The following full text is a preprint version which may differ from the publisher's version.

For additional information about this publication click this link.
http://hdl.handle.net/2066/161285

Please be advised that this information was generated on 2018-09-08 and may be subject to change.
Abstract

Healthiness is a good old question in program logics that dates back to Dijkstra. It asks for an intrinsic characterization of those predicate transformers which arise as the (backward) interpretation of a certain class of programs. There are several results known for healthiness conditions: for deterministic programs, nondeterministic ones, probabilistic ones, etc. Building upon our previous works on so-called state-and-effect triangles, we contribute a unified categorical framework for investigating healthiness conditions. This framework is based on a dual adjunction induced by a dualizing object and on our notion of relative Eilenberg-Moore algebra. The latter notion seems interesting in its own right in the context of monads, Lawvere theories and enriched categories.

Categories and Subject Descriptors F.3.2 [Semantics of Programming Languages]: Algebraic Approaches to Semantics

Keywords program logic, category theory, duality

1. Introduction

Predicate Transformer Semantics of Computation Program logics are formal systems for reasoning about programs. They come in different styles: in the Floyd-Hoare logic [14] one derives triples of a precondition, a program and a postcondition; dynamic logics [11] are logics that have programs as modal operators; type-theoretic pre- and postconditions would have predicates as refinement (or dependent) types, allowing smooth extension to higher-order programs; and many program verification tools for imperative programs have programs represented as control flow graphs, where predicates are labels to the edges. Whatever presentation style is taken, the basic idea that underlies these variations of program logics is that of weakest precondition, dating back to Dijkstra [7]. It asks: in order to guarantee a given postcondition after the execution of a given program, what precondition does it suffice to assume, before the execution?

Through weakest preconditions a program gives rise to a (backward) predicate transformer that carries a given postcondition to the corresponding weakest precondition. This way of interpreting programs—sometimes called axiomatics semantics [14]—is in contrast to (forward) state transformer semantics where programs are understood as functions (possibly with branching or side effects) that carry input states/values to output ones.

Predicate Transformer Semantics and Quantum Mechanics The topic of weakest precondition and predicate transformer semantics is classic in computer science, in decades of foundational and practical studies. Recently, fresh light has been shed on their structural aspects: the same kind of interplay between dynamics and observations for quantum mechanics and quantum logic appears in predicate transformer semantics, as noted by one of the current authors—together with his colleagues [3, 15, 20]. This enabled them to single out a simple categorical scheme—called state-and-effect triangles—that is shared by program semantics and quantum mechanics.

On the program semantics side, the scheme of state-and-effect triangles allows the informal “duality” between state and predicate transformer semantics to be formalized as a categorical duality. Interestingly, the quantum counterpart of this duality is the one between the Schrödinger and Heisenberg pictures of quantum mechanics. In this sense the idea of weakest precondition dates back before Dijkstra, and before the notion of program.

State-and-effect triangles will be elaborated on in Section 2.1. We note at this stage that the term “effect” in the name refers to a notion in quantum mechanics and should be read as predicate in the programming context. In particular, it has little to do with computational effect.

In Search of Healthiness The question of healthiness conditions is one that is as old as the idea of weakest precondition [7]: it asks for an intrinsic characterization of those predicate transformers which arise as the (backward) interpretation of programs. One basic healthiness result is for nondeterministic programs. The result is stated, in elementary terms, as follows.

Theorem 1.1 (healthiness under the “may”-nondeterminism). Let \( R \subseteq X \times Y \) be a binary relation; it is thought of as a nondeterministic computation from \( X \) to \( Y \). This \( R \) induces a predicate transformer (wp for “weakest precondition”) \( \wp_X(R) : 2^Y \rightarrow 2^X \), defined by:

\[
\wp_X(R)(f)(x) = 1 \iff \exists y \in Y. (xRy \land f(y) = 1),
\]

for each \( f : Y \rightarrow 2 \) (thought of as a predicate and more specifically as a postcondition) and each \( x \in X \).

2. (Healthiness) Let \( \varphi : 2^Y \rightarrow 2^X \) be a function. The following are equivalent.

(a) The function \( \varphi \) arises in the way prescribed above. That is, there exists \( R \subseteq X \times Y \) such that \( \varphi = \wp_X(R) \).

(b) The map \( \varphi \) is join-preserving, where \( 2^Y \) and \( 2^X \) are equipped with (the pointwise extensions of) the order \( 0 \prec 1 \).

Here we interpret \( 0 \in 2 \) as false and \( 1 \in 2 \) as true, a convention we adopt throughout the paper.
There are many different instances of healthiness results. For example, the works \cite{12,13} study probabilistic computations in place of nondeterministic ones; the (alternating) combination of nondeterministic and probabilistic branching is studied in \cite{13}; and Dijkstra's original work \cite{1} deals with the (alternating) combination of nondeterminism and divergence. In fact it is implicit in our notation \( w_{\mathcal{P}} \) that there is a possible "must" variant of Theorem 1.1. In this variant, another predicate transformer \( w_{\mathcal{P}} \) is defined by

\[
w_{\mathcal{P}}(R)(f)(x) = 1 \iff \forall y \in Y. (x R y \Rightarrow f(y) = 1),
\]

requiring that every possible poststate must satisfy the postcondition \( f \). The corresponding healthiness result has it that the resulting predicate transformers are characterized by meet-preservation.

The goal of the current work is to identify a structural and categorical principle behind healthiness, and hence to provide a common ground for the existing body of healthiness results, also providing a methodology that possibly aids finding new results.

As a concrete instance of this goal, we wish to answer why join-preservation should characterize "may"-nondeterministic predicate transformers \( w_{\mathcal{P}} \) in Theorem 1.1. A first observation would be that the powerset monad \( P \)—that occurs in the alternative description \( R: X \rightarrow PY \) of a binary relation \( R \)—has complete join-semilattices as its Eilenberg-Moore algebras. This alone should not be enough though—the framework needs to account for different modalities, such as \( \Diamond \) ("may") vs. \( \Box \) ("must") for nondeterminism. (In fact it turns out that this "first observation" is merely a coincidence. See Section 2.1 later.)

**Our Contributions**

We shall answer to the above question of "categorical healthiness condition" by unifying two constructions—or recipes—of state-and-effect triangles.

- One recipe \cite{12,13} is called the *modality* one, whose modeling of situations like in Theorem 1.1 is centered around the notion of *monad*. Firstly, the relevant class of computations (nondeterministic, diverging, probabilistic, etc.) is determined by a monad \( T \), and a computation is then a function of the type \( X \rightarrow TY \). Secondly, the set \( \Omega \) of truth values (such as \( 2 \) in Theorem 1.1) carries a \( T \)-algebra \( \tau: \Omega \rightarrow \Omega \); it represents a modality such as \( \Diamond \) and \( \Box \).

- The other recipe \cite{12,13} is referred to as the *dual adjunction* one. It takes a dual adjunction \( \mathcal{C} \leftarrow \mathcal{D} \rightarrow \mathcal{D}^{op} \) as an ingredient; and uses two *comparison functors*—from a Kleisli category and to an Eilenberg-Moore category—to form a state-and-effect triangle, additionally exploiting \( D \)'s completeness assumption. One notable feature is that the resulting state-and-effect triangle is automatically "healthy"—this is because comparison functors are full and faithful.

Combining the two recipes we take advantages of both: the former provides a concrete presentation of predicate transformers by a modality; and the latter establishes healthiness. We demonstrate that many known healthiness results are instances of this framework.

The key to combining the two recipes is to interpret a monad \( T \) on \( Set \) in a category \( D \) that is other than \( Set \). For this purpose—assuming that the dual adjunction in the second recipe is given with a dualizing object—we introduce the notion of \( D \)-relative \( T \)-algebra and develop its basic theory. Notably the structure map of a \( D \)-relative \( T \)-algebra is given by a *monad map* from \( T \) to a suitable continuation-like monad (that arises from the aforementioned dual adjunction). This notion seems to be more than a tiny side-product of the current venture: we expect it to play an important role in the categorical model theory (see e.g. \cite{1,28,30}) where the equivalence between (finitary) monads and Lawvere theories is fundamental. See below for further discussions.

**Related and Future Work**

We believe the current results allow rather straightforward generalization (from ordinary, \( Set \)-based category theory) to enriched category theory \cite{29}. For example, the use of the \( [X,Y] \)-fold product \( \Omega^X \) can be replaced by the cotensor \( [X,Y] \). Doing so, and identification of this generalization's relevance in program logics, is left as future work.

The current theoretical developments are heavily influenced by Lawvere theories, another categorical formalization of algebraic structures that is (if finitary) equivalent to monads. In particular, our notion of relative algebra is aimed to be a (partial) answer to the oft-heard question: *A Lawvere theory can be interpreted in different categories. Why not a monad?* We intend to establish formal relationships in future work, possibly in an enriched setting. There the line of works on enriched Lawvere theories will be relevant \cite{16,23}.

The first observation in this direction is that: a monad \( T \) on \( Set \) gives rise to a (possibly large) "Lawvere theory" \( K(T)^{op} \); and then its "algebra" in a category \( D \) (with enough products) is a product-preserving functor \( K(T)^{op} \rightarrow D \).

What is definitely lacking in the current work (and in our previous work \cite{12,13}) is syntax for programs/computations and program logics. In this direction the work \cite{23} presents a generic set of inference rules—that is sound and relatively complete—for a certain class of monadic computations.

We are grateful to a referee who brought our attention to recent \cite{15}. Motivated by the modal logic question of equivalences between Kripke frames and modal algebras—possibly equipped with suitable topological structures—they are led to a framework that is close to ours. Their aim is a dual equivalence between a Kleisli category \( K(T) \) and a category of algebras \( D \), and our goal of healthiness (i.e. a full and faithful functor \( K(T) \rightarrow D^{op} \)) comes short of such only by failure of iso-denseness. Some notable differences are as follows. Firstly, in \cite{15} principal examples of a monad \( T \) is for nondeterminism, so that a Kleisli arrow is a relation, whereas we have probability and alternation as other leading examples. Secondly, in place of relative algebra (that is our novelty), in \cite{15} they use the notion of algebra that is syntactically presented with operations. Unifying the results as well as the motivations of the two papers is an exciting direction of future research. See also Remark 2.2.

Another closely related work \cite{23} studies healthiness from a domain-theoretic point of view. While it is based on syntactic presentations of algebras (differently from our monadic presentations), notable similarity is found in its emphasis on continuation monads. Its domain-theoretic setting—every construct is \( DC\text{-po} \)-enriched—will be relevant when we wish to accommodate recursion in our current results, too.

**Organization of the Paper**

We exhibited our leading example in Theorem 1.1. In Section 2 we describe its proof—in a categorical language—and this will motivate our general framework. After recalling the scheme of state-and-effect triangles in Section 3 in Section 4 we unify two known recipes for them to present a new relativized algebra recipe. The basic theory of relativized algebras is developed there, too. Section 5 is devoted to probabilistic instances of our framework. Finally in Section 6 we further extend the generic framework to accommodate alternating branching that involve two players typically with conflicting interests.

Some missing proofs are found in the appendix.

