Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 352 (2020) 257–281

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Combining Algebraic Effect Descriptions Using the Tensor of Complete Lattices

Niels Voorneveld[1,2]

Department of Software Science
Tallinn University of Technology
Tallinn, Estonia

Abstract

Algebras can be used to interpret the behaviour of effectful programs. In particular, we use Eilenberg-Moore algebras given over a complete lattices of truth values, which specify answers to queries about programs. The algebras can be used to formulate a quantitative logic of behavioural properties, specifying a congruent notion of program equivalence coinciding with a notion of applicative bisimilarity. Many combinations of effects can be interpreted using these algebras. In this paper, we specify a method of generically combining effects and the algebras used to interpret them. At the core of this method is the tensor of complete lattices, which combines the carrier sets of the algebras. We show that this tensor preserves complete distributivity of complete lattices. Moreover, the universal properties of this tensor can then be used to properly combine the Eilenberg-Moore algebras. We will apply this method to combine the effects of probability, global store, cost, nondeterminism, and error effects. We will then compare this method of combining effects with the more traditional method of combining equational theories using interaction laws.

Keywords: Algebraic effects, Eilenberg-Moore algebra, Tree monad, Complete lattice, Tensor product, Program equivalence, Quantitative logic, Applicative bisimilarity, Probability, Nondeterminism, Global store.

1 Introduction

Effects can alter the behaviour of functional programs in many ways. In order to interpret the behaviour of effectful programs, we choose a set of answers called a *quantitative truth space*, and a set of questions (theoretical tests on programs) given by algebras. In [29], such truth spaces and algebras are used to formulate a logic of quantitative behavioural properties, which induces a notion of program equivalence. This is a generalisation of earlier work based on *modalities* (Boolean algebras) [27].

We study *algebraic effects* in the sense of [24]. Such algebraic effects are given by a signature of effect operations. For each effect we choose a truth space A of answers

[1] This research was supported by the ESF funded Estonian IT Academy research measure (project 2014-2020.4.05.19-0001).

[2] Email: niels.voorneveld@taltech.ee

https://doi.org/10.1016/j.entcs.2020.09.013
1571-0661/© 2020 The Author(s). Published by Elsevier B.V.

given by a *complete lattice*, and one or more algebras. Such algebras can be specified by *local functions* over the effect operations (forming an observation algebra). If such functions preserve non-empty suprema, they induce an ω-continuous *Eilenberg-Moore algebra*. Many examples of algebraic effects can be expressed using such algebras, including probability, global store, nondeterminism, error, and cost.

An ω-continuous Eilenberg-Moore algebra specifies a notion of *program equivalence* on functional languages with effects and general recursion. This program equivalence can be formulated as *applicative bisimilarity* [1,5,6], and as logical equivalence via a quantitative logic [29,30]. Moreover, the algebra induces a *compositional equational theory*, which for the various examples coincide with the usual equational theories for algebraic effects formulated in the literature (see e.g. [23,25,26] for such equational theories).

The main contribution of this paper is the development of a generic method for combining effects described by algebras constructed using local functions over effect operations. This method uses the *tensor product* on complete lattices, featured e.g. in [9,16,28,32]. This operation is a symmetric monoidal product which naturally combines the different notions of truth for the different effects. We show that the tensor product preserves *complete distributivity* of complete lattices. Complete distributivity is useful for characterising the structure of the tensor of complete lattices.

We combine algebras by combining their local functions, following an elegant definition in terms of the universal property of the tensor product. Given that the complete lattices we use are completely distributive, this combination of local functions preserves the supremum preservation property mentioned before. As such, the induced algebras are suitable for specifying notions of program equivalence.

We compare this method of combining effects with the more traditional way of combining equational theories, which is done by specifying interaction laws such a commutativity [14,15]. We see that our method of combining effects does not uniformly coincide with either the sum or the tensor of equational theories. Instead, it "chooses" interaction laws appropriate to each combination of effects.

In Section 2, we start with some preliminaries on algebraic effects and complete lattices. In Section 3, we formulate how to interpret effects using *effect descriptions*, which allow us to construct algebras. In Section 4 we develop a general theory for combining quantitative truth spaces, given by completely distributive lattices, using the tensor product. Then in Section 5, we use universal properties of the tensor product to combine functions on complete lattices. In Section 6, we formulate how to combine effect descriptions and their algebras, and apply this method to study several combinations of effects. Lastly, in Section 7, we compare this method with methods for combining equational theories.

2 Preliminaries

We represent effects using *algebraic effect operations* following [24]. Each operation op has an *arity* $ar(\mathsf{op}) \in \mathbb{N}$, which tells how many arguments the operation has. For example, we can consider the operation of nondeterministic choice nor, which has arity 2 and chooses nondeterministically between two possible continuations given by its two arguments.

For each effect, or combination of effects, we specify an *effect signature* Σ given by a set of effect operations.

Definition 2.1 A Σ-*effect tree* (henceforth *tree*) over a set X is a possibly infinite tree whose nodes are:

(i) leaf nodes labelled $\langle x\rangle$ for $x \in X$,

(ii) leaf nodes labelled $\bot$, describing divergence,

(iii) and internal nodes op for $\mathsf{op} \in \Sigma$, which has $ar(\mathsf{op})$ many children.

We write $\mathsf{op}\langle t_1, \dots, t_{ar(\mathsf{op})}\rangle$ for the tree whose root has node op, and whose children are given by $t_1, \dots, t_{ar(\mathsf{op})}$. If a tree only has finitely many nodes, we call it *finite*. We write $T^\nu_\Sigma(X)$ for the set of trees over X, and $T^\mu_\Sigma(X)$ for the subset of $T^\nu_\Sigma(X)$ containing the finite trees over X.

Given an order on X, we can inductively define an order on $T^\mu_\Sigma(X)$ according to the following rules:

- For any $x, y \in X$, $x \leq_X y \Rightarrow \langle x\rangle \leq_{T^\mu_\Sigma(X)} \langle y\rangle$.
- For any $t \in T^\mu_\Sigma(X)$, $\bot \leq_{T^\mu_\Sigma(X)} t$.
- For any $\mathsf{op} \in \Sigma$, and $l_1, \dots, l_{ar(\mathsf{op})}, r_1, \dots, r_{ar(\mathsf{op})} \in T^\mu_\Sigma(X)$,
  $(\forall 1 \leq i \leq ar(\mathsf{op}).\ l_i \leq_{T^\mu_\Sigma(X)} r_i) \Rightarrow \mathsf{op}\langle l_1, \dots, l_{ar(\mathsf{op})}\rangle \leq_{T^\mu_\Sigma(X)} \mathsf{op}\langle r_1, \dots, r_{ar(\mathsf{op})}\rangle$.

The above definition can be adapted to give a coinductive formulation of an order on $T^\nu_\Sigma(X)$. If X is an ω-cpo (meaning it contains limits of increasing sequences), then $T^\nu_\Sigma(X)$ is an ω-cpo as well. Moreover, each tree is the limit of a sequence of finite trees.

The operations $T^\mu_\Sigma(-)$ and $T^\nu_\Sigma(-)$ give endofunctors in the category of posets. Taking some order preserving map $f : X \to Y$, we can lift it to an order preserving map on trees $T^\mu_\Sigma(f) : T^\mu_\Sigma(X) \to T^\mu_\Sigma(Y)$, where $T^\mu_\Sigma(f)(t)$ is the result of replacing in t each leaf $\langle x\rangle$ with $\langle f(x)\rangle$, leaving the rest of the tree unchanged. We define the function $T^\nu_\Sigma(f) : T^\nu_\Sigma(X) \to T^\nu_\Sigma(Y)$ in the same way.

There is an alternative formulation of these functors. First, we have the partiality functor $(-)_\bot$ on posets which adds a minimal element $\bot$ to its argument. Given an effect signature Σ, we define the signature functor F_Σ as $F_\Sigma(X) = \sum_{\mathsf{op}\in\Sigma} X^{ar(\mathsf{op})}$. Then we define the functors using smallest and largest fixpoint constructions:

$$T^\mu_\Sigma(X) = \mu Y.(X + F_\Sigma Y)_\bot, \qquad\qquad T^\nu_\Sigma(X) = \nu Y.(X + F_\Sigma Y)_\bot\ .$$

Both $T^\mu_\Sigma(-)$ and $T^\nu_\Sigma(-)$ form monads in the category of posets, where the unit η is given by $\eta_X(x) = \langle x\rangle$, and the monad multiplication μ is given by locally replacing each leaf $\eta(t)$ by the subtree t (we "flatten" the input).

