and a multiplication $\mu_X\colon (\mathfrak{D}_T)^2(X) \to \mathfrak{D}_T(X)$ given by $\mu(H)(q) = H\big(\lambda(t,k).\, k(q)(t)\big)$.

In this general situation we can define a map of monads $\sigma\colon \mathfrak{S}_T \Rightarrow \mathfrak{D}_T$, where $\mathfrak{S}_T$ is the T-state monad $X \mapsto T(S \times X)^S$ from Definition 1. This σ has components $\sigma_X\colon T(S\times X)^S \to \mathbf{Sets}(\Omega^{S\times X}, \Omega^S)$ given by weakest precondition: $\sigma_X(f) = \mathit{Pred}(f) = f^* = \mathit{wp}(f)\colon \Omega^{S\times X} \to \Omega^S$.

Thus, in this purely set-theoretic setting we can define for an arbitrary monad T an associated Dijkstra monad $\mathfrak{D}_T$ as in (15), together with a 'weakest precondition' map of monads $\mathfrak{S}_T \Rightarrow \mathfrak{D}_T$. However, the general formulation (15) does not take into account that predicate transformers preserve certain logical structure, as in the concrete examples in Section 5.

We conclude with two more observations.

1. In the triangle (14) there are two functors $\mathcal{K}\ell(T) \to \mathcal{EM}(T)$, namely the comparison functor K and $L = \mathit{Hom}(-,\Omega) \circ \mathit{Pred} = \mathbf{Sets}(\Omega^{(-)}, \Omega)$. There is a natural transformation $\tau\colon K \Rightarrow L$ with components:

$$\tau_X(u)(p) = \big(\omega \circ T(p)\big)(u) \qquad \text{where } u \in K(X) = T(X) \text{ and } p \in \Omega^X.$$

The triangle (14) commutes in both directions if this τ is an isomorphism.

2. By composing the two adjunctions $\mathbf{Sets} \rightleftarrows \mathcal{EM}(T) \rightleftarrows \mathbf{Sets}^{\mathrm{op}}$ in (14) one obtains a composite adjunction, which yields another monad T_ω on $\mathbf{Sets}$, namely:

$$T_\omega(X) = \big(U \circ \Omega^{(-)} \circ \mathit{Hom}(-, \omega) \circ F\big)(X) \cong \mathbf{Sets}(\Omega^X, \Omega).$$

This is what Lawvere [17] calls the dual monad; a similar construction occurs for instance in [6, Section 5]. There is in this case a map of monads $T \Rightarrow T_\omega$.

7 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. There are many issues that remain to be investigated.

- We have concentrated on Dijkstra monads $\mathfrak{D}$, but there is also the Hoare monad $\mathfrak{H}$, see [24,29]. It may be described explicitly as:

$$\mathfrak{H}(X) = \coprod_{P \subseteq S} \coprod_{Q \subseteq S \times X \times S} \{f \colon P \to X \times S \mid \forall s \in S.\, Q(s, f(s))\}.$$

  where S is the set of states. It would be nice to extend this Hoare construction also to other monads than powerset.
- As already mentioned in the beginning, we only scratch the surface when it comes to the enrichment involved in the examples. This also requires further investigation, especially in connection with the algebraic effects approach, see *e.g.* [25], or the (enriched) monad models of [22].

Acknowledgements 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. 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.
3. E. D'Hondt and P. Panangaden. Quantum weakest preconditions. *Math. Struct. in Comp. Sci.*, 16(3):429–451, 2006.
4. E. Dijkstra and C. Scholten. *Predicate Calculus and Program Semantics*. Springer, Berlin, 1990.
5. A. Dvurečenskij and S. Pulmannová. *New Trends in Quantum Structures*. Kluwer Acad. Publ., Dordrecht, 2000.

6. J. Egger, R.E. Møgelberg, and A. Simpson. Linearly-used continuations in the enriched effect calculus. In L. Ong, editor, *Foundations of Software Science and Computation Structures*, number 6014 in Lect. Notes Comp. Sci., pages 18–32. Springer, Berlin, 2010.
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 2009)*, 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. 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.
11. B. Jacobs. Introduction to coalgebra. Towards mathematics of states and observations. Book, in preparation, version 2, 2012.
12. 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*.
13. B. Jacobs. Measurable spaces and their effect logic. In *Logic in Computer Science*. IEEE, Computer Science Press, 2013.
14. B. Jacobs and J. Mandemaker. Coreflections in algebraic quantum logic. *Found. of Physics*, 42(7):932–958, 2012.
15. 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.
16. 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.
17. F. Lawvere. Ordinal sums and equational doctrines. In B. Eckman, editor, *Seminar on Triples and Categorical Homology Theory*, number 80 in Lect. Notes Math., pages 141–155. Springer, Berlin, 1969.
18. S. Liang, P. Hudak, and M. Jones. Monad transformers and modular interpreters. In *Principles of Programming Languages*, pages 333–343. ACM Press, 1995.
19. 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.
20. 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.
21. S. Mac Lane. *Categories for the Working Mathematician*. Springer, Berlin, 1971.
22. R. Møgelberg and S. Staton. Linearly-used state in models of call-by-value. In A. Corradini, B. Klin, and C. Cïrstea, editors, *Conference on Algebra and Coalgebra in Computer Science (CALCO 2011)*, number 6859 in Lect. Notes Comp. Sci., pages 293–313. Springer, Berlin, 2011.
23. E. Moggi. Notions of computation and monads. *Inf. & Comp.*, 93(1):55–92, 1991.
24. 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.

25. G. Plotkin and J. Power. Computational effects and operations: An overview. In *Proc. of the Workshop on Domains VI*, number 73 in Elect. Notes in Theor. Comp. Sci., pages 149–163. Elsevier, Amsterdam, 2004.
26. S. Pulmannová and S. Gudder. Representation theorem for convex effect algebras. *Commentationes Mathematicae Universitatis Carolinae*, 39(4):645–659, 1998.
27. M. Stone. Postulates for the barycentric calculus. *Ann. Math.*, 29:25–30, 1949.
28. 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.
29. 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.