**Preliminaries and Notations**

We assume familiarity with basic category theory, from references like \cite{2,29}. We list some categories that we will use, mostly for fixing notations: the category \( Set \) of sets and functions; the category \( \text{Rel} \) of sets and binary relations; and the categories \( \text{CL}_V \) and \( \text{CL}_A \) of complete join- and meet-semilattices.
and join- and meet-preserving maps between them, respectively. Given a monad \(T\), its Eilenberg-Moore and Kleisli categories are denoted by \(\mathcal{EM}(T)\) and \(\mathcal{K}(T)\), respectively. Their definitions are found e.g. in [25,26].

Let \(S, T\) be monads on \(C\). The standard notion of monad map from \(S\) to \(T\) is defined by a natural transformation \(\alpha : S \to T\) that is compatible with the monad structures. For its explicit requirements see Appendix A.

We shall be using various “hom-like” entities such as homsets, exponentials, cotensors and so on; they are denoted by \(C(X, Y)\), \(Y^X\), \([X, Y]\), etc. For those entities we abuse the notations \(f^*\) and \(f_*\) and use them uniformly for the precomposition and postcomposition morphisms, such as:

\[
\begin{align*}
  f^* &= (-) \circ f : Z^Y \to Z^X \\
  f_* &= f \circ (-) : X^Z \to Y^Z
\end{align*}
\]

for \(f : X \to Y\). Another generic notation we will use for those hom-like entities is \((-)^\otimes\) for correspondences like:

\[
f : B \to A^X \quad /\!/ \quad f^\otimes : X \to A^B
\]

An example of such is via the universality of products:

\[
f : B \to A^X \quad \text{in a category } D \text{ with arbitrary products}
\]

\[
f^\otimes : X \to D(B, A)_\text{in Set}
\]

where \(A, B \in D, X \in \text{Set}\) and \(A^X\) is the \([X]\)-fold product of \(A\).

We shall use a somewhat unconventional notion of writing \(X_\sigma\) for an (Eilenberg-Moore) \(T\)-algebra \(x : TX \to X\). In our arguments the monad \(T\) is mostly obvious from the context, and this notional convention turns out to be succinct and informative.

2. Leading Example: Nondeterministic Computation and Join- (or Meet-) Preservation

In this section, as a leading example, we revisit the well-known healthiness result in Theorem 1.1 together with its “must” variant. We shall prove the results in an abstract categorical language, paving the way to the general and axiomatic modeling in Section 3.

2.1 “May”-Nondeterminism

In Section 1 regarding Theorem 1.1 we noted the coincidence between the healthiness condition (join-preservation) and Eilenberg-Moore \(\mathcal{P}\)-algebras (complete join-semilattices). This turns out to be a deceptive coincidence—the essence lies rather in a factorization of the powerset monad \(\mathcal{P}\) by a dual adjunction, as we shall describe.

We have a dual adjunction between \(\text{Set}\) and the category \(\text{CL}^\vdash\) of complete join-semilattices and join-preserving maps.

\[
\text{Set} \overset{2(\cdot)}\longrightarrow \text{CL}^\vdash \overset{2(\cdot)^\otimes}\longrightarrow (\text{CL}^\vdash)^\text{op}
\]

It is given by a dualizing object 2, in the “homming-in” manner:

\[
\begin{align*}
  \text{Set} &\to (\text{CL}^\vdash)^\text{op} : X \to 2^X \\
  [-, 2]: (\text{CL}^\vdash)^\text{op} &\to \text{Set} : L \to [L, 2]^\vdash
\end{align*}
\]

Here 2 is the poset \(\{0 < 1\}\), the poset \(2^X\) is the \([X]\)-fold product of 2, and \([L, 2]^\vdash = \text{CL}^\vdash(L, 2)\) is the set of join-preserving maps. This adjunction yields a monad \(X \to 2^X \to 2^{[X, 2]^\vdash}\) on \(\text{Set}\); the unit \(\eta\) of the monad \(2^{(-)}: 2^{[\cdot, 2]^\vdash}\) is defined by \(\eta_X(x) = f(x)\) and the multiplication \(\mu_X\) is \(\mu_X(Z) = \lambda f . \exists f \in \epsilon(f)\).

The following is the first key observation.

\[\footnote{Here a complete join-semilattice is a poset with arbitrary joins \(\lor\). It is well-known that in this case arbitrary meets \(\land\) exist, too; we say “join-” to indicate the notion of homomorphism we are interested in.}

**Lemma 2.1.** The monad \(2^{(-)}: 2^\vdash\) is isomorphic to the powerset monad \(\mathcal{P}\), with an isomorphism \(\sigma : \mathcal{P} \cong 2^{(-)}\) given by \(\sigma_X(S) = \lambda f . \forall s \in S . f(x)\).

The isomorphism in Lemma 2.1 puts us in the following situation.

\[
\begin{align*}
  \text{Set} &\to (\text{CL}^\vdash)^\text{op} : 2^\vdash \\
  K &\longrightarrow (\text{CL}^\vdash)^\text{op}
\end{align*}
\]

Here \(\sigma\) is the functor induced by the isomorphism \(\sigma\) in Lemma 2.1 and \(K\) is the comparison functor from the Kleisli adjunction as the “initial” factorization of a monad. See e.g. [25,26].

The second key observation is that the top composite \(\mathcal{K}(\sigma)\) of its action on arrows, precisely—coincides with the predicate transformer \(wp_\mathcal{P}\) in Theorem 1.1. Indeed, identifying a binary relation \(R \subseteq X \times Y\) with a function \(X \to \mathcal{P} Y\) and hence with a morphism \(X \to Y\) in \(\mathcal{K}(\mathcal{P})\), the action of \(\mathcal{K}(\sigma)\) can be concretely described as follows. The arrows on the second line are all in \(\text{Set}\).

\[
\begin{align*}
  \mathcal{K}(\mathcal{P}) &\quad \mathcal{K}(\mathcal{P}) \quad \mathcal{K}(\mathcal{P}) \\
  \mathcal{K}(\sigma) &\quad \mathcal{K}(\sigma) \quad \mathcal{K}(\sigma)
\end{align*}
\]

\[
\begin{align*}
  (X, R) &\quad (X, R) \quad (X, R) \\
  \mathcal{P} Y &\quad \mathcal{P} Y \quad \mathcal{P} Y
\end{align*}
\]

Unfolding the construction of the comparison functor \(K\), the function \(K(\sigma (\gamma) \circ R) : 2^Y \to 2^X\) in the end is presented as follows. Given \(f : Y \to 2\),

\[
K(\sigma (\gamma) \circ R(f)) = \lambda x . (\sigma (\gamma) \circ R(x))(f) = \lambda x . \{f(y) \mid x R y\} .
\]

This is nothing but the predicate \(wp_\mathcal{P}(R)(f) : X \to 2\) as defined in Theorem 1.1. Thus we have established

\[
(K \circ \mathcal{K}(\sigma))_{X, Y} = (wp_\mathcal{P})_{X, Y} : \mathcal{K}(\mathcal{P})(X, Y) \to \text{CL}^\vdash(2^Y, 2^X)
\]

for each \(X\) and \(Y\).

The last key observation is that a comparison functor is full and faithful in general. The action \((K \circ \mathcal{K}(\sigma))_{X, Y}\) is therefore bijective, hence so is \((wp_\mathcal{P})_{X, Y}\). This proves Theorem 1.1.

In the arguments above the key observations have been: (1) factorization of a monad via a dual adjunction (Lemma 2.1); (2) a monad map \(\sigma\) giving rise to a predicate transformer \(wp_\mathcal{P} = K \circ \mathcal{K}(\sigma)\); and (3) the role of a comparison functor \(K\) in particular that its fullness entails healthiness. Our general framework will be centered around these three notions (dual adjunction, monad map and comparison), with our notion of realizable algebra bonding them together.

**Remark 2.2.** In the above (and in Theorem 1.1) we established a full and faithful functor \(K(\mathcal{P}) \to (\text{CL}^\vdash)^\text{op}\). Cutting down its codomain, together with a well-known isomorphism \(\mathcal{K}(\mathcal{P})\) and the category \(\text{Rel}\) of sets and relations, gives us a dual equivalence \(\text{Rel} \cong (\text{CABA}^\vdash)^\text{op}\). Here \(\text{CABA}^\vdash\) is the category of complete atomic Boolean algebras and join-preserving maps between them. The last dual equivalence is a well-known one, found e.g. in [10, Section II.9] and [23].

Our principal interest in this paper—motivated by healthiness in program logics—is in a full and faithful functor. A dual equivalence, in contrast, is pursued typically in the context of modal logic (specifically for correspondences between modal algebras and relational frames); see e.g. [13]. The relevance of such equivalences in program logics would lie in identification of (not only programs but) appropriate state spaces that realize desired predicate transformers. Further investigation is future work.

**Remark 2.3.** For a join-semilattice \(L\) there is a poset isomorphism \(L^{\text{op}} \cong [L, 2]^\vdash\). This isomorphism \(L^{\text{op}} \cong [L, 2]^\vdash\) however tends to oversimplify arguments, often leading to errors in our experience.
For a similar reason we explicitly write the isomorphism \( \sigma \) in the situation \( [\text{5}] \).

### 2.2 “Must”-Nondeterminism

We noted after Theorem \([1.1]\) that a “must”-predicate transformer \( \mathsf{wp} \) can be conceived for nondeterministic computations, besides the “may” one \( \mathsf{wp}_i \). See \( [1] \). We shall briefly describe how this variant is supported by the same line of arguments as in Section \( [2.1] \).

The only difference from Section \( [2.1] \) is that we replace the dual adjunction \( [\text{2}] \) with

\[
\begin{array}{c}
\text{Set} \xrightarrow{\mathsf{wp}^\text{op}} \text{(CL}_{\Lambda}\text{)} \\
\text{Set} \xrightarrow{\mathsf{wp}} \text{(CL}_{\Lambda}\text{)} \\
\end{array}
\]

that is given, as before, by \( \mathsf{wp}^\text{op} : \text{Set} \to (\text{CL}_{\Lambda})^\text{op} ; X \mapsto 2X \) and \( \mathsf{wp} : (\text{CL}_{\Lambda})^\text{op} \to \text{Set} ; L \mapsto [L, 2]_{\Lambda} \). The new adjunction \( [\text{4}] \) factorizes the powerset monad \( \mathcal{P} \), as shown much like Lemma \( [\text{2.3}] \).

**Lemma 2.4.** A natural transformation \( \alpha' : \mathcal{P} \to [2^\text{op}, 2]_{\Lambda} \) given by \( \alpha'_K(S) = \lambda f. \bigwedge x \in S f(x) \) is an isomorphism of monads.

Now we are in a situation that is analogous to \( [\text{3}] \), in particular it gives us a composite \( K(\mathcal{P}) \xrightarrow{\alpha'} [2^\text{op}, 2]_{\Lambda} \xrightarrow{\mathsf{wp}} (\text{CL}_{\Lambda})^\text{op} \), where \( K' \) is a suitable comparison functor (that is full and faithful).

Working out the concrete definitions we easily observe that

\[
(K' \circ \alpha')_{X,Y} = (\mathsf{wp}^\text{op}_{\mathcal{P}})(X,Y) = \lambda R. \lambda f. \lambda x. \bigwedge \{ f(y) \mid x R y \}
\]

\( : K(\mathcal{P})(X,Y) \to (\text{CL}_{\Lambda})(2^X, 2^Y) \).