2.1 Motivation

In this subsection we discuss how algebras as defined in this paper can be used. A *computation* is program that needs to be evaluated, or reduced, in order to produce a result. In general, we consider a computation to either return a *value* or *diverge*, potentially encountering effects along the way. As such, we interpret the operational semantics of a programming language as giving a function: $| - | : \mathit{Computations} \to T^{\nu}_{\Sigma}(\mathit{Values})$. In the absence of recursion, this function could be given by $| - | : \mathit{Computations} \to T^{\mu}_{\Sigma}(\mathit{Values})$.

When we test a property of a program, we often start with a property on the values this program can return. For instance, a property of natural numbers can be "Evenness". A property on values is in general given by a quantitative predicate $P : \mathit{Values} \to A$. This uses a truth space A, like the Booleans, or the real number interval $[0, 1]$ of probabilities. To translate this predicate on values to a predicate on programs which return such values, we want to lift the predicate P to a quantitative predicate $P' : \mathit{Computations} \to A$. For this purpose, we specify an algebra $\alpha : T^{\nu}_{\Sigma}(A) \to A$ in order to perform the following composition: $(\alpha \circ T^{\nu}_{\Sigma}(P) \circ | - |) : \mathit{Computations} \to A$.

This composition implements the following intuition. We have a program which, after invoking some effects, produces a value according to the operational semantics $| - |$. We have determined a degree of truth for each potential value the program may return, using a quantitative predicate P. Lastly, we use an algebra α, which interprets how effects encountered during the execution of the program combines the degrees of truths of all possible return values, and the effectful ways we could get to those values, into a singular degree of truth.

Take for instance a program which tosses a coin, and on heads will return an 3, and on tails a 6. We want to determine to what degree this program produces an even number. So we take a predicate P which associates to 3 a 0 probability of being even, and to 6 a 1 probability of being even. Then our algebra combines these two results, with the knowledge that the coin is fair. This allows it to determine that the program is expected to produce an even number half of the time.

In the above example, we use as truth space $A = [0, 1]$, the real number interval representing probabilities that certain properties (like "evenness") are held. However, depending on the effect in question, we may need other notions of truth. For instance, for computations using a global store, truth is conditional on the state of this global store upon initiation of evaluation. In that case, we can use the powerset of states as truth space. In the next subsection we study which properties A, the truth space for our quantitative predicates, needs to satisfy to be useful, keeping it general enough to accommodate a plethora of examples.

2.2 Complete lattices

To interpret effectful behaviour, we use a poset of *truth values*, potential answers to questions asked of (or tests performed on) programs. Intuitively, if $a < b$, then b represents a truth which holds in "more" evaluations of a program than the truth of a. The notion of "more" depends on the effect in question. It may mean: more likely, or for more initial states.

In order for this space of truth values to be suitable for specifying a congruent notion of program equivalence, it needs to be a complete lattice. This has two theoretical reasons:

- In order to construct enough quantitative predicates for distinguishing between behaviourally different programs, we need to close our collection of predicates under arbitrary suprema and infima.
- To formulate applicative bisimilarity, a relation lifting device called a *relator* is defined using our algebras. To facilitate this process, suprema are used to prove that such relators preserve composition of relations.

We will not go further into these two points in this paper. See [29,30] for more details.

Given a poset A and a subset $X \subseteq A$, we write $\bigvee X$ for the supremum of X: the unique smallest element of A greater than or equal to any element in X. Let $\bigwedge X$ be the infimum of X: the unique largest element of A less than or equal to any element in X. These elements may not exist. We distinguish between two notions of completeness.

Definition 2.2 A poset A is a *complete lattice* if for any $X \subseteq A$, $\bigvee X$ exists.
A poset A is an *inhabitant complete lattice* (or icomplete lattice)[3] if for any *non-empty* $X \subseteq A$, $\bigvee X$ exists.

If A is a complete lattice, the infimum $\bigwedge X$ always exists, and is equal to $\bigvee\{x \in A \mid \forall y \in X, x \leq y\}$.

Note that a complete lattice is also icomplete. The difference between the two notions of completeness is whether or not the element $\bigvee \emptyset$, the smallest element of A, exists. The top element, $\bigwedge \emptyset = \bigvee A$, always exists in an icomplete lattice. We denote $\mathbf{F}_A$ for the smallest element of A, and $\mathbf{T}_A$ for the biggest element of A.

We expand the intuition of the complete lattice as a truth space. The complete lattices contains degrees of truth, which are possible answers to tests on programs. For instance, we make ask ourselves whether: "this program returns an even number". The element $\mathbf{T}_A$ denotes the fact that the property is always observed, regardless of effects that may occur in the evaluation of the program. This happens for instance if the program terminates with a desirable result, without any interference of effects. The element $\mathbf{F}_A$ denotes the fact the property is never observed. This can for instance happen if the program diverges, or produces an undesirable result, without any interference of effects. Even when effects occur, $\mathbf{T}_A$ and $\mathbf{F}_A$ may be attained. For instance, a randomized algorithm may produce different results but still have a 100% probability of producing an even number.

Examples of complete lattices, used as truth spaces for effects, include:

- The Booleans $\mathbb{B}$ containing only true and false.

[3] This terminology does not appear in the literature. It is used here to distinguish it from (total) completeness. The same holds for the term 'ilinear' later in the paper.

- The real number interval $[0,1]$ describing probabilities that a statement is true.
- The powerset $\mathcal{P}(S)$ of states S describing initial states of a global store.
- The natural numbers $\mathbb{N}^\infty$ with limit element ∞, describing costs.

There is a functor $!(-)$ from the category of icomplete lattices Icom to the category of complete lattices Com, which adds a smallest element to its input. This functor is left adjoint to the forgetful functor $U : \mathsf{Com} \to \mathsf{Icom}$ (which forgets that its input has a smallest element). Note that both $U\circ\,!$ and $!\circ U$ give an endofunctor akin to the partiality functor $(-)_\bot$ on categories Icom and Com respectively.

Importantly, $T^\mu_\Sigma(-)$ and $T^\nu_\Sigma(-)$ are not functors in these categories, since $T^\mu_\Sigma(X)$ and $T^\nu_\Sigma(X)$ do not tend to have top elements. For instance, two different effect operators do not have a common upper bound. This is why we will remain firmly inside the category of posets when discussing algebras.

We write $\Pi_{i\in I}A_i$ for the I-indexed product of sets A_i. An element of this space is given in lambda notation, $\lambda i.a_i$, which represents an I-indexed family where for each $i \in I$, $a_i \in A_i$.

Definition 2.3 A function $f : \Pi_{i\in I}A_i \to B$ from a product of icomplete lattices to an icomplete lattice is *ilinear* if for any family of nonempty subsets $\{S_i \subseteq A_i\}_{i\in I}$, $f(\lambda i. \bigvee S_i) = \bigvee\{f(\lambda i.x_i) \mid \forall i.x_i \in S_i\}$.

A function $f : \Pi_{i\in I}A_i \to B$ from a product of complete lattices to a complete lattice is *linear* if for any family of subsets $\{S_i \subseteq A_i\}_{i\in I}$, $f(\lambda i. \bigvee S_i) = \bigvee\{f(\lambda i.x_i) \mid \forall i \in I.\ x_i \in S_i\}$.

If a function is linear, then it is ilinear. Conversely, an ilinear function $f : \Pi_{i\in I}A_i \to B$ on complete lattices is linear if and only if for any family of elements $\{x_i \in A_i\}_{i\in I}$ such that $\exists i \in I.x_i = \mathbf{F}_{A_i}$, then $f(\lambda i.x_i) = \mathbf{F}_B$.

The reason why we distinguish between linearity and ilinearity is because of a disconnect between the demands on the truth space A, and the demands on how to answer questions using an algebra. Linearity is the inherent property of morphisms in Com which contains our truth spaces, whereas the local functions which will make up our algebras are ilinear, but not linear in general (see the forthcoming examples in Subsection 3.2).