This leads to the following analogue to Theorem \( [\text{1.1}] \).

**Theorem 2.5** (healthiness under the “must”-nondeterminism). Let \( \varphi : 2^X \to 2^Y \) be a function. The following are equivalent.

1. There exists \( R \subseteq X \times Y \) such that \( \varphi = \mathsf{wp}^\text{op}_{\mathcal{P}}(R) \). Here \( \mathsf{wp}^\text{op}_{\mathcal{P}} \) is from \( [\text{4}] \).
2. The map \( \varphi \) is meet-preserving.

### 3. State-and-Effect Triangles

We continue Section \( [2] \) and present a general and categorical framework for establishing (possibly partial) healthiness results. We shall first recall the scheme of state-and-effect triangles \( [\text{1.2}, \text{1.3}] \), and two of its “recipes” \( [\text{1.5}, \text{1.6}] \) which are relevant here.

#### 3.1 State-and-Effect Triangles, in Quantum Logic and Program Logic

In the previous work \( [\text{1.3}, \text{1.9}] \) situations called state-and-effect triangles have been found to be fundamental in various examples of predicate transformers. More specifically, the triangular scheme dictates how computations, forward state-transformer semantics and backward predicate-transformer semantics are organized, in terms of categories, functors and a dual adjunction.

\[
\begin{array}{c}
\text{computations} \\
\text{state transformers} \\
\text{(or “states”)}
\end{array}
\xleftarrow{\gamma} 
\begin{array}{c}
\text{predicate transformers} \\
\text{Predicate semantics} \\
\text{op}
\end{array}
\]

The name “state-and-effect triangle” comes from the operational study of quantum logics; here the term “state” refers to a state of a quantum system—possibly a mixed state, i.e. a probabilistic ensemble \( \sum_i \psi_i \langle \psi_i | \rangle \) over pure states—and the term “effect” refers to the notion in quantum theory, i.e. a convex-linear map from (quantum) states to the values in the interval \([0, 1]\). The dual adjunction at the top of \( [\text{5}] \), in such quantum settings, represents the duality between the so-called Schrödinger and Heisenberg pictures of quantum mechanics.

In our current context of program semantics and program logics, the term “state” in the state-and-effect triangles is more intuitively understood as superposed states, and the term “effect” is understood as predicates. See \( [\text{5}] \). We emphasize, in particular, that the term “effect” in the state-and-effect triangles refers to the quantum notion and has nothing to do with computational effects in functional programs.

It is interesting that the same categorical scheme underlies quantum logics and program logics. This is essentially because they share the combination of logic and dynamics. For example, in quantum mechanics predicates (or “effects”) have a distinctively operational flavor—measurements cause projection of quantum states.

#### An Example

Let us exhibit an example. It is based on the constructions in Section \( [\text{2.1}] \), although the triangle itself was not explicit.

\[
\begin{array}{c}
\mathcal{CL}^\text{op} \xrightarrow{\mathsf{wp}^\text{op}} \mathcal{EM}(\mathcal{P}) \xrightarrow{R} \mathcal{EM}(\mathcal{P}) \\
\mathsf{wp}^\text{op} \xrightarrow{\mathcal{K}\ell}(\mathcal{P}) \xrightarrow{\mathcal{K}} \mathcal{EM}(\mathcal{P}) \xrightarrow{\text{with } \mathsf{wp}_{\mathcal{K}} \cong \mathcal{L.K.} (6)}
\end{array}
\]

The fact that \( \mathsf{wp}_{\mathcal{K}} \) is a contravariant functor means that the (predicate transformer) semantics expressed by it is a backward one. The comparison functor \( \mathcal{K} \) from the Kleisli category \( \mathcal{K}\ell(\mathcal{P}) \) to the Eilenberg-Moore category \( \mathcal{EM}(\mathcal{P}) \) acts concretely as follows.

\[
\mathcal{K}(X) = \{ U \subseteq X \to \mathcal{P}X \mid X \xrightarrow{\mathcal{K}} \mathcal{EM}(\mathcal{P}) \}
\]

The intuition is that \( U \subseteq \mathcal{P}X \) is a “superposed state” that indicates which states are possibly the current state. The triangle \( \mathcal{K} \) stipulates that \( \mathsf{wp}_{\mathcal{K}} \) factors through \( \mathcal{K} \). Finally, the healthiness condition—that the image of \( \mathsf{wp}_{\mathcal{K}} \) is characterized by join-preservation—translates to the statement that the functor \( \mathsf{wp}_{\mathcal{K}} \) in \( [\text{6}] \) is full.

Instances of state-and-effect triangles abound, from quantum mechanics to computations with various notions of branching. See e.g. \( [\text{18}, \text{19}] \); later in this paper there will be further examples, too.

#### 3.2 The Dual Adjunction Recipe

One “recipe” for state-and-effect triangles is introduced in \( [\text{18}] \); we refer to it as the dual adjunction recipe. It works as follows.

- One starts with a monad \( T \) on a category \( C \), and its “factorization”

\[
\begin{array}{c}
\xrightarrow{G} \\
\xleftarrow{F} \\
\end{array}
\]

\[
\xrightarrow{T = G F} \xrightarrow{D^\text{op}} \xrightarrow{C}
\]

We assume that the adjunction is contravariant, for the sake of argument.

- As is well-known (see e.g. \( [\text{6}, \text{29}] \)), there arise two comparison functors \( K \) and \( R \), induced by the “universality” of the Kleisli and Eilenberg-Moore constructions respectively, as below.

\[
\begin{array}{c}
\xrightarrow{K} \xrightarrow{\mathcal{K}\ell(T)} \xrightarrow{\mathcal{K}} \xrightarrow{\mathcal{EM}(T)} \\
\xrightarrow{R} \xrightarrow{\mathcal{L.K.}} \xrightarrow{\mathcal{EM}(T)} \xrightarrow{\text{with } K \xrightarrow{\mathcal{K}\ell(T)} \xrightarrow{\mathcal{EM}(T)}}
\end{array}
\]

\footnote{The comparison functor \( K : \mathcal{K}\ell(T) \to \mathcal{EM}(T) \) here is different from the one \( K : \mathcal{K}\ell(P) \to \mathcal{EM}(P) \) in Section \( [\text{2.1}] \) although they arise from the same “universality” of \( \mathcal{K}\ell(P) \). Using the same notation \( K \) will not cause confusion.}
We organize the three categories on the top in the previous diagram 6 as a triangle. This gives rise to the following situation.

\[
\begin{array}{ccc}
D^{\text{op}} & \xrightarrow{R} & K(\mathcal{M}(T)) \\
\downarrow L & & \downarrow R \circ K \\
\mathcal{K}(T) & \xrightarrow{\tau} & \mathcal{M}(T)
\end{array}
\]

A left adjoint to \( R \) will complete a state-and-effect triangle. For its existence we assume equivalent categories in \( D \) (hence coequalizers in \( D^{\text{op}} \)) and use a variant of Beck’s monad coherence theorem.

A formal statement is as follows.

**Theorem 3.1** (the dual adjunction recipe [18, Theorem 1]). Assume an adjunction \( \mathcal{D} \) and a monad \( T = GF \). Assume further that the category \( \mathcal{D} \) has equalizers of reflexive pairs. Then we have a situation

\[
\begin{array}{ccc}
D^{\text{op}} & \xrightarrow{R} & K(\mathcal{M}(T)) \\
\downarrow L & & \downarrow R \circ K \\
\mathcal{K}(T) & \xrightarrow{\tau} & \mathcal{M}(T)
\end{array}
\]

where \( L R K \cong K \). Moreover \( K \) is full and faithful.

*Proof.* The constructions have already been sketched in the above; see [18] for details. That the comparison functor \( K \) is full and faithful is standard; see e.g. [6, 29].

Note that the dual adjunction recipe in Theorem 3.1 automatically derives soundness (that \( K \) is full and faithful). This recipe, though powerful, is also restrictive: it obviously cannot be used to derive a non-full predicate transformer semantics. Furthermore, the example in [6] cannot be directly derived using the dual adjunction recipe: to do so we would need a slight generalization of the recipe in (6) cannot be directly derived using the dual adjunction recipe

\[\tau = \Omega \Rightarrow \text{the equality "combined" recipe later in Section 4 will address these issues.}\]

Recipe to do so we would need a slight generalization of the recipe in (6) cannot be directly derived using the dual adjunction recipe

\[\tau = \Omega \Rightarrow \text{the equality "combined" recipe later in Section 4 will address these issues.}\]

Note that the dual adjunction recipe in Theorem 3.1 automatically derives soundness (that \( K \) is full and faithful). This recipe, though powerful, is also restrictive: it obviously cannot be used to derive a non-full predicate transformer semantics. Furthermore, the example in [6] cannot be directly derived using the dual adjunction recipe: to do so we would need a slight generalization of the recipe in (6) cannot be directly derived using the dual adjunction recipe

\[\tau = \Omega \Rightarrow \text{the equality "combined" recipe later in Section 4 will address these issues.}\]

Note that the dual adjunction recipe in Theorem 3.1 automatically derives soundness (that \( K \) is full and faithful). This recipe, though powerful, is also restrictive: it obviously cannot be used to derive a non-full predicate transformer semantics. Furthermore, the example in [6] cannot be directly derived using the dual adjunction recipe: to do so we would need a slight generalization of the recipe in (6) cannot be directly derived using the dual adjunction recipe

\[\tau = \Omega \Rightarrow \text{the equality "combined" recipe later in Section 4 will address these issues.}\]

Note that the dual adjunction recipe in Theorem 3.1 automatically derives soundness (that \( K \) is full and faithful). This recipe, though powerful, is also restrictive: it obviously cannot be used to derive a non-full predicate transformer semantics. Furthermore, the example in [6] cannot be directly derived using the dual adjunction recipe: to do so we would need a slight generalization of the recipe in (6) cannot be directly derived using the dual adjunction recipe

\[\tau = \Omega \Rightarrow \text{the equality "combined" recipe later in Section 4 will address these issues.}\]

3.3 The Modality Recipe

Here we review the other previous recipe that we will be based on; it is derived from the framework of monadic predicate transformers from [12, 13]. It is centered around the notion of modality—given as an Eilenberg-Moore \( T \)-algebra \( \tau : T \Omega \to \Omega \) over the domain \( \Omega \) of truth values—and interprets functions of the type \( X \to TY \), that is, \( T \)-branching computations.

**Definition 3.2** (\( P^T \)). Let \( \tau : T \Omega \to \Omega \) be a \( T \)-algebra; it is called a modality. It induces a functor \( P^T : K(\mathcal{M}(T)) \to \text{Set}^{\text{op}} \) that is defined by:

\[P^T(X) = \Omega^X \text{ and } P^T(Y \in K(\mathcal{M}(T))) = \left( \Omega^Y \xrightarrow{\tau^Y} \Omega^{TY} \xrightarrow{\tau^T} \Omega^X \right).
\]

Recall that \( \tau^T \) denotes precomposition of \( \tau \). Here \( \tau^T \) is the extension map that \( \tau^T \) denotes precomposition of \( \tau \). Here \( \tau^T \) is the extension map that

\[Y \to \Omega \text{ in } \text{Set} \quad / \quad (TY \xrightarrow{\tau^T} \Omega) \xrightarrow{\tau} \Omega^X \text{ in } \mathcal{M}(T).
\]

Note that \( P^T(f) \) can be alternatively described as follows. Given \( f : X \to TY \) (a computation) and \( h : Y \to \Omega \) (a postcondition), the function \( P^T(f)(h) : X \to \Omega \) (the weakest precondition) is the composite \( X \xrightarrow{h} TY \xrightarrow{T\tau} T\Omega \xrightarrow{\tau} \Omega \).

\[\tau = \Omega = \bigwedge : P^2 \to 2 \quad \text{and} \quad \tau_0 = \bigvee : P^2 \to 2,
\]

where inf and sup refer to the order \( 0 < 1 \). The former is the "must" modality, whereas the latter is the "may" one.

A modality \( \tau : T\Omega \to \Omega \) gives rise to an instance of the state-and-effect triangle.

**Theorem 3.4** (the modality recipe, [12, 13]). Let \( \tau : T\Omega \to \Omega \) be an Eilenberg-Moore algebra. It gives rise to the following situation, with \( P^T \) factorized as \( P^T \cong [-, \Omega]\circ \tau \circ K \).

\[\begin{array}{ccc}
\text{Set}^{\text{op}} & \xrightarrow{P^T} & \mathcal{M}(T) \\
\downarrow K(\mathcal{M}(T)) & & \downarrow \mathcal{M}(T)
\end{array}\]

\[\begin{array}{cccc}
\tau & \cong & \Omega & \cong \\
\Delta & \cong & \Omega & \cong \\
\end{array}
\]

Here \( P^T \) is from Definition 3.2 and \( K \) is the comparison functor. The dual adjunction on the top is induced by the dualizing object \( \Omega \).

Specifically, the functor \( [-, \Omega]^\tau = \mathcal{M}(T)(-, \Omega) \) is the homset functor; and \( \Omega_\tau \) for a set \( X \) is the \( X \)-fold product of the \( T \)-algebra \( \Omega \). The latter is explicitly given by the transpose of

\[\begin{array}{cccc}
X \xrightarrow{\text{id}^X} & \text{Set}(\Omega^X, \Omega) & \xrightarrow{T\Omega^X, \Omega} & \text{Set}(T(\Omega^X), \Omega) \\
\end{array}\]

where \( \text{id}^X \) is the transpose of the identity \( \text{id} : \Omega^X \to \Omega^X \) and \( T\Omega^X, \Omega \) is the action of \( T \) on homsets.

4. The “Relative Algebra” Recipe for State-and-Effect Triangles

We unify the two recipes (dual adjunction and modality) to render a general one. It is called the relative algebra recipe, because of the role played by our notion of relative Eilenberg-Moore algebra.

4.1 Relative Eilenberg-Moore Algebra

We shall introduce the notion of relative Eilenberg-Moore algebra for a monad \( T \) on \( \text{Set} \) and a category \( D \) with small products. Notably its carrier object is an object of \( D \); hence what we do is arguably to interpret a monad \( T \) on \( \text{Set} \) over a different category \( D \).

**Remark 4.1.** We expect further generalization is possible. The developments below bear a strong enriched flavor; and we envisage a general framework where a \( \mathcal{V} \)-monad \( T \) on an SMCC \( \mathcal{V} \) is interpreted over an arbitrary \( \mathcal{V} \)-enriched category \( D \). Working out the precise statements is future work.

Let \( D \) be a category with arbitrary products. For each object \( A \in D \) there is a dual adjunction, with \( A \) playing the role of a dualizing object.

\[\begin{array}{cccc}
\text{Set} & \xrightarrow{\text{Set}(\cdot, A)} & D^{\text{op}} & \xrightarrow{D(\cdot, A)} \\
\end{array}
\]

\[\begin{array}{cccc}
B \xrightarrow{A} & A^X \text{ in } D \\
X \xrightarrow{D(B, A)} & \text{Set} \quad (10)
\end{array}
\]

7In [12, 13] the framework is \( \text{Pos} \)-based rather than \( \text{Set} \)-based. Here for simplicity we present a \( \text{Set} \)-based framework; our generalization later will account for the \( \text{Pos} \)-based one as an instance.

8Recall our notational convention that an Eilenberg-Moore algebra \( \tau : T\Omega \to \Omega \) is denoted by \( \Omega \). See Section 3.
This is much like [2]; recall that $A^X$ denotes the $|X|$-fold product of $A \in D$ (i.e. a cotensor, in the enriched terms).

This adjunction induces a continuation-like monad $D(A(-), A)$.

**Definition 4.2 (D-relative T-algebra).** Let $T$ be a monad on $\text{Set}$, and $D$ be a category with small products. A $D$-relative $T$-algebra is a pair $\left( A, \alpha : T \to D(A(-), A) \right)$ of an object $A \in D$ and a monad map $\alpha$ from $T$ to the continuation-like monad $D(A(-), A)$ from [10].

A morphism of $D$-relative $T$-algebras, say from $(A, \alpha)$ to $(B, \beta)$, is a morphism $f : A \to B$ in $D$ such that the following diagram commutes for each $X \in D$.

$$
\begin{array}{ccc}
A^X & \xrightarrow{\alpha^X} & A^{TX} \\
\downarrow{\beta^X} & & \downarrow{f^*} \\
B^X & \xrightarrow{\beta^X} & B^{TX}
\end{array}
$$

Here $\alpha^X$ is induced canonically from $\alpha : TX \to D(A^X, A)$, via the bijective correspondence in [10] (the universality of products).

**Proposition 4.3.** Let $C$ be a complete category and $T$ be a monad on $C$. For each object $A \in C$, there is a canonical bijective correspondence between: 1) $T$-algebras $\hat{\alpha} : TA \to A$ with $A$ being their carrier objects; and 2) monad maps $\alpha : T \to A^C(-, A)$.

The concrete correspondence is given by: $\alpha_X = \{ TX \xrightarrow{\alpha} TX \}_{X \in C(X, A)}$ and $\hat{\alpha} = (TA \xrightarrow{\alpha} A^{TX})_{\pi \circ \alpha}$.

Moreover, $f : A \to B$ is a $T$-algebra morphism from $(A, \hat{\alpha})$ to $(B, \beta)$ if and only if the following diagram commutes.

$$
\begin{array}{ccc}
C(X, A) & \xrightarrow{\alpha^X} & C(TX, A) \\
\downarrow{\beta^X} & & \downarrow{f^*} \\
C(X, B) & \xrightarrow{\beta^X} & C(TX, B)
\end{array}
$$

Here $\alpha^X$ is defined analogously to Definition [2].

This result and Definition [2] yields the following. There we also need the isomorphism $\text{Set}(A^X, A) \cong \text{Set}(X, A)$ that identifies homsets and cotensors. This is available since $\text{Set}$ is self-enriched.

**Corollary 4.4.** Let $T$ be a monad on $\text{Set}$. We have an isomorphism $\mathcal{EM}(T) \cong \mathcal{EM}(T')$.

**Remark 4.5.** There is a Lawvere theory-like intuition behind Proposition 4.3 (from which we came up with Definition 2.2). Given an algebra $\hat{\alpha} : TA \to A$, the corresponding monad map $\alpha_X : TX \to A^{TX}$ is understood as: “given an algebraic/syntactic term $t \in TX$ with variables from $X$, and a valuation $V : X \to A$, the element $\alpha_X(t)(V) \in A$ is how $t$ is interpreted under $V$ (interpreting variables) and $\hat{\alpha}$ (interpreting algebraic operations).”

Lawvere theories are interpretation-free—hence “syntactic”—presentations of algebraic structures. They are therefore subject to interpretation in any category $D$ with finite products; see e.g. [13]. In contrast, monads—although their equivalence to Lawvere theories is well-known, see e.g. [28]—are always tied to their base category. Our notion of $D$-relative $T$-algebra is how to “interpret” the algebraic structure embodied as a monad $T$ (on $\text{Set}$) on another category $D$.

**Example 4.6.** Let $\mathcal{EM}$ denote the list monad on $\text{Set}$, whose Eilenberg-Moore algebras are monoids. For $D = \text{Top}$, the category of topological spaces and continuous maps, the category $\mathcal{EM}(\text{Set}; \text{Top})$ is exactly the category of topological monoids. Similarly for $D = \text{Pos}$, the category of posets and monotone maps, the category $\mathcal{EM}(\text{Set}; \text{Pos})$ is that of ordered monoids. The same phenomena can be observed for many other monads $T$ and categories $D$.

We exhibit a change-of-base result. In the case of Lawvere theories, we can map a $D$-model of a theory to a $D'$-model along a (finite) product-preserving functor $H : D \to D'$.

**Proposition 4.7.** Let $D, D'$ be categories with small products and $H : D \to D'$ be a product-preserving functor. Then $H$ canonically lifts to a functor $\mathcal{EM}(T; D) \to \mathcal{EM}(T; D')$, with

$$
\begin{array}{ccc}
\mathcal{EM}(T; D) & \xrightarrow{\mathcal{EM}(H)} & \mathcal{EM}(T; D') \\
\downarrow{U_D} & & \downarrow{U_{D'}} \\
D & \xrightarrow{H} & D'
\end{array}
$$

The functor $\mathcal{EM}$ preserves arbitrary products. Moreover, if $H$ is faithful, so is $\mathcal{EM}$.

**4.2 A State-and-Effect Triangle via Relative Algebras**

In the “modality” recipe in Section 3.3 the key to a dual adjunction between $\text{Set}$ and $\mathcal{EM}(T)$ was to use a $T$-algebra as a dualizing object. We shall now extend this from $\text{Set}$ to a general category $D$, using a $D$-relative $T$-algebra in place of a $T$-algebra.

**Theorem 4.8** (the relative algebra recipe). Let $\Omega_D \in D$ be an object in a complete category $D$, and

$$
\Omega = \left( \Omega_D, \tau : T \to D(\Omega_D^{\tau}, \Omega_D) \right)
$$

be a $D$-relative $T$-algebra. This yields a state-and-effect triangle:

$$
\begin{array}{ccc}
\text{List}(T) & \xrightarrow{\mathcal{EM}(\tau)} & \mathcal{EM}(T) \\
\downarrow{\mathcal{EM}(\Omega)} & & \downarrow{U_D} \\
\text{List}(\tau) & \xrightarrow{\mathcal{EM}(\Omega_D)} & \text{List}(\Omega_D)
\end{array}
$$

Here $K$ is the comparison functor. The other three functors are defined as follows.

- $([\cdot], \mathcal{EM}(\Omega_D))$ for each $D \in D$, the object $[D, \mathcal{EM}(\Omega_D)]$ is the set $D(D, \mathcal{EM}(\Omega_D))$ equipped with a $T$-algebra structure $\xi$ defined by

$$
T(D(D, \Omega_D)) \xrightarrow{\tau} D(\Omega_D^{\tau}, \Omega_D) \xrightarrow{\mathcal{EM}(\Omega)} D(D, \Omega_D) .
$$

The last arrow precomposes $\mathcal{EM}(\tau) : D \to D(\Omega_D^{\tau}, \Omega_D)$. For a $D$-algebra morphism $k : D \to E$, the $T$-algebra morphism $k^* = [k^*, \mathcal{EM}(\Omega_D)] : [D, \mathcal{EM}(\Omega_D)] \to [E, \mathcal{EM}(\Omega_D)]$ is defined by the precomposition $k^* = D(E, \mathcal{EM}(\Omega_D)) \to D(D, \mathcal{OE})$ between the carrier sets.