A complete lattice is *completely distributive* if the infima distributes over the suprema operation. See for instance [7] for an overview of properties of complete lattices. There is an equivalent definition to complete distributivity using linearity:

Definition 2.4 A complete lattice A is *completely distributive* if for any set I, the infima operation $\bigwedge_I \Pi_{i\in I}A \to A$, sending $\lambda i.x_i$ to $\bigwedge_{i\in I} x_i$, is linear.

Observe that it is sufficient to require the infima operations $\bigwedge_I$ to be ilinear, since they automatically satisfy the extra property of bottom preservation discussed below Definition 2.3. Assuming that our truth spaces are completely distributive will be necessary later on for establishing some properties. Note however, that it is always possible to freely generate a completely distributive lattice from any complete lattice. So in theory, this restriction to complete distributivity is not restrictive in terms of applications to describing effects.

3 Effect Descriptions

For each effect, or combination of effects, we choose a completely distributive lattice A of truth values, or space of observables. To interpret the behaviour of effects, we specify an algebra $\alpha : T^{\mu}_{\Sigma}(A) \to A$ in the category of posets. This algebra is constructed using the following recipe.

For each $\mathsf{op} \in \Sigma$, we specify a function $\alpha_{\mathsf{op}} : A^{ar(\mathsf{op})} \to A$. We call such functions the *local functions* of α. We combine all these local functions to create a function $\alpha_{\Sigma} : (A + F_{\Sigma}(A))_{\perp} \to A$ with the following definition:

$$\alpha_{\Sigma}(\perp) := \mathbf{F}_A = \bigvee \emptyset, \qquad \alpha_{\Sigma}(\mathit{inleft}(a)) = a,$$
$$\alpha_{\Sigma}(\mathit{inright}(\mathsf{op}(a_1, \ldots, a_{ar(\mathsf{op})}))) = \alpha_{\mathsf{op}}(a_1, \ldots, a_{ar(\mathsf{op})}).$$

This function inductively induces an algebra $\alpha : T^{\mu}_{\Sigma}(A) \to A$ on $T^{\mu}_{\Sigma}(-) = \mu Y.((-) + F_{\Sigma}(Y))_{\perp}$.

If an algebra is constructed using the above recipe, we call it a *locally constructed* algebra.

Definition 3.1 Given a monad (M, η, μ), an algebra $a : MA \to A$ is an *Eilenberg-Moore algebra* (henceforth EM-algebra) if the following two diagrams commute:

$$\begin{tikzcd} A \arrow[r, "\eta_A"] \arrow[dr, "id_A"'] & MA \arrow[d, "a"] \\ & A \end{tikzcd} \qquad \begin{tikzcd} MMA \arrow[r, "Ma"] \arrow[d, "\mu_A"'] & MA \arrow[d, "a"] \\ MA \arrow[r, "a"'] & A \end{tikzcd}$$

By a simple induction on $T^{\mu}_{\Sigma}(A)$, we get the following result.

Lemma 3.2 *Any locally constructed algebra α forms an* Eilenberg-Moore algebra.

As discussed before, we can use the algebra to interpret the behaviour of effectful computations, by lifting predicates on values to predicates on computations. Given an algebra $a : MA \to A$, we can lift a quantitative predicate $P : X \to A$ to a quantitative predicate $a(P) := (a \circ M(P)) : MX \to A$.

Definition 3.3 An algebra $a : MA \to A$ on an icomplete lattice A is *leaf-ilinear* if for any set X, element $t \in T^{\mu}_{\Sigma}(X)$ and function $f : X \to \mathcal{P}_{\neq\emptyset}(A)$ associating to each $x \in X$ a non-empty subset of A,

$$a(\bigvee \circ f)(t) \quad = \quad \bigvee \{a(P)(t) \mid P : X \to A, \forall x \in X. P(x) \in f(x)\} \ .$$

The concept of leaf-ilinearity implements the notion of ilinearity of algebras $\alpha : T^{\mu}_{\Sigma}(A) \to A$ to the extend that it is possible, given that $T^{\mu}_{\Sigma}(A)$ is not necessarily an icomplete lattice. The leaves of $T^{\mu}_{\Sigma}(A)$ are taken from the icomplete lattice A, and as such are closed under non-empty suprema. Leaf-ilinearity asserts that the algebra preserves such suprema within the leaves. One consequence of this property is that, as a predicate lifting device, a leaf-ilinear algebra preserves non-empty suprema.

By induction on the structure of $T^{\mu}_{\Sigma}(A)$, we have the following result.

Lemma 3.4 *If the algebra α is locally constructed by ilinear local functions, then α is leaf-ilinear.*

3.1 Infinitary trees

We use leaf-ilinearity to extend the algebra $\alpha : T^{\mu}_{\Sigma}(A) \to A$ to an algebra $\widehat{\alpha} : T^{\nu}_{\Sigma}(A) \to A$ capable of interpreting infinite trees. Firstly, note that any ilinear function is monotone. As such, we can establish the following result by induction:

Lemma 3.5 *Suppose a is a leaf-ilinear EM-algebra such that $\alpha(\perp) = \boldsymbol{F}_A$. For any two trees $t, t' \in T^{\mu}_{\Sigma}(A)$ such that $t \leq_{T^{\mu}_{\Sigma}(A)} t'$, $\alpha(t) \leq \alpha(t')$.*

Consider a tree $t \in T^{\nu}_{\Sigma}(A)$, then there is an ascending sequence of trees $t_0 \leq t_1 \leq t_2 \leq \ldots$ in $T^{\mu}_{\Sigma}(A)$ such that $t = \bigvee_n t_n = \bigvee\{t_n \mid n \in \mathbb{N}\}$. We say that the sequence $t_0, t_1, \ldots$ *approximates* t. We define $\widehat{\alpha}(t)$ as $\bigvee_n \alpha(t_n)$. This is well-defined, since for any two sequences $t_0, t_1, \ldots$ and $t'_0, t'_1, \ldots$ approximating t, there is an m for any n such that $t_n \leq t'_m$ and hence by Lemma 3.5, $\alpha(t_n) \leq \alpha(t'_m)$. Hence $\bigvee_n \alpha(t_n) \leq \bigvee_n \alpha(t'_n)$ and vice versa, $\bigvee_n \alpha(t_n) \geq \bigvee_n \alpha(t'_n)$.

Last but not least, we have the following result, for which we give a brief sketch of a proof.

Lemma 3.6 *Suppose α is a locally constructed leaf-ilinear algebra. Then $\widehat{\alpha} : T^{\nu}_{\Sigma}(A) \to A$ is an EM-algebra.*

Proof. Any ilinear function $f : A^n \to A$ is ω-continuous: Given a series of ascending sequences $\{a^1_i\}_{i\in\mathbb{N}}, \{a^2_i\}_{i\in\mathbb{N}}, \ldots, \{a^n_i\}_{i\in\mathbb{N}}$ in A, then, by ilinearity and monotonicity: $f(\bigvee_i a^1_i, \ldots, \bigvee_i a^n_i) = \bigvee\{f(a_{i_1}, \ldots, a_{i_n}) \mid i_1, \ldots, i_n \in \mathbb{N}\} = \bigvee_i f(a^1_i, \ldots, a^n_i)$. Using this, we can by induction establish that, for any ascending sequence of trees $t_0 \leq_{T^{\mu}_{\Sigma}(A)} t_1 \leq_{T^{\mu}_{\Sigma}(A)} \ldots$, $\widehat{\alpha}(\bigvee_{i\in\mathbb{N}} t_i) = \bigvee_{i\in\mathbb{N}} \widehat{\alpha}(t_i)$.