We speculate that, when a monad $T$ is bounded, our notion of relative $T$-algebra coincides with the models of the Lawvere theory $\mathcal{L}_T$ induced by $T$. We note however that relative $T$-algebras can be defined even for unbounded $T$. We need this feature, too, since we deal with unbounded monads like the powerset monad $\mathcal{P}$. 

\[\text{Page 6 of 2016/5/3}\]
Given a T-algebra $A_α = \langle A, a: TA → A \rangle$, the D-object $[A_α, \Omega]_T$ is defined as the equalizer of $\alpha^∗: \Omega^A_2 \rightrightarrows \Omega^A_1$; see the top row of (15) below. Given a morphism $f: A_α → B_β$ of $T$-algebras, a $D$-morphism $f^∗ = [f, \Omega]_T: [B_β, \Omega]_T → [A_α, \Omega]_T$ is induced by the universality of an equalizer, as below:

$$[A_α, \Omega]_T \xrightarrow{eq} \Omega^A_2 \xrightarrow{\alpha^∗} \Omega^A_1 \xrightarrow{\pi_1} [B_β, \Omega]_T \xrightarrow{f^∗} \Omega^B_1 \xrightarrow{\pi_2} \Omega^B_2.$$  \hspace{1cm} (15)

Remark 4.9. The notations $[A_α, \Omega]_T$ and $[D, \Omega]_D$ are sort of abusively, because $\Omega$ is not a $T$-algebra or a $D$-object. The notations reflect the dual nature of a $D$-relative $T$-algebra, in the sense that it is precisely described in the above. Later in Section 4.3 we develop this point (and notations) more systematically.

The third relative algebra recipe in Theorem 4.8 combines the previous two recipes. Indeed, the modality recipe in Section 4.2 is a special case, much like usual $T$-algebras are special cases of relative $T$-algebras (Corollary 4.4). The current generalization allows us to have a category $D$—possibly other than $Set$—at the top-left of a state-and-effect triangle. See (9) and (13).

Regarding the relationship to the dual adjunction recipe in Section 4.2—that automatically ensures healthiness, see Theorem 3.1—we have the following cornerstone result towards analysis of general healthiness conditions. Note that the assumption $T = GF$ in (7) translates, in the context of Theorem 4.8, to the condition that $\tau: T → D(\Omega^\circ_2, \Omega^\circ_2)$ is the identity.

Theorem 4.10 (categorical (partial) healthiness condition). Let $X, Y ∈ Set$. In the setting of Theorem 4.8:

- if $Γ_T: TY → D(Ω^Y_2, Ω^Y_2)$ is injective, the functor $P^∗$’s action $P^∗_{XY}: KI(T)(X, Y) → D(Ω^X_2, Ω^Y_2)$ is injective;
- if $Γ_T$ is surjective, so is $P^∗_{XY}$.

It follows that, if $Γ_T: T → D(Ω^\circ_2, Ω^\circ_2)$ is a natural isomorphism, the functor $P^∗$ is full and faithful.

The last corollary accounts for (part of) Theorem 1.1 generalizing the arguments in Section 4.2. The proof of Theorem 4.10 is like the proof of Theorem IV.3.1, giving the correspondence between monic/epic (co)units and fullness/faitiness of adjoints.

4.3 Relative Algebras over a Concrete Category

We have obtained the third, unified recipe for state-and-effect triangles in Theorem 4.8 together with a general healthiness result (Theorem 4.10). The remaining piece towards the full coverage of healthiness results like Theorem 1.1 is: how specific predicate transformer semantics—specified by a concrete modality (like $wp$)—is related to constructs in the general recipe in Theorem 4.8.

To fill this missing piece we shall study a situation where $D$ is concrete, by which we specifically mean that: 1) we have a faithful “forgetful” functor $V: D → Set$; and 2) the functor $V$ preserves small limits. Examples are: the Eilenberg-Moore category $EM(T)$ of a monad $T$ on $Set$; the categories $Top$ and $Pos$; and other categories of “sets with additional structures.”

By Proposition 4.4 and Corollary 4.4.1 the functor $V: D → Set$ lifts to $\overline{\cdot}: EM(T; D) → EM(T)$. This gives rise to the following.

$$D \xleftarrow{V} EM(T; D) \xrightarrow{\overline{\cdot}} EM(T) \xrightarrow{U_D} Set \xleftarrow{U} D.$$  \hspace{1cm} (16)

The diagram (16) is skewed (compared to (12)) to convey the intuition that: an object in $EM(T; D)$ is a set equipped both with a $D$-structure and with a $T$-algebra structure, in a compatible manner. The developments below are aimed at formalizing this intuition.

Notation 4.11 $(A, A_δ, A_α, Λ)$. From now on we adopt a notational convention of writing: $A$ for a set; $A_δ$ for an object in $D$ such that $V(A_δ) = A$; $A_α$ for a $T$-algebra $α: TA → A$ (hence $U(A_α) = A$); and $Λ ∈ EM(T; D)$ for a relative $T$-algebra such that $VU_Λ = A$. See below, and compare it to (16).

$$A_δ \xrightarrow{U_δ} A_α \xleftarrow{U} Λ \xrightarrow{π_Λ} VU_Λ \xrightarrow{V_Λ} A.$$  \hspace{1cm} (17)

This convention, though adumbrated confusing at first sight, follows some literature on dualities (such as (6)) and allows us to describe our technical developments in a succinct manner. We emphasize that fixing $A$ does not fix $A_δ, A_α$ or $Λ$.

The following characterization of $D$-relative $T$-algebras—it assumes that $D$ is concrete—embodies the intuition that they are “$T$-algebras whose algebraic structures are compatible with $D$-structures.” This is much like a topological monoid is a monoid whose multiplication is continuous; see Example 4.6.

Proposition 4.12. Let $A_δ = \langle A, δ: TA → A \rangle$ be a $T$-algebra and $A_α ∈ D$ be such that $V(A_α) = A$. The following are equivalent.

1. There exists a $D$-relative $T$-algebra $Λ$ such that $V(Λ) = A_α$ and $U_Λ Λ = A_δ$. See below:

$$A_δ \xrightarrow{U_δ} A_α \xleftarrow{U} Λ \xrightarrow{π_Λ} VU_Λ \xrightarrow{V_Λ} A.$$  \hspace{1cm} (18)

2. The following lifting condition holds: the monad map $α: T → Set(A_α, A)$ induced by $δ$ (Proposition 4.3) factors through $V: D(A_δ, A) → Set(A_α, A)$, as in

$$\xrightarrow{π_Λ} \downarrow \xrightarrow{V_Λ} TX \xrightarrow{α_X} Set(A_α, A).$$  \hspace{1cm} (19)

The latter is more concretely stated as follows: for each $X ∈ Set$ and $t ∈ TX$, the function $α_X(t): A_α → A$ lifts to a $D$-morphism $π_Λ(t): A_δ → A_α$.

If the conditions hold, we say that $A_α$ and $A_δ$ are compatible. \hspace{1cm} □

This result means: to render $A ∈ Set$ into a $D$-relative $T$-algebra, it suffices to find a $T$-algebra structure and a $D$-structure and then to check the above lifting condition. The lifting condition in Proposition 4.12 is a direct generalization of the monotonicity condition (precisely its pointwise version) used in (12, 13); when $D = Pos$ we get the original monotonicity condition.

Under a further assumption that $T$ is finitary, we can restrict the required check to finite sets.

Proposition 4.13. Assume the setting of Proposition 4.12 and $T$ is finitary. $A_δ$ and $A_α$ are compatible if and only if the lifting condition (19) holds for any natural number $n$ in place of $X$. \hspace{1cm} □

It follows that, in case the monad $T$ is induced by some known algebraic specification $(Σ, E)$, checking the lifting condition can further be restricted to “basic operations” $σ ∈ Σ$. For instance, a monoid $(Ω, *, e)$ and $Ω_2 ∈ D$ satisfy the lifting condition if and only if both the multiplication $*: Ω → Ω$ and the unit $e: 1 → Ω$ lift to $D$-morphisms.
We can exploit the construction in Theorem 4.8 when \( D \) is concrete. Assume we have a functor \( V : D \to \text{Set} \) that is faithful and limit-preserving. Given a \( D \)-object \( D \), the \( T \)-algebra \([\Omega, D]_{\Omega} \) can be understood as a homset \( D(D, D) \) with the pointwise \( T \)-algebra structure along \( \Omega \), i.e. we regard it as a subalgebra of the product \( \Omega^{y_1} \). Similarly we can see \([\Lambda, D]_{\Omega} \) as the \( D \)-object whose "carrier set" is \( \mathcal{EM}(T)(\Lambda, \Omega) \) with the pointwise \( \Omega \)-structure. It is precisely stated as follows.

**Proposition 4.14.** In the situation of Theorem 4.8 we have \( U \circ [\Omega, D]_{\Omega} = D(D, D) \).

Furthermore, assume we have a limit-preserving functor \( V : D \to \text{Set} \) and let \( \Omega^T = \Omega \). Then \([\Omega, D]_{\Omega} \) is a \( T \)-subalgebra of \( \Omega^{D} \) meaning that the algebraic structure of the former is a pointwise extension of \( \tau \)-and we have \( V \circ [\Omega, D]_{\Omega} = \mathcal{EM}(T)(- : \Omega) \).

In the latter setting of Proposition 4.14 where \( D \) is “concrete” with \( V : D \to \text{Set} \), let \( \Omega = (\Omega, \tau, \Omega) \) be its underlying \( T \)-algebra. These data give rise to two different predicate transformer semantics: one is \( \mathcal{P}^T : \mathcal{K}(T) \to \mathcal{D} \) via the relative algebra \( \eta \) via the relative algebra recipe (Theorem 4.8); and the other is \( \mathcal{P}^T : \mathcal{K}(T) \to \mathcal{D} \) from the algebraic structure of \( \Omega \) via the modality recipe (Theorem 4.4). Between these we have the following correspondence, as we announced in the beginning of Section 4.3.

**Proposition 4.15.** In the situation of Theorem 4.8 additionally assume that \( V : D \to \text{Set} \) is a faithful and limit-preserving functor. Then, in terms of the above notations we have \( \mathcal{P}^T \cong V \circ \mathcal{P}^T \).

We have required \( V : D \to \text{Set} \) to be limit-preserving; in fact it mostly suffices to assume product-preservation. In that case the only thing that fails is the isomorphism \( V \circ [\Omega, D]_{\Omega} \cong \mathcal{EM}(T)(- : \Omega) \) in Proposition 4.14, which is the result that connects the top-right corner of the two (relative algebra and modality) triangles. Proposition 4.15 is only concerned about the top-left and bottom corners, and survives under \( V \) that preserves only products.

### 4.4 Finitary Predicate Transformers

The categorical results so far for finiteness (Theorem 4.8 and 4.10) are not enough for some instances of finiteness results, as we will see in the examples of Section 5. Specifically, besides the structural aspects covered by those results, we need to take account of sizes.

Throughout Section 4.4 we adopt the setting in Proposition 4.15 i.e. the relative algebra recipe with a faithful and limit-preserving \( V : D \to \text{Set} \). In particular we have a \( D \)-relative \( T \)-algebra \( \Omega = (\Omega, \tau, \Omega) \) as its underlying \( T \)-algebra. Recall the correspondence between \( \Omega \) and a monad \( \pi \) (Proposition 4.3).

The key observation is the following lemma.

**Lemma 4.16.** Let \( X \) be a set and \( t \in TX \). If \( T \) is finitary, the map \((\tau x)(t) : \Omega^X \to \Omega^{X t} \) factors through a precomposition map \( s \) and \( \pi \) factors through \( s \).

We formulate a size restriction on predicate transformers.

**Definition 4.17** (finitary predicate transformer). A predicate transformer \( \varphi : \Omega^Y \to \Omega^X \) is finitary if for each \( x \in X \) there exists a finite subset \( s : Y \to X \) such that \( \pi_s \circ \varphi \) factors through the precomposition \( s \). See below.

\[
\begin{array}{ccc}
\Omega^Y & \xrightarrow{\varphi} & \Omega^X \\
\uparrow s \circ \varphi & = & \downarrow \pi_s \circ \Omega^X \\
\Omega^Y & \xrightarrow{\varphi} & \Omega^X \\
\end{array}
\]

**Corollary 4.18.** Let \( T \) be finitary. For each \( f : X \to Y \) in \( \mathcal{K}(T) \), the predicate transformer \( \mathcal{P}(f) : \Omega^Y \to \Omega^X \) is finitary.

**Theorem 4.19** (healthiness, in a finitary setting). Let \( T \) be a monad and \( \Omega = (\Omega, \tau, \Omega, \Omega, \Omega) \) be a \( D \)-relative \( T \)-algebra. Assume \( T \) is finitary, and that \( \pi_X \) is surjective—much like in Theorem 4.10—but for each finite set \( X \). Then, for each map \( \varphi : \Omega^Y \to \Omega^X \), the following are equivalent (healthiness).

- There exists \( f : X \to Y \) in \( \mathcal{K}(T) \) such that \( \mathcal{P}(f) = \varphi \).
- \( \mathcal{P} \) is finitary (Definition 4.17) and lifts to \( D \). The latter means there exists \( \pi : \Omega^Y \to \Omega^X \) such that \( \pi \circ \pi = \varphi \).

In particular, in case a monad \( T \) is finitary, every predicate transformer \( \mathcal{P}(f) : \Omega^Y \to \Omega^X \) that arises from a "computation" \( f : X \to Y \) is finitary in the sense of Definition 4.17. We will see that this is indeed the case for the (sub)distribution monads (Sections 5.2.3.4.4.4). These monads are finitary because we restrict to (sub)distributions with a finite support.

We end with a topological interpretation of Definition 4.17.

**Proposition 4.20.** Let \( \Omega \) be a finite set with the discrete topology. A predicate transformer \( \varphi : \Omega^Y \to \Omega^X \) is finitary if and only if \( \varphi \) is continuous with respect to the product topology of \( \Omega^Y \) and \( \Omega^X \).

### 5. (Purely) Probabilistic Examples

The term probabilistic computation in the literature often refers to one with an alternation of probabilistic and nondeterministic branching, the latter modeling (totally unknown) environments’ behaviors, or a (demonic) scheduler. This will be an example of our extended alternating framework of Section 6. Here we deal with computations with purely probabilistic branching.

#### 5.1 Monads and Modalities for Probabilistic Branching

One of the following monads replaces \( \mathcal{P} \) in Section 2. We impose the restriction of countable supports.

**Definition 5.1** (the (sub)distribution monad \( D_{\leq 1}, D_{\leq 1} \)). The distribution monad \( D_{\leq 1} \) on \( \text{Set} \) is such that: \( D_{\leq 1} X = \{P : X \to [0, 1] | \sum_{x \in X} P(x) = \frac{1}{n}, P(x) = 0 \text{ for all but finitely many } x \in X\} \). The distribution monad \( D_{\leq 1} \) is such that \( \sum_{x \in X} P(x) \leq 1 \).

The (sub)distribution monad \( D_{\leq 1} \) is a variant defined by \( D_{\leq 1} X = \{P \mid \text{finite supp. } \sum_{x \in X} P(x) \leq 1\} \).}

\( D_{\leq 1} \)-algebras are often called convex spaces, with convex subsets in \( \mathbb{R}^n \) as typical examples. \( D_{\leq 1} \)-algebra morphisms are convex linear maps, accordingly. Since any (finite) convex combination can be expressed by a repetition of suitable binary convex combinations \( x \lor y = (1 - p)x + py \), a \( D_{\leq 1} \)-algebra structure is totally determined by how binary convex combinations are interpreted.

**Remark 5.2.** Not all convex spaces are represented as convex subsets of \( \mathbb{R}^n \)-vector spaces: a two-point set \( \{x, y\} \) is a convex space by defining \( (1 - p)x + py \) as \( x \) if \( p = 0 \) and \( y \) otherwise. In general we have a monad map \( D_{\leq 1} \to \mathcal{P}_\omega \) that takes the support of a distribution; consequently each join-semilattice (i.e. \( \mathcal{P}_\omega \)-algebras) yields a convex space. See 8 for more on convex spaces.
A $D_{\leq 1}$-algebra $x: D_{\leq 1}X \to X$ is in turn called a convex cone, with the point $x(0) \in X$ (where 0 is the zero subdistribution) assuming the special role of the apex of a cone. Indeed it is straightforward to see that a convex cone is a “pointed convex space.”

Let us turn to modalities—i.e., $D_{\leq 1}$- and $D_{\leq 1}$-algebras—that would induce predicate transformer semantics by the modality recipe (Theorem 5.3). We adopt $\Omega = [0, 1]$, the unit interval, with the intuition that “probabilistic predicates” are $[0, 1]$-valued random variables whose values express the likelihood of truth.

**Definition 5.3** (modalities for $D_{\leq 1}$). For $D_{\leq 1}$ we use $\tau: D_{\leq 1}[0, 1] \to [0, 1]$; it uses the usual convex structure of $[0, 1]$.

For $D_{\leq 1}$ we have a continuum of modalities: for each real number $r \in [0, 1]$ a modality $\tau_r: D_{\leq 1}[0, 1] \to [0, 1]$ is given by $\tau_r(p) = \sum_{x \in [0, 1]} f(x)p(x)$.

We will in particular use the two extremes $\tau_{total} = \tau_0$ and $\tau_{partial} = \tau_1$; they are called the total and partial modalities for $D_{\leq 1}$, respectively. In the latter divergence—whose probability is expressed by $1 - \sum x^2p(x)$—is deemed to yield truth. Hence $\tau_{total}$ and $\tau_{partial}$ are analogues of $\circ$ and $\Box$ in the nondeterministic setting (Section 2).

### 5.2 Healthiness for: Possibly Diverging Probabilistic Computations and the Total Modality

We shall first focus on the subdistribution monad $D_{\leq 1}$, and the total modality $\tau_{total}$. These data give rise to predicate transformers—ins the form of a state-and-effect triangle (9)—via the modality recipe (Theorem 5.3). In particular we obtain a functor $\mathbb{P}^{\tau_{total}}: K(D_{\leq 1}) \to \mathbb{Set}^{\Omega}$ for interpreting a function $X \to D_{\leq 1}Y$; the latter is identified with a probabilistic computation from $X$ to $Y$ that is possibly diverging (accounted for by sub-probabilities).

Our goal is a healthiness result in this setting, towards which we rely on our relative algebra recipe. As we noted the original Theorem 4.8 is not enough; we use its finitary variant (Theorem 4.19), providing its ingredient (a relative algebra $\mathcal{H}$) by means of the lifting result (Proposition 4.13).

It turns out that the category $D$ in the relative algebra recipe is given by so-called generalized effect modules, they have been used in the context of categorical quantum logics (13) and the more general theory of effectuses (5).

**Definition 5.4** ($\mathbb{GEMod}$). A partial commutative monoid (PCM) is a set $M$ with a partial binary sum $\oplus$ and a zero element $0 \in M$ that are subject to: $(x \oplus 0) \oplus z = x \oplus (0 \oplus z) = x \oplus 0 = z \oplus 0 = x$, where $z$ is the Kleene equality. A generalized effect algebra is a PCM $(M, \oplus, 0)$ that is positive $(x \oplus y = 0 \Rightarrow x = y = 0)$ and cancellative $(x \oplus y = z \Rightarrow x = y = z)$.

A generalized effect module is a generalized effect algebra $M$ with a scalar multiplication $\cdot: [0, 1] \times M \to M$ that satisfies $r \cdot (x \oplus s) = (r \cdot x) \oplus (s \cdot x) = r \cdot (x \oplus y) \oplus (r \cdot y), 1 \cdot x = x$, and $r \cdot (s \cdot x) = (r \cdot s) \cdot x$. For $r, s \in [0, 1]$ the partial sum $r \oplus s = r + s$ is defined when $r + s \leq 1$. The category of general effect modules (with a straightforward notion of their morphism, see (4)) is denoted by $\mathbb{GEMod}$.

An example of a generalized effect module is the set $D_{\leq 1}X$ of subdistributions over $X$. Here $p \oplus q \in D_{\leq 1}X$ is given by $(p \oplus q)(x) = p(x) + q(x)$, which is well-defined clearly only if $\sum_{x \in X} p(x) + q(x) \leq 1$. The set $D_{\leq 1}X$ comes with an obvious scalar multiplication, too. The unit interval $[0, 1]$ is another example; so is its products $[0, 1]^X$. See (4) for details.

For our purpose of healthiness conditions, we have to study the monad map induced by the $D_{\leq 1}$-algebra $\tau_{total}$. It shall also be denoted by $\tau_{total}$. The following is easy.

**Lemma 5.5.** The monad map $\tau_{total}: D_{\leq 1} \to \mathbb{Set}([0, 1], [0, 1])$ is concretely given by $(\tau_{total})x(p)(f) = \sum_{x \in X} f(x)p(x)$.

1. It lifts to $\tau_{total}: D_{\leq 1} \to \mathbb{GEMod}([0, 1], [0, 1])$, that is, the map $(\tau_{total})x(p): [0, 1]^X \to [0, 1], \text{for each } X$ and $p \in D_{\leq 1}X$ preserves $\circ$ and scalar multiplication.

2. Furthermore $(\tau_{total})X: D_{\leq 1}X \to \mathbb{GEMod}([0, 1]^X, [0, 1])$ is an isomorphism for each finite set $X$.

From the last lemma the following healthiness result follows immediately, via our general results. Specifically: Lemma 5.5.1 discharges the condition of Proposition 4.14 and provides the ingredient for the relative algebra recipe; we then exploit Lemma 5.5.2 and that $\tau_{total}$ is finitary in applying Theorem 4.19.