Since any tree $\eta(a)$ is finite, the η rule is simple to prove. For the μ-rule, take $d \in T^{\nu}_{\Sigma}(T^{\nu}_{\Sigma}(A))$, and take some sequence of elements $d_0, d_1, \ldots$ of $T^{\mu}_{\Sigma}(T^{\mu}_{\Sigma}(A))$ approximating d. Then $\{T^{\nu}_{\Sigma}(\widehat{\alpha})(d_i)\}_{i\in\mathbb{N}}$ is an increasing sequence of elements of $T^{\nu}_{\Sigma}(A)$. Hence: $\widehat{\alpha}(T^{\nu}_{\Sigma}(\widehat{\alpha})(d)) = \widehat{\alpha}(T^{\nu}_{\Sigma}(\widehat{\alpha})(\bigvee_i d_i)) = \widehat{\alpha}(\bigvee_i T^{\mu}_{\Sigma}(\alpha)(d_i)) = \bigvee_i \alpha(T^{\mu}_{\Sigma}(\alpha)(d_i)) = \bigvee_i \alpha(\mu d_i) = \widehat{\alpha}(\bigvee_i \mu d_i) = \widehat{\alpha}(\mu(\bigvee_i d_i)) = \widehat{\alpha}(\mu d)$. □

We may sometimes simply write α instead of $\widehat{\alpha}$, as the latter is an extension of the former.

Another way trees could be infinite, is by having effect operations with infinite width. It is not difficult to extend the above theory to include effectful operations with countable arities. To deal with this, we specify an ilinear local function $f : A^{\omega} \to A$. To inductively define an algebra, we use $f' : A^{\omega} \to A$ constructed as:

$$f'(\lambda n.a_n) = \bigvee_{m\in\mathbb{N}} f\left(\lambda n. \begin{cases} a_n & \text{if } n < m \\ \mathbf{F} & \text{if } n \geq m \end{cases}\right) . \tag{1}$$

If f is ilinear, then f' is ω-continuous, hence the induced algebras on $T^{\mu}_{\Sigma}(A)$ and $T^{\nu}_{\Sigma}(A)$ are both ω-continuous EM-algebras.

Lemma 3.7 *If $f : A^{\omega} \to A$ is ilinear, then $f' : A^{\omega} \to A$ as defined in (1) is ω-continuous.*

Proof. For $m \in \mathbb{N}$, let $f_m : A^m \to A$ be the ilinear function $f_m(a_0, a_1, \ldots, a_{m-1}) := f(a_0, a_1, \ldots, a_{m-1}, \mathbf{F}_A, \mathbf{F}_A, \ldots)$. Since f'_m has finite arity and is ilinear, it is ω-continuous (see proof of Lemma 3.6). So, the function $f'_m : A^{\omega} \to A$ defined by $f'_m(\lambda n.a_n) = f_m(a_0, a_1, \ldots, a_{m-1})$ is ω-continuous too. We conclude that $f' = \bigvee_m f'_m$ is ω-continuous. $\square$

Moreover, if f already is ω-continuous, then $f' = f$. The notion of *finiteness* can be adapted to infinite-width trees by saying that a tree is finite if there are only finitely many non-$\perp$ leaves. Using f' in our inductive definition of an algebra effectively formalises the idea that any result of the algebra can be approximated by finite trees.

In the rest of this paper, we will concern ourselves mostly with ilinear local functions over effect operations with finite arity. We specify the following structure for interpreting effects.

Definition 3.8 An *effect description* (Σ, A, α) consists of an effect signature Σ, a completely distributive lattice A, and an *interpretation* α given by an ilinear function $\alpha_{\mathsf{op}} : A^{ar(\mathsf{op})} \to A$ for each $\mathsf{op} \in \Sigma$.

3.2 Examples

We look at some examples of effects and their effect descriptions.

Example 3.9 [Probability] We consider the effect signature $\Sigma_{prob} = \{\mathsf{por}\}$ with a single effect operation por for binary probabilistic choice. The operation has arity 2, and chooses fairly between two continuations. We give this the effect description $(\Sigma_{prob}, [0,1], \mathsf{Exp})$, with as complete lattice of truth values the real number interval $[0,1]$. We give probabilistic computations the *expectation* interpretation Exp, given by the local function $\mathsf{Exp}_{\mathsf{por}} : [0,1]^2 \to [0,1]$ which calculates the average between its two arguments $\mathsf{Exp}_{\mathsf{por}}(a,b) = (a+b)/2$. The constructed algebra $\mathsf{Exp} : T^{\nu}_{\Sigma}([0,1]) \to [0,1]$ will calculate the expected value of a leaf $a \in [0,1]$, assuming each choice in a tree $t \in T^{\nu}_{\Sigma}([0,1])$ is resolved fairly.

Example 3.10 [Global Store] We consider a set of global store locations Loc for storing Boolean values. We consider the effect signature $\Sigma_{global} := \{\mathsf{lookup}_l, \mathsf{update}_l(\mathbf{T}), \mathsf{update}_l(\mathbf{F}) \mid l \in \mathsf{Loc}\}$ which for each global store location l has a lookup operation lookup_l of arity 2 and two update operations $\mathsf{update}_l(\mathbf{T})$, $\mathsf{update}_l(\mathbf{F})$ of arity 1. The update operations save a Boolean value to a global store, whereas the lookup operations look up a Boolean value from a global store location, and uses it to choose one of its two continuations: The left continuation if the value is $\mathbf{T}$ and the right continuation if the value is $\mathbf{F}$. We write $\mathsf{S} := \mathbb{B}^{\mathsf{Loc}}$ for the set of *global states*, and we call elements of $\mathcal{P}(\mathsf{S}) \simeq (\mathsf{S} \to \mathbb{B})$ *assertions* on the global state.

We give this effect the description $(\Sigma_{global}, \mathcal{P}(\mathsf{S}), \mathsf{Wp})$ with the *weakest precondition* interpretation Wp, where:

$\mathsf{Wp}_{\mathsf{lookup}_l}(a, b) := \{s \in a \mid s(l) = \mathbf{T}\} \cup \{s \in b \mid s(l) = \mathbf{F}\}$.

$\mathsf{Wp}_{\mathsf{update}_l(v)}(a) := \{s[l := v] \mid s \in a\}$, where $s[l := v](l) = v$ and $s[l := v](l') = s(l')$ for any $l' \neq l$.

$\mathsf{Wp} : T^{\nu}_{\Sigma}(\mathcal{P}(\mathsf{S})) \to \mathcal{P}(\mathsf{S})$ will calculate the set of correct starting states for which, when the computation is evaluated with that state, it terminates in some leaf $\langle R \rangle$ with a final state satisfying assertion $R \in \mathcal{P}(\mathsf{S})$.

Alternatively, we could also have a global store used for storing natural numbers. In such a case, the lookup operation will have a countably infinite arity, and the set of states is given by $\mathsf{S} = \mathbb{N}^{\mathsf{Loc}}$. This works out the same way as Boolean store. In the rest of the paper we will, for the sake of clarity, only look at Boolean store.

Example 3.11 [Cost] We consider the effect signature $\Sigma_{cost} := \{\mathsf{cost}_q \mid q \in \mathbb{Q}_{>0}\}$, where for each positive rational number q we have an effect operation cost_q with arity 1, which requires a cost q to be spend before continuing evaluation. For example, a sleep operation which delays computation for some time, or a save operation which requires a certain amount of memory to be reserved. We see the nonnegative reals $[0, \infty]$ with reverse order as the space of *total costs* (which contain all limits of rational costs). We give the effect the description $(\Sigma_{cost}, [0, \infty], \mathsf{Tal})$ with the *tally* interpretation Tal summing all costs together, where

$\mathsf{Tal}_{\mathsf{cost}_q}(a) = a + q$.

Example 3.12 [Nondeterminism] We consider the effect signature $\Sigma_{non} = \{\mathsf{nor}\}$ with one effect operation nor with arity 2 for binary nondeterministic choice. We give this the effect description $(\Sigma_{non}, \mathbb{A}, \mathsf{Pos})$, with as truth space the three element chain $\mathbb{A} := \{\mathbf{F}, \Diamond, \mathbf{T}\}$ containing the three degrees of possibility (this is e.g. used in [13] for describing nondeterminism). The smallest element $\mathbf{F}$ represents impossibility, the middle element $\Diamond$ represents possibility, and the largest element $\mathbf{T}$ represents inevitability. We give the effect the interpretation Pos which issues the degree of possibility and follows the algebraic structure established in [4,3]:

$\mathsf{Pos}_{\mathsf{nor}}(a, a) = a, \qquad \mathsf{Pos}_{\mathsf{nor}}(a, b) = \Diamond \quad \text{if } a \neq b.$

The constructed algebra $\mathsf{Pos} : T^{\nu}_{\Sigma}(\mathbb{A}) \to \mathbb{A}$ will produce $\mathbf{T}$ if any resolution of choice leads to a leaf labelled $\mathbf{T}$. It will produce $\Diamond$ if it is possible to get to a leaf labelled $\mathbf{T}$ or $\Diamond$, but not all resolutions yield $\mathbf{T}$.