**Theorem 5.6** (healthiness for $D_{\leq 1}$ and $\tau_{total}$). For a function $\varphi: [0, 1]^Y \to [0, 1]^X$ the following are equivalent: 1) there is $f: X \to D_{\leq 1}Y$ such that $\varphi = \mathbb{P}^{\tau_{total}}(f); 2) \varphi$ is finitary (Definition 4.17) and is a morphism of generalized effect modules, meaning that $\circ$ and scalar multiplications are preserved by $\varphi$.

**6. Alternating Branching**

In this last section we further extend our general framework to accommodate alternating branching, in which two players in conflicting interests interplay. Its instances are pervasive in computer science, such as: games, i.e. a two-player variant of automata, in which two players alternate in choosing next states (see e.g. (13)); and various modeling of probabilistic systems where it is common to include additional nondeterministic branching for modeling demonic behaviors of the environments (or schedulers). See e.g. (12).

In (13) the modality recipe (Theorem 5.3) is extended to alternating branching; the central observation is a compositional treatment of two branching layers, using a monad $T$ on $\mathbb{Set}$ (for one) and a monad $R$ on $\mathbb{EM}(T)$ (for the other). See (22) later. It turns out
that the same idea works for our current generalized relative algebra recipe (Theorem 4.3).

After the general framework we will describe some examples. A notable one is (a variant of) probabilistic predicate transformers (31).

6.1 The Relative Algebra Recipe for Alternation

Definition 6.1 ($R \circ T$). Let $T$ be a monad on $Set$ and $R$ be a monad on $EM(T)$. Then a monad $R \circ T$ is defined by the composite of the canonical adjunction $F : \iota : EM(T) \to Set$ and the monad $R$.

\[
R \circ T = URF \subseteq Set \xrightarrow{\iota} EM(T) \xrightarrow{R} \quad (22)
\]

Example 6.2. One example is given by $T = P$ on $Set$ and $R = Up$, the up-closed powerset monad, on $EM(P) \cong CL_Y$. It is given by $Up(L, \leq) = \{ \{ S \subseteq L \mid S \text{ is up-closed } \} \};$ note that the inclusion order is reversed. This combination is for alternating branching where both layers are nondeterministic.

Another is given by $T = D_{\neq 1}$ on $Set$ and $R = Cw$, the nonempty convex powerset monad, on $EM(D_{\neq 1}) \cong Conv$ (the category of convex spaces and convex-linear maps). The latter is given by

\[
CwX = \{ S \subseteq X \mid S \text{ is nonempty and convex-closed, i.e. } x_1, \ldots, x_n \in S \text{ and } \lambda_1 + \cdots + \lambda_n = 1 \text{ implies } \sum \lambda_i x_i \in S \} .
\]

This is for alternating probabilistic and nondeterministic branching. It is important that these $R$ are not monads on $Set$ per se; they involve $T$-algebra structures.

Remark 6.3. We have comparison functors $K : K_!(R \circ T) \to K_!(R)$ and $L : EM(R) \to EM(R \circ T)$ as follows.

\[
K_!(R \circ T) \xrightarrow{K} K_!(R) \xrightarrow{RA} EM(R) \xrightarrow{L} EM(R \circ T) (23)
\]

We aim at reproducing the relative algebra recipe (Theorem 4.3) for the current alternating setting. The first ingredient for the recipe was a dual adjunction $Set \xrightarrow{\eta} Set^{D^{op}}$ from which we derived a continuation-like monad $D(\Omega^T, \Omega^D)$. For the alternating version of the recipe, in view of (28) it is natural to use a dual adjunction $EM(T) \xrightarrow{\hat{\eta}} EM(T)^{D^{op}}$ (where $EM(T)$ replaced $Set$ in the non-alternating counterpart). Interestingly, for such an ingredient $EM(T) \xrightarrow{\hat{\eta}} EM(T)^{D^{op}}$ we can exploit the (original, non-alternating version of) relative algebra recipe itself. See (13) in Theorem 4.8 this yields a continuation-like monad $[[-, \Omega]^T, \Omega]^D$ over $EM(T)$.

Then it is clear that the next key ingredient in the original recipe—namely a monad $\tau : T \to D(\Omega^T, \Omega^D)$—has a monad map $\rho : R \to [[-, \Omega]^T, \Omega]^D$ as its alternating counterpart.

Given such data we obtain predicate transformer semantics.

Definition 6.4 ($P^\rho, P^{(\rho, \tau)}$). Let $T$ be a monad on $Set$, $D$ be a complete category and $\Omega = (\Omega^D, \tau)$ be a $D$-relative $T$-algebra. Moreover let $R$ be a monad on $EM(T)$ and $\rho : R \to [[-, \Omega]^T, \Omega]^D$ be a monad map. Here $[[-, \Omega]^T, \Omega]^D$ is the monad that arises from the dual adjunction $EM(T) \xrightarrow{\hat{\eta}} EM(T)^{D^{op}}$ induced by $\Omega$ as in Theorem 4.3.

We define a functor $P^\rho : K_!(R) \to D^{op}$ by: $P^\rho A = A$ and $P^\rho(A) \xrightarrow{\rho} B$ in $K_!(R)$ \(\iff\) \(\{ [B_0, \Omega]^T \xrightarrow{\rho} [RB_0, \Omega]^T \xrightarrow{\rho} [A_0, \Omega]^T \}\).

Furthermore, by precomposing the comparison functor $K : K_!(R \circ T) \to K_!(R)$, we have another functor:

\[
P^{(\rho, \tau)} = P^\rho \circ K : K_!(R \circ T) \to D^{op} . \quad (24)
\]

The last functor $P^{(\rho, \tau)}$ is what we want: it interprets a function $X \to (R \circ T) Y$—a computation from $X$ to $Y$, with alternation of $T$- and $R$-branching—in the category $D$, in a backward manner.

The following extends Theorem 4.10.

Theorem 6.5 (alternating healthiness condition). Assume the setting of Definition 6.4 and let $X$ and $Y$ be sets. If the map

\[
UPFY : URFY \to U[[FY, \Omega]^T, \Omega]^D \simeq D(\Omega^Y, \Omega^D)
\]

is surjective (injective), then the action $P^{(\rho, \tau)} : K_!(R \circ T)(X, (R \circ T)Y) \to D(\Omega^X, \Omega^Y)$ of $P^{(\rho, \tau)}$ is surjective (injective).

Let us now assume that $D$ is concrete, and develop an alternating counterpart of Section 4.3.

Theorem 6.6. Suppose we have a monad $T$ on $Set$, a complete category $D$ with a faithful and limit-preserving functor $V : D \to Set$, and a $D$-relative $T$-algebra $\Omega = (\Omega^D, \tau)$. Moreover assume we have a monad $R$ on $EM(T)$ and an $R$-algebra structure $\rho : RA_0 \to \Omega_0$. Then the following are equivalent.

1. The monad map $\rho : R \to \Omega$, $EM(T)(\cdot, \cdot)$ that corresponds to $\rho$ (Proposition 4.2) lifts to a monad map $\rho : R \to [[-, \Omega]^T, \Omega]^D$, equipped with a suitable $D$-structure.

2. (Lifting condition) For each $T$-algebra $A_\rho$, the extension map $\rho_{A_\rho} : EM(T)(A_\rho, \Omega) \to EM(T)(RA_\rho, \Omega_\tau)$, which maps a $T$-algebra morphism $f : A_\rho \to \Omega_\tau$ to $\rho \circ RF : RA_\rho \to \Omega_\tau$, lifts (along $V$) to a $D$-morphism $\rho : [A_\rho, \Omega]^T \to [RA_\rho, \Omega]^T$. That is, there exists $\rho^D$ such that $V\rho^D = \rho^\rho$.

3. (Pointwise lifting condition) For each $A_\rho$ and $x \in URA_\rho$, the map $(\rho^\rho)_x = \pi_x \circ \rho : EM(A_\rho, \Omega) \to \Omega$ lifts to a $D$-morphism $(\rho^D)_x : [A_\rho, \Omega]^T \to [RA_\rho, \Omega]^T$. Here $\pi_x$ is defined by the following composite:

\[
\pi_x = \left( EM(T)(RA_\rho, \Omega_\tau) \xrightarrow{U} Set(URA_\rho, \Omega) \xrightarrow{\pi_x} \Omega \right) \quad \text{in Set}
\]

and $\pi_x$ evaluates a function $h : URA_\rho \to \Omega$ by $x$.

In the above we implicitly used the isomorphism $V[A_\rho, \Omega]^T \cong EM(T)(A_\rho, \Omega)$ given in Proposition 4.13.

6.2 Examples

We list healthiness results for some alternating situations. We indicate how we exploit the general framework above; the details are omitted for space reasons.

Nondeterminism and Divergence

In Dijkstra’s original work [31] the first healthiness result is presented for computations with alternation between divergence and nondeterminism. They are described by functions of the type $X \to (P_+ \times L)Y$, where: $LX = 1 + X$ is the lift monad on $Set$ (modeling potential divergence); and $P_+$ is the nonempty powerset monad on $EM(L) \cong EM_{\subset}$, the category of pointed sets. The latter monad is given specifically by $P_+(X, x) = \{ S \subseteq X \mid S \neq \emptyset, \{ x \} \}$.

Suitable modalities $\tau$ and $\rho$ are found to capture the setting of (18). For the category $D$ for predicate transformers we introduce the notion of strict complete meet-semilattice. It is a poset with the least element $0$ and arbitrary but nonempty meets.

Theorem 6.7 (healthiness for nondeterminism and divergence). For a function $\varphi : 2^X \to 2^L$ the following are equivalent: 1) there is an $f : X \to (P_+ \times L)Y$ such that $\varphi \equiv P^{(\rho, \tau)}(f)$; 2) $\varphi$ preserves $0$ and nonempty meets.
Alternating Nondeterminism

Alternation of two layers of nondeterminism is found e.g. in games. In program logic point of view—one player ensures a postcondition, no matter the other player’s move—is such computation is best modeled as a function \( X \rightarrow (\uparrow \uparrow \setminus \setminus \Y) \). Here Up is the monad on \( EM(\mathcal{P}) \cong \mathcal{C} \), from Example 6.2. There are modalities \( \tau \) and \( \rho \) suited to capture the above game-theoretic intuitions; see [13]. Here we choose a combination in which: the opponent moves first, and the protagonist follows. Towards healthiness we take posets and monotone functions in \( D \).

Theorem 6.8 (healthiness for alternating nondeterminism). For a function \( \varphi : 2^X \rightarrow 2^Y \) the following are equivalent: 1) there is a function \( f : X \rightarrow (\uparrow \uparrow \setminus \setminus \Y) \) s.t. \( \varphi = \phi^{(\rho,\tau)}(f) \); 2) \( \varphi \) is monotone. □

Nondeterminism and Probability

Finally we study a common setting in the study of probabilistic systems where: a demonic nondeterministic choice occurs first, followed by an angelic probabilistic choice. This is modeled by a function \( X \rightarrow (\mathcal{C}_v \times \mathcal{D}_\mathcal{C}_v)_Y \), where \( \mathcal{C}_v \) on \( EM(\mathcal{D}_\mathcal{C}_v) \cong \mathcal{C} \), is from Example 6.2. Predicate transformer semantics of such computations has [0,1] as the domain of truth values; and suitable modalities \( \tau \) and \( \rho \) are found much like in [13]. The outcome is a (slight variation of) probabilistic predicate transformers in [31].