Example 3.13 [Error] We consider the effect signature $\Sigma_{error} := \{\mathsf{raise}\}$ with a single effect operation raise of arity 0, which aborts evaluation displaying an error message. We use the effect description $(\Sigma_{error}, \mathbb{A}, \mathsf{Err})$ where $\mathbb{A}$ is as given in the previous example. We give this the interpretation Err where $\mathsf{Err}_{\mathsf{raise}_e}() = \Diamond$. This effect may be combined with itself using the forthcoming method for combining effects, in order to get an interpretation of multiple errors. For simplicity, we consider only one error at this time.

Example 3.14 [Pure computation] Last but not least, we consider the situation in which there is no effectful behaviour at all (except for divergence). This has the effect description $(\emptyset, \mathbb{B}, \downarrow)$ with an empty signature, the Booleans as truth space, and the *termination* interpretation, which has no local functions since the signature is empty. The constructed algebra is $\downarrow: T^{\nu}_{\Sigma}(\mathbb{B}) \to \mathbb{B}$, where $T^{\nu}_{\Sigma}(\mathbb{B}) = (\mathbb{B})_{\perp}$, sends $\perp$ to $\mathbf{F}$ and $v \in \mathbb{B}$ to v.

We end this section with a short motivational discussion on how these algebras can give rise to notions of program equivalence.

As noted before, an algebra $\alpha : T^{\nu}_{\Sigma}(A) \to A$ can be used to lift a predicate on values to a predicate on computations. This can be used to generate a logic of quantitative formulas, as done in [29]: For each type in the language, we have a collection of formulas. A program of a type functions as a model for such formulas, satisfying each formula to a certain degree $a \in A$. We say that two programs of the same type are *behaviourally equivalent* if they satisfy each formula to the same degree.

It is also possible to define a notion of *applicative bisimilarity*, in the sense of [1,6]. We define a *relator* using our algebra, which lifts a relation on values to a relation on computations. This relator specifies a notion of *applicative bisimulation*, a relation which is closed under certain operations (like application). Two programs are *applicatively bisimular* if there is an applicative bisimulation which relates the two. Given that our algebra is an ω-continues EM-algebra, this notion of applicative bisimilarity coincides with the notion of behavioural equivalence (see [29]).

Last but not least, we can define the notion of *contextual equivalence*. We specify a *basic relation* on computations of base type: Two programs of base type are equivalent if they satisfy each predicate, lifted by the algebra, to the same degree. The contextual equivalence is then the largest *compatible* (or *congruent*) relation which, on base types, coincide with the basic relation. The preorder A need not be a complete lattice for the definition of contextual equivalence to work, and this notion of program equivalence does not always coincide with the above two notions (in particular in the presence of nondeterminism). However, compared to the other notions of program equivalence, it is in general more difficult to show that two programs are contextually equivalent given the above formulation.

4 Tensor of Complete Lattices

We have defined effect descriptions to interpret the behaviour of algebraic effects. We will now start building the foundation for combining effects and their descriptions. In particular, given two effect descriptions $(\Sigma_1, A_1, \alpha_1)$ and $(\Sigma_2, A_2, \alpha_2)$, we want to find an effect description $(\Sigma_{12}, A_{12}, \alpha_{12})$ for the combination of effects.

Firstly, the combined signature Σ_{12} is given by the sum or disjoint union of the original two signatures $\Sigma_{12} := \Sigma_1 + \Sigma_2$. Combining truth spaces and local functions is more involved. In this section, we study the theory for combining these things, starting with the tensor operation on complete lattices. This tensor, and its two representations, are featured in [9,16,28,32].

Definition 4.1 The *tensor product* of two complete lattices A and B is a complete lattice $A \otimes B$ such that there is a universal linear function $u_{A,B} : A \times B \to A \otimes B$ with the property that: Any linear function $f : A \times B \to C$ into a complete lattice C is the composition of $u_{A,B}$ with some linear function $f^{\otimes} : (A \otimes B) \to C$.

In particular, this factorisation gives us a natural bijection between linear functions $f : (A) \times (B) \to C$ with two arguments and linear functions $g : (A \otimes B) \to C$ with one argument. This can be generalised to a bijection between linear functions $f : \Pi_{i\in I}(A_i \times B_i) \to C$ and linear functions $g : \Pi_{i\in I}(A_i \otimes B_i) \to C$.

We look at different representations of the tensor $A \otimes B$ of two complete lattices (featured in the aforementioned literature), respectively the *powerset representation* and the *function representation*:

(i) $(A \otimes B)^{\mathcal{P}} := \{S \subseteq A \times B \mid \forall x \subseteq A, y \subseteq B,\ x \times y \subseteq S \Leftrightarrow (\bigvee x, \bigvee y) \in S\}$, with inclusion order.
For any $a \in A$, $b \in B$, $u^{\mathcal{P}}_{A,B}(a,b) := \{(a',b') \in A \times B \mid a' = \mathbf{F}_A \vee b' = \mathbf{F}_B \vee (a' \leq a \ \wedge\ b' \leq b)\}$.
A linear function $f : A \times B \to C$ factors through $g : (A \otimes B)^{\mathcal{P}} \to C$ given by $g(S) := \bigvee\{f(a,b) \mid (a,b) \in S\}$.

(ii) $(A \otimes B)^{\rightarrow} := \{f : A \to B \mid \forall x \subseteq A.\ f(\bigvee x) = \bigwedge\{f(a) \mid a \in x\}\}$, with pointwise (extensional) order.
A linear function $f : A \times B \to C$ factors through $g : (A \otimes B)^{\rightarrow} \to C$ given by $g(h) := \bigvee\{f(a, h(a)) \mid a \in A\}$.

Depending on which two complete lattices we combine, we may choose an appropriate representation of the tensor. In the general theory, we will mainly stick to the powerset representation. Since $(A \otimes B)^{\mathcal{P}}$ and $(A \otimes B)^{\rightarrow}$ both represent the same complete lattice, there is an isomorphism between the two, given by:

- $R : (A \otimes B)^{\mathcal{P}} \to (A \otimes B)^{\rightarrow}$, $R(S) = \lambda a. \bigvee\{b \in B \mid (a,b) \in S\}$.
- $L : (A \otimes B)^{\rightarrow} \to (A \otimes B)^{\mathcal{P}}$, $L(h) = \{(a,b) \in A \times B \mid b \leq h(a)\}$.

We give a concrete definition to the aforementioned bijection between linear functions $f : \Pi_{i\in I}(A_i \times B_i) \to C$ and linear functions $g : \Pi_{i\in I}(A_i \otimes B_i) \to C$ with respect to the powerset representation of the tensor product.

$$\begin{aligned} &F : (\Pi_{i\in I}(A_i \times B_i) \to C) \to (\Pi_{i\in I}(A_i \otimes B_i)^{\mathcal{P}} \to C), \\ \text{by}\quad &F(f)(\lambda i.S_i) := \bigvee\{f(\lambda i.(a_i,b_i)) \mid \forall i \in I.\ (a_i,b_i) \in S_i\}. \end{aligned} \tag{2}$$

We look at some known properties of the tensor product, using the two representations interchangeably.

Lemma 4.2 *The tensor product $\mathbb{B} \otimes A$, of the Booleans and a complete lattice A, is isomorphic to A.*

Proof. We use the function representation. Elements of $(\mathbb{B} \otimes A)^{\rightarrow}$ are given by supremum reversing functions $f : \mathbb{B} \to A$. These are precisely the functions $f : \mathbb{B} \to A$ such that $f(\mathbf{F}) = \mathbf{T}_A$, hence they are in one-to-one correspondence with elements of A (values given by $f(\mathbf{T})$). □

Proposition 4.3 *The tensor product gives a symmetric monoidal product in the category of complete lattices* Com*, with the unit given by the Booleans* $\mathbb{B}$*.*

Proof. First note that the powerset representation immediately gives us symmetry. Using the function representation, it can be shown that $(A \otimes (B \otimes C)^{\to})^{\to}$ is isomorphic to $(B \otimes (A \otimes C)^{\to})^{\to}$. Using these isomorphisms:
$(A \otimes (B \otimes C)) \simeq (A \otimes (C \otimes B)) \simeq (A \otimes (C \otimes B)^{\to})^{\to} \simeq (C \otimes (A \otimes B)^{\to})^{\to} \simeq (C \otimes (A \otimes B)) \simeq ((A \otimes B) \otimes C)$.
Lastly, Lemma 4.2 shows the Booleans are a unit for the tensor product. □

We look at one more example of a tensor product relevant to combining effects (Example 1.2.9 from [8]).

Lemma 4.4 *The tensor product* $(\mathcal{P}(S) \otimes A)$*, of the powerset lattice* $\mathcal{P}(S)$ *and a complete lattice* A*, is isomorphic to the complete lattice* $(S \to A)$*, of functions from* S *to* A*, with pointwise order.*

Proof. We use the function representation. Take $f \in (\mathcal{P}(S) \otimes A)^{\to}$, which is a supremum reversing function $f : \mathcal{P}(S) \to A$. Then for any set $K \subseteq S$, $f(K) = f(\bigcup_{s\in K}\{s\}) = \bigwedge_{s\in K} f(\{s\})$. Hence f is completely determined by the function $f' : S \to A$ given by $\lambda s.f(\{s\})$. Vice versa, each function $g : S \to A$ determines an f sending K to $\bigwedge_{s\in K} g(s)$. Hence $(\mathcal{P}(S) \otimes A)^{\to}$ is isomorphic to the function space $S \to A$. □

4.1 The supremum operation for tensor products

We have a closer look at the powerset representation of the tensor of two complete lattices. It is easy to establish that the infimum operation on $(A \otimes B)^{\mathcal{P}}$ is given by the intersection on sets. The supremum operation on $(A \otimes B)^{\mathcal{P}}$ is more complicated. Luckily, if the tensor is taken over completely distributive lattices, this supremum can be given a clear and useful characterisation.

Firstly, we investigate the following closure operation.

Definition 4.5 Given two complete lattices A and B, and a subset $S \subseteq A \times B$, let $\widehat{S} \subseteq A \times B$ be the set $\{(\bigvee x, \bigvee y) \mid x \subseteq A, y \subseteq B, x \times y \subseteq S\}$.

Note that if $S \in (A \otimes B)$, then $S = \widehat{S}$. Moreover, for $S \subseteq S' \subseteq A \times B$, $\widehat{S} \subseteq \widehat{S'}$, hence $\widehat{S}$ is always included in the smallest element of $(A \otimes B)$ containing S. With the following lemma, we can prove that under certain conditions, $\widehat{S}$ is the smallest element of $(A \otimes B)$ including S.

We call a subset $S \subseteq A \times B$ *down-closed* if for any $(a, b) \in S$, $a' \leq a$, and $b' \leq b$, we have $(a', b') \in S$.

Lemma 4.6 *Suppose that both* A *and* B *are completely distributive lattices, then for any down-closed* $S \subseteq A \times B$*,* $\widehat{S} \in (A \otimes B)$*.*

Proof. Suppose $(\bigvee x, \bigvee y) \in \widehat{S}$, then there are $x' \subseteq A$ and $y' \subseteq B$ such that $(x' \times y') \subseteq S$, $\bigvee x' = \bigvee x$ and $\bigvee y' = \bigvee y$. Hence for any $a \in x$ and $b \in y$, by down-closure of S, $(\{a \wedge a' \mid a' \in x'\} \times \{b \wedge b' \mid b' \in y'\}) \subseteq S$, hence $(\bigvee\{a \wedge a' \mid a' \in x'\}, \bigvee\{b \wedge$

$b' \mid b' \in y'\}) \in \widehat{S}$. Now, by distributivity, $\bigvee\{a \wedge a' \mid a' \in x'\} = a \wedge \bigvee x' = a \wedge \bigvee x = a$ and similarly $\bigvee\{b \wedge b' \mid b' \in y'\} = b$. Hence $(a, b) \in \widehat{S}$.

Suppose $(x \times y) \subseteq \widehat{S}$, then for any $a \in x$ and $b \in y$, there are $x_a^b \subseteq A$ and $y_a^b \subseteq B$ such that $(x_a^b \times y_a^b) \subseteq S$, $\bigvee x_a^b = a$, and $\bigvee y_a^b = b$. For a family of sets of elements $\{z_i\}_{i \in I}$ of a complete lattice, we denote $\overline{\wedge}_{i \in I} z_i$ for the set $\{\bigwedge_{i \in I} c_i \mid \forall i \in I.\ c_i \in z_i\}$. Since S is down-closed, it holds that for any $a \in x$, $b' \in y$, $((\overline{\wedge}_{b \in y} x_a^b) \times y_a^{b'}) \subseteq S$, hence for any $a \in x$, $((\overline{\wedge}_{b \in y} x_a^b) \times (\bigcup_{b \in y} y_a^b)) \subseteq S$ (where $\bigcup$ is union). Repeating this process, we can derive that $((\bigcup_{a \in x}(\overline{\wedge}_{b \in y} x_a^b)) \times (\overline{\wedge}_{a \in x}(\bigcup_{b \in y} y_a^b))) \subseteq S$, hence $(\bigvee(\bigcup_{a \in x}(\overline{\wedge}_{b \in y} x_a^b)), \bigvee(\overline{\wedge}_{a \in x}(\bigcup_{b \in y} y_a^b))) \in \widehat{S}$. Now, by distributivity,

$$\bigvee(\textstyle\bigcup_{a \in x}(\overline{\wedge}_{b \in y} x_a^b)) = \bigvee_{a \in x}(\bigvee(\overline{\wedge}_{b \in y} x_a^b)) = \bigvee_{a \in x}(\bigwedge_{b \in y}(\bigvee x_a^b)) = \bigvee_{a \in x}(\bigwedge_{b \in y} a) = \bigvee_{a \in x} a = \bigvee x,$$

$$\bigvee(\overline{\wedge}_{a \in x}(\textstyle\bigcup_{b \in y} y_a^b)) = \bigwedge_{a \in x}(\bigvee(\bigcup_{b \in y} y_a^b)) = \bigwedge_{a \in x}(\bigvee_{b \in y}(\bigvee y_a^b)) = \bigwedge_{a \in x}(\bigvee_{b \in y} b) = \bigwedge_{a \in x}(\bigvee y) = \bigvee y.$$

We conclude that $(\bigvee x, \bigvee y) \in \widehat{S}$. □

Corollary 4.7 *For A and B two completely distributive lattices, then for any $X \subseteq (A \otimes B)$, $\bigvee X = \widehat{\bigcup X}$.*

Proof. Since all members of X, as elements of $(A \otimes B)$, are down-closed, we know $\bigcup X$ is down-closed as well. Hence by the previous lemma, $\widehat{\bigcup X} \in (A \otimes B)$. So, considering $\bigvee X$ must be the smallest element of $(A \otimes B)$ containing $\bigcup X$, it must be equal to $\widehat{\bigcup X}$. □

An example: $u_{A,B}(a, b) \vee u_{A,B}(a', b') = u_{A,B}(a, b) \cup u_{A,B}(a', b') \cup u_{A,B}(a \vee a', b \wedge b') \cup u_{A,B}(a \wedge a', b \vee b')$.

5 Combining functions

The tensor product gives us a clear way of combining two linear functions $f : \Pi_{i \in I} A \to A$ and $g : \Pi_{i \in I} B \to B$ into a single linear function $(f \otimes g) : \Pi_{i \in I}(A \otimes B) \to (A \otimes B)$. This is done in the following way:

(i) Compose f and g with $u_{A,B}$ into a single linear function $u_{A,B} \circ (f \times g) : \Pi_{i \in I} A \times \Pi_{i \in I} B \to (A \otimes B)$.

(ii) Permute the arguments of the function to get a linear function from $\Pi_{i \in I}(A \times B)$ to $(A \otimes B)$.

(iii) Apply F from (2) to the result to get a linear function $(f \otimes g) : \Pi_{i \in I}(A \otimes B) \to (A \otimes B)$.