For healthiness we use the category of: effect algebras (Definition 5.8) and what we call regular-sublinear maps between them. The latter are subject to: (subadditivity) if \( x \leq y \) then \( f(x) \leq f(y) \) and we have \( f(x) \otimes f(y) \leq f(x \otimes y) \); (scaling) \( f(\lambda x) = \lambda f(x) \); and (translation) \( f(x + \lambda \cdot y) = f(x) + \lambda f(y) \) if \( x \leq \lambda \cdot y \). It deviates from sublinear maps in [31] in that we require \( = \) in (translation).

In the following we assume \( \Y \)’s finiteness: this is like in [23].

Theorem 6.9 (healthiness for nondeterminism and probability). Assume \( \Y \) is finite. For a function \( \varphi : [0,1]^X \rightarrow [0,1]^Y \) the following are equivalent: 1) there is a function \( f : X \rightarrow (\mathcal{C}_v \times \mathcal{D}_\mathcal{C}_v)_Y \) such that \( \varphi = \phi^{(\rho,\tau)}(f) \); 2) \( \varphi \) is regular-sublinear.

Acknowledgments

We thank Toshiki Kataoka for helpful discussions, and the anonymous referees for useful comments. Special thanks are due to John Power for the lectures he gave on the occasion of his visit to Tokyo; the notion of relative algebra is inspired by them. W.H., H.K. and I.H. are supported by Grants-in-Aid No. 24680001, 15K11984 & 15KT0012, JSPS.

References

A. Omitted Proofs and Details

A.1 Explicit Definition of Monad Maps

**Definition A.1.** Let \( S, T \) be monads on \( C \). A monad map from \( S \) to \( T \) is a natural transformation \( \alpha : S \rightarrow T \) that makes the following diagram commute.

\[
\begin{array}{c}
S X \xrightarrow{\eta_X} X \xrightarrow{\eta^X} TX \\
\downarrow{\alpha_X} \quad \downarrow{\mu^X} \quad \downarrow{\mu^X} \\
S TX \xrightarrow{S\alpha_X} S\alpha_TX \xrightarrow{\alpha_TX} TTX
\end{array}
\]  

(25)

Here \( \eta(-) \) and \( \mu(-) \) are the unit and the multiplication of monads.

A.2 Proof of Lemma 2.1

**Proof.** Let the monad \([2^{(-)}, 2]_V\) be denoted by \( T \) in the current proof, for brevity.

We need to check that \( \sigma_X \) is join-preserving for each \( X \). Indeed, for each \( S \subseteq X \), we have \( \bigvee_{x \in S} (\bigvee_{f \in F} f(x)) = \bigvee_{f \in F} (\bigvee_{x \in S} f(x)) \) for any family \( F \subseteq 2^X \). It is easy to check that \( \sigma \) is natural, and that it is compatible with monad units. Compatibility with monad multiplications requires the following diagram to commute.

\[
\begin{array}{c}
\mathcal{P} \mathcal{P} X \xrightarrow{\mathcal{P}\sigma_X} \mathcal{P} T X \xrightarrow{\sigma_TX} TTX \\
\downarrow{\mu_X} \quad \downarrow{\mu_X} \quad \downarrow{\mu_X} \\
\mathcal{P} X \xrightarrow{\sigma_X} TX
\end{array}
\]

Indeed, for each \( S \in \mathcal{P} \mathcal{P} X \) we have

\[
\mu \circ \sigma_TX \circ \mathcal{P}\sigma_X(S) = \mu \circ \sigma_TX(\{ \sigma_X(S) \mid S \in S \}) = \\
\mu \left( \lambda \varphi. \bigvee_{S \in S} \varphi \circ \sigma_X(S) \right) = \\
\lambda \varphi'. \left( \lambda \varphi. \bigvee_{S \in S} \varphi \circ \sigma_X(S) \right) (\lambda \xi. \xi(f')) = \\
\lambda \varphi'. \bigvee_{S \in S} \sigma_X(S)(f') = \\
\lambda \varphi'. \bigvee_{S \in S} f'(x) = \\
\sigma_X(\bigcup S).
\]

Finally we check that \( \sigma_X \) is bijective. Its inverse is given by

\[
(\sigma_X)^{-1}(\xi) = \{ x \in X \mid \xi(\delta_x) = 1 \},
\]

where \( \delta_x : X \rightarrow 2 \) is given by: \( \delta_x(x) = 1 \), and \( \delta_x(y) = 0 \) if \( x \neq y \).

A.3 Proof of Theorem 3.4

**Proof.** For any set \( X \) we have \([KX, \Omega, \ell_T] \equiv \Omega^X\) since \( KX \) is the free \( T \)-algebra over \( X \). It is natural in \( X \in \mathcal{K}(T) \). Indeed, for \( f : X \rightarrow TY \), the diagram

\[
\begin{array}{c}
[TX, \Omega, \ell_T] \xleftarrow{(Kf)^*} [TY, \Omega, \ell_T] \\
\downarrow{\Omega^X} \quad \downarrow{\Omega^Y} \quad \downarrow{\Omega^Y} \\
\Omega^X \xleftarrow{f^*} \Omega^{TY} \xleftarrow{\ell^{-1}} \Omega^Y
\end{array}
\]

commutes by direct calculation.

A.4 Proof of Proposition 4.7

**Proof.** Since the functor \( H \) is product preserving, the canonical map \( \theta : H(A^X) \rightarrow (HA)^X \) in \( D' \), defined by the transpose of

\[
X \xrightarrow{1_X} D(A^X, A) \xrightarrow{H} D'(H(A^X), HA),
\]

(26)
is an isomorphism. Using this isomorphism $\theta$, we define a natural transformation $\psi^A$ by $\psi^A = (\theta^{-1})^* \circ H$, that is, 
\[ D(A^X, A) \xrightarrow{H} D'(\eta^X, HA) \xrightarrow{\epsilon^X} D'((H A)^X, HA). \]
This $\psi$ is seen to be a monad map by some diagram chasing. We define a functor $\mathcal{P}$ by 
\[ \mathcal{P}(A, \alpha) = (HA, \psi \circ \alpha) \quad \text{and} \quad \mathcal{P}((A, \alpha), (B, \beta)) = ((HA, \psi \circ \alpha) \xrightarrow{H} (HB, \psi \circ \beta)). \]
It is a routine to check that $H f$ defined by

\[ (\eta f, \psi f) \]
where $K$ is the comparison functor and $A, \alpha \in \mathcal{A}$. The commutativity of the left diagram in (27) is equivalent to

\[ Ω^X \xrightarrow{\tau} (\eta f, \psi f) \xrightarrow{\epsilon^X} (\psi f, \eta f) \xrightarrow{\alpha^X} (\psi f, \eta f) \xrightarrow{\beta^X} (\psi f, \eta f), \]
which is seen to be a monad map by some diagram chasing. We use the following lemma on elementary topology. It easily follows immediately from (28).

**A.6 Proof of Theorem 4.10**

Proof. It is easy to check the functor $P^*$ coincides with the composite

\[ \mathcal{K}(T) \xrightarrow{\mathcal{K}(\tau)} \mathcal{K}(D^0(T)) \xrightarrow{K} D^0, \]
where $K$ is the comparison functor and $\mathcal{K}(\tau)$ denotes the functor defined by

\[ \mathcal{K}(\tau)(X) = X, \quad \text{and} \quad \mathcal{K}(\tau)(X, Y) = \{ X \xrightarrow{\tau} TY \xrightarrow{\tau} D^0(Y, \Omega_D) \text{ (in } \mathcal{S} \}). \]

Since $K$ is full and faithful, it is enough to show the action $\mathcal{K}(\tau)(X, Y) \to \mathcal{K}(D^0(Y, \Omega_D))(X, Y)$ is surjective (resp. injective). By the definition of $\mathcal{K}(\tau)$, this action is the postcomposition map by $\tau$.

\[ \tau : \mathcal{S}(X, TY) \to \mathcal{S}(X, D^0(Y, \Omega_D)) \]

Here we use the identification $\mathcal{K}(T)(X, Y) = \mathcal{S}(X, TY)$ and $\mathcal{K}(D^0(Y, \Omega_D))(X, Y) = \mathcal{S}(X, D^0(Y, \Omega_D))$. When $\tau$ is injective, the postcomposition $\tau$ is injective by the definition of mono. When $\tau$ is surjective, it is split epi (by the axiom of choice) hence so is $\tau$.

**A.7 Proof of Lemma 4.16**

Proof. Since $T$ is finitary, $t : 1 \to TX$ factor through some finite subset $s : X' \to X$ as $t = Ts \circ t'$. Then we get a desired factorization as follows:

\[ \begin{array}{c}
1 \\
\uparrow t \\
TX \\
\downarrow \tau \\
\Omega X \end{array} \xrightarrow{\tau} \Omega^X \xleftarrow{\tau} \begin{array}{c}
\Omega^X \\
\uparrow \Omega t' \\
\Omega^X \\
\downarrow t' \\
\Omega^X \\
\uparrow \Omega t' \\
\Omega^X. \end{array} \]

**A.8 Proof of Theorem 4.19**

Proof. For a finite discrete space $\Omega$ and an arbitrary set $X$, each clopen set $C \subseteq \Omega^X$ is written as $(s^*)^{-1}(S)$ for some finite subset $s : X' \to X$ and some subset $S$ of $\Omega^{X'}$. We can assume $\Omega^X$ is nonempty. Fix a retraction $r : Y \to Y'$. We have $\varphi = \varphi \circ r^*$. Then $r^*$ also lifts to an $D$-morphism $\varphi$. By the surjectiveness of $\sigma_{\Omega^X}$, there exists some $t' \in TY'$ with $\sigma_{\Omega^X}(t') = f'$. Take $t = Ts(t')$, then we have $\tau X(t) = \varphi$, which concludes the proof.

**A.9 Proof of Theorem 4.20**

We use the following lemma on elementary topology. It easily follows from the compactness of $\Omega^X$.

**Lemma A.2.** For a finite discrete space $\Omega$ and an arbitrary set $X$, each clopen set $C \subseteq \Omega^X$ is written as $(s^*)^{-1}(S)$ for some finite subset $s : X' \to X$ and some subset $S$ of $\Omega^{X'}$.

**Proof of Proposition 4.20** We can assume $X = 1$.

If: Assume $\varphi$ is continuous. For each $z \in \Omega$, the inverse image $\varphi^{-1}(z)$ is clopen, so it can be described as $(z^*)^{-1}(S_z)$ for some finite subset $S_z : Y_z \to Y$ and $Y_z \subseteq \Omega^X$. Then $t : Y' \subseteq \bigcup_{z \in \Omega} Y_z \to Y$ is still finite and $\varphi$ factors through $t^*$. If: Assume $\varphi$ is finitary; we have $s : n \to Y$ and $\varphi : \Omega^0 \to \Omega$ such that $\varphi \circ s^* = \varphi$. The map $s^*$ is obviously continuous, and so is $\varphi$ since its domain $\Omega^0$ (finite) discrete. Therefore their composite $\varphi$ is continuous.