To get a clearer picture of what is going on, we can apply this construction directly to the powerset representation $(A \otimes B)^{\mathcal{P}}$ of the tensor product. This gives us a concrete definition of the combination of functions $(f \otimes g)^{\mathcal{P}} : \Pi_{i \in I}(A \otimes B)^{\mathcal{P}} \to (A \otimes B)^{\mathcal{P}}$, which is as follows:

$$(f \otimes g)^{\mathcal{P}}(\lambda i.S_i) \quad := \quad \bigvee\{u_{A,B}(f(\lambda i.a_i), g(\lambda i.b_i)) \mid \forall i \in I.\ (a_i, b_i) \in S_i\}.$$

Using the powerset representation, we can make the following observation.

Proposition 5.1 *If both A and B are completely distributive lattices, then $A \otimes B$ is completely distributive.*

Proof. We use the alternative definition for complete distributivity given in Definition 2.4.

Take some set I, and consider the infimum functions $\bigwedge_I^A$ and $\bigwedge_I^B$ on A and B respectively. Since A and B are completely distributive, both infimum functions are linear. Hence their combination $(\bigwedge_I^A \otimes \bigwedge_I^B)$ is linear too. Consider some family $\{S_i\}_{i \in I}$ of elements of $(A \otimes B)$.

$$\begin{aligned}(\textstyle\bigwedge_I^A \otimes \bigwedge_I^B)^{\mathcal{P}}(\lambda i.S_i) &= \textstyle\bigvee\{u_{A,B} \circ (\bigwedge_I^A, \bigwedge_I^B)(\lambda i.a_i, \lambda i.b_i) \mid \forall i \in I.\ (a_i, b_i) \in S_i\} \\ &= \textstyle\bigvee\{u_{A,B}(\bigwedge_{i\in I} a_i, \bigwedge_{i \in I} b_i) \mid \forall i \in I.\ (a_i, b_i) \in S_i\} \\ &= \textstyle\bigvee\{u_{A,B}(a,b) \mid (a,b) \in \bigcap_{i \in I} S_i\} \\ &= \textstyle\bigvee\{u_{A,B}(a,b) \mid (a,b) \in \bigwedge_{i \in I} S_i\} = \bigwedge_{i\in I} S_i\,.\end{aligned}$$

This used the facts that, for $X \subseteq (A \otimes B)$, $\bigcap X = \bigwedge X$, and for $S \in (A \otimes B)$, $\bigvee\{u_{A,B}(a,b) \mid (a,b) \in S\} = S$.

Hence, $(\bigwedge_I^A \otimes \bigwedge_I^B)$ is the infimum function on $(A \otimes B)$, which we know is linear. This holds for all indexing sets I, so we conclude by Definition 2.4 that $(A \otimes B)$ is completely distributive. □

5.1 Combining ilinear functions

As can be seen in the examples of Subsection 3.2, local functions are not linear in general, they are ilinear. For example, $\mathsf{Exp}_{\mathsf{por}}(\mathbf{F}_{[0,1]}, \mathbf{T}_{[0,1]}) = (0+1)/2 \neq 0 = \mathbf{F}_{[0,1]}$. So we need a method of combining ilinear functions. Luckily, if all the complete lattices concerned are completely distributive, the bijection F given in (2) preserves ilinearity. This allows us to use the same construction to combine ilinear functions.

Lemma 5.2 *Suppose that A and B are completely distributive lattices, then for any ilinear function $f : \Pi_{i\in I}(A \times B) \to C$, $F(f)$, the function $F(f) : \Pi_{i \in I}(A \otimes B) \to C$ is ilinear.*

Proof. Take for any $i \in I$ an arbitrary set $X_i \subseteq (A \otimes B)$. By Corollary 4.7, $\forall i \in I. \bigvee X_i = \widehat{\bigcup X_i}$. Since for any $a \in A$ and $b \in B$, $\bigcup X_i$ contains $(a, \mathbf{F}_B)$ and $(\mathbf{F}_A, b)$, it already contains $(\bigvee x, \bigvee y)$ for either x or y empty, hence we have $\bigvee X_i = \widehat{\bigcup X_i} = \{(\bigvee x, \bigvee y) \mid x \subseteq A, y \subseteq B, \text{both non-empty and } x \times y \subseteq \bigcup X_i\}$.

$$\begin{array}{rcl}
F(f)(\lambda i. \bigvee X_i) & = & \bigvee\{f(\lambda i.(a_i, b_i)) \mid \forall i \in I.\ (a_i, b_i) \in \bigvee X_i\} \\
& = & \bigvee\{f(\lambda i.(\bigvee x_i, \bigvee y_i)) \mid \forall i \in I.\ x_i \subseteq A, y_i \subseteq B, \\
& & \qquad\qquad \text{both non-empty s.t. } x_i \times y_i \subseteq \bigcup X_i\} \\
\text{(by ilinearity)} & = & \bigvee\{f(\lambda i.(a_i, b_i)) \mid \forall i \in I.\ a_i \in x_i \subseteq A, b_i \in y_i \subseteq B, x_i \times y_i \subseteq \bigcup X_i\} \\
& = & \bigvee\{f(\lambda i.(a_i, b_i)) \mid \forall i \in I.\ (a_i, b_i) \in \bigcup X_i\} \\
& = & \bigvee\{f(\lambda i.(a_i, b_i)) \mid \forall i \in I.\ \exists S_i \in X_i.\ (a_i, b_i) \in S_i\} \\
& = & \bigvee\{\bigvee\{f(\lambda i.(a_i, b_i)) \mid \forall i \in I.\ (a_i, b_i) \in S_i\} \mid \forall j \in I.\ S_j \in X_j\} \\
& = & \bigvee\{F(f)(\lambda i.S_i) \mid \forall i \in I.\ S_i \in X_i\}\ .
\end{array}$$

We conclude that $F(f) : \Pi_{i \in I}(A \otimes B) \to C$ is ilinear. □

We finish this section with this final important result, showing that we can safely combine ilinear functions.

Proposition 5.3 *Suppose A and B are two completely distributive lattices, and consider two ilinear functions $f : (\Pi_{i \in I} A) \to A$ and $g : (\Pi_{i \in I} B) \to B$. Then $(f \otimes g) : (\Pi_{i \in I}(A \otimes B)) \to (A \otimes B)$ given by $F(u_{A,B} \circ (f \times g))$ is ilinear.*

Proof. Note that $u_{A,B}$ is linear, hence ilinear. So $(u_{A,B}) \circ (f \times g)$ is ilinear as well. Hence we get the result by a simple application of Lemma 5.2. □

There is an alternative, but equivalent, formulation for the above combination of ilinear functions. We define the tensor $\otimes^i$ of icomplete lattices, the *itensor*, using Definition 4.1 but replacing linearity with ilinearity. This gives us a bijective function $F^i : \Pi_{i \in I}(A \times B) \to (A \otimes^i B)$ between spaces of ilinear function. Using the adjunction $U \vdash\ !$, we can find an ilinear function $(UA \otimes^i UB) \to U(A \otimes B)$ over complete lattices A and B. We compose F^i with that function to retrieve our function F. Proving that these functions coincide however, requires some effort. As such, the approach in this paper uses the more direct, and theoretically less cumbersome, approach. Further comparison between the two tensors could be an interesting research topic for the future.

6 Combining effect descriptions

We combine two effect descriptions $(\Sigma_1, A_1, \alpha_1)$ and $(\Sigma_2, A_2, \alpha_2)$ into an effect description $(\Sigma_{12}, A_{12}, \alpha_{12})$, where $\Sigma_{12} := \Sigma_1 + \Sigma_2$, $A_{12} := (A_1 \otimes A_2)$, and α_{12} is defined by the following ilinear local functions:

For $\mathsf{op} \in \Sigma_1$, let $\alpha_{12,\mathit{inleft}(\mathsf{op})} : (A \otimes B)^{ar(\mathsf{op})} \to (A \otimes B)$ be $(\alpha_{1,\mathsf{op}} \otimes \bigwedge_{ar(\mathsf{op})})$.

For $\mathsf{op} \in \Sigma_2$, let $\alpha_{12,\mathit{inright}(\mathsf{op})} : (A \otimes B)^{ar(\mathsf{op})} \to (A \otimes B)$ be $(\bigwedge_{ar(\mathsf{op})} \otimes \alpha_{2,\mathsf{op}})$.

The combined effect interpretation is defined by combining the relevant interpretation of an effect operation from its source with the infimum function. Hence, since A_1 and B_1 are completely distributive, α_{12} is made up of ilinear local functions.

For ease of notation, we will write $\alpha_1 * \alpha_2$ for α_{12} when looking at examples.

Lemma 6.1 *Given some ilinear function $f : A^I \to A$ and infimum/conjunction $\bigwedge : \mathbb{B}^I \to \mathbb{B}$ on the Booleans. Then $(f \otimes \bigwedge)$ is under the isomorphism $(A \otimes \mathbb{B}) \simeq A$ from Lemma 4.2 equal to f.*

Proof. $(f \otimes \bigwedge)(\lambda i.S_i) = \bigvee\{u_{A,\mathbb{B}}(f(\lambda i.a_i), \bigwedge_{i\in I} b_i) \mid \forall i \in I.\ (a_i, b_i) \in S_i\}$. Since for any $S \in (A \otimes \mathbb{B})$ and all $a \in S$, $(a, \mathbf{F}) \in S$, we need only concern ourselves with the case where $\bigwedge_{i\in I} b_i \neq \mathbf{F}$, or equivalently, when $\forall i \in I.\ b_i = \mathbf{T}$. So $(f \otimes \bigwedge)(\lambda i.S_i) = \bigvee\{u_{A,\mathbb{B}}(f(\lambda i.a_i), \mathbf{T}) \mid \forall i \in I.(a_i, \mathbf{T}) \in S_i\}$. Using the isomorphism $(A \otimes \mathbb{B}) \simeq A$ we can transport $(f \otimes \bigwedge)$ to a function from A^I to A, which is given by sending $\lambda i.a_i$ to:
$\bigvee\{a \mid (a, \mathbf{T}) \in \bigvee\{u_{A,\mathbb{B}}(f(\lambda i.a'_i), \mathbf{T}) \mid \forall i \in I.\ (a'_i, \mathbf{T}) \in u_{A,\mathbb{B}}(a_i, \mathbf{T})\}\} = f(\lambda i.a_i)$.□

In general, for $\mathsf{op} \in \Sigma_1$, $\alpha_{12,\mathsf{op}}(\lambda i.S_i) = \bigvee\{u_{A,B}(\alpha_{1,\mathsf{op}}(\lambda i.a_i), b) \mid b \in B, \forall i \in I.\ (a_i, b) \in S_i\}$.

Proposition 6.2 *The method of combining effects gives a symmetric and associative operation on effect descriptions, with a unit given by pure computations $(\emptyset, \mathbb{B}, \downarrow)$.*

Proof. The sum on signatures Σ is a symmetric and associative operation, with a unit given by the empty set $\emptyset$. By Proposition 4.3, we know the tensor to be a symmetric and associative operation on complete lattices, with a unit given by $\mathbb{B}$. Lastly, the tensor combination of ilinear functions is symmetric, associative[4], and has a unit given by an infimum function over $\mathbb{B}$ (see Lemma 6.1). □

6.1 Examples of combining effects

To illustrate how the above method yields valid interpretations of combinations of effects, we look at a handful of examples. In each case, we add a specific effect to an arbitrary effect description. In Subsection 7.1, we will look at some more specific combinations of effects.

Example 6.3 [Adding nondeterminism] Take some effect description (Σ, A, α). To this, we add nondeterminism from Example 3.12, which has choice operator $\mathsf{nor} : \alpha^2 \to \alpha$, truth space $\mathbb{A}$ and interpretation Pos. The combined truth space $(A \otimes \mathbb{A})$ is given by the space of pairs $\{(a, b) \in A^2 \mid a \leq b\}$. Such a pair (a, b) represents a worst-case value a denoting the minimum degree of truth that is always reached, and a best case value b denoting the maximum degree of truth that could be reached. These correspond respectively to demonic and angelic nondeterminism.

Let β be the combined interpretation given by $\alpha * \mathsf{Pos}$. Our method yields the following local functions:

- $\forall \mathsf{op} \in \Sigma.\ \beta_{\mathit{inleft}(\mathsf{op})}(\lambda i.(a_i, b_i)) = (\alpha_{\mathsf{op}}(\lambda i.a_i), \alpha_{\mathsf{op}}(\lambda i.b_i))$.
- $\beta_{\mathit{inright}(\mathsf{nor})}((a_1, b_1), (a_2, b_2)) = (a_1 \wedge a_2, b_1 \vee b_2)$.

[4] This fact is not completely trivial, but is straightforward to prove.

We can give β an alternative description. The algebra β will resolve nondeterministic choices of an input tree both in the worst possible way and in the best possible way. It will then apply α to the resulting two trees, and return two values of A respectively.

Example 6.4 [Adding Global Store] Take some effect description (Σ, A, α). To this we add Boolean Global Store from Example 3.10, which has effect signature $\Sigma_{global} = \{\mathsf{lookup}_l, \mathsf{update}_l(\mathbf{T}), \mathsf{update}_l(\mathbf{F}) \mid l \in \mathsf{Loc}\}$, assertions $\mathcal{P}(\mathsf{S})$ and weakest precondition interpretation Wp. As shown in Lemma 4.4, the tensor of the two complete lattices is given by the function space $\mathsf{S} \to A$. We can see this as the space of *valuations* or quantitative A-valued assertions on global states S. Let β be the combination of interpretations $(\alpha * \mathsf{Wp})$, then we get:

- $\forall \mathsf{op} \in \Sigma.\ \beta_{\mathit{inleft}(\mathsf{op})}(\lambda i.f_i) = \lambda s \in \mathsf{S}.\alpha_{\mathsf{op}}(\lambda i.f_i(s))$.
- $\beta_{\mathit{inright}(\mathsf{lookup}_l)}(f_1, f_2) \ = \ \lambda s \in \mathsf{S}. \begin{cases} f_1(s) & \text{if } s(l) = \mathbf{T}, \\ f_2(s) & \text{if } s(l) = \mathbf{F}. \end{cases}$
- $\beta_{\mathit{inright}(\mathsf{update}_l(v))}(f) \ = \ \lambda s \in \mathsf{S}.f(s[l := v])$.

We can give β an alternative description. The algebra β will resolve, for each starting state s, the global store operations of its input tree appropriately, and give its leaves (which are functions from S to A) the final state as argument. To each resulting tree, it will apply α to compute the appropriate value in A.

Example 6.5 [Adding Cost] Take some effect description (Σ, A, α). To this, we add the cost effect from Example 3.11, which has effect signature $\Sigma_{cost} := \{\mathsf{cost}_q \mid q \in \mathbb{Q}_{>0}\}$, truth space $[0, \infty]$ with reverse order, and the tally interpretation Tal. The tensor of the truth spaces is given by functions $\mathbb{R}_{\geq 0} \to A$ which are infimum preserving with respect to the standard ordering of the real numbers. Technically, it is functions from $[0, \infty]$, but ∞ is always sent to $\mathbf{T}_A$, so we can remove it from the definition without loss of generality.

Let $\beta = (\alpha * \mathsf{Tal})$ be the combination of interpretations. Then:

- For $\mathsf{op} \in \Sigma$, $\beta_{\mathit{inleft}(\mathsf{op})}(\lambda i.f_i) \ = \ \lambda r \in \mathbb{R}_{\geq 0}.\ \alpha_{\mathsf{op}}(\lambda i.f_i(r))$.
- $\beta_{\mathit{inright}(\mathsf{cost}_q)}(f) \ = \ \lambda r \in \mathbb{R}_{\geq 0}.\ f(r + q)$.

We can give β an alternative description. The algebra β will, given a certain allowance $r \in \mathbb{R}_{\geq 0}$ to spend, go through the tree spending the allowance on resolving any cost operations. Once it encounters a cost operation it cannot pay for, it puts a $\perp$ leaf at that location. If it encounters another leaf, which contains a function from $\mathbb{R}_{\geq 0}$ to A, it feeds this function the remaining allowance as an argument and puts the result as the new value of the leaf. It then applies α to the resulting tree to compute the appropriate value of A.

7 Equations and interaction laws

Given some countable set of variables $\mathbb{V}$, we see an element of $T^{\nu}_{\Sigma}(\mathbb{V})$ as an *algebraic expression*. A pair of algebraic expressions $(e_1, e_2) \in T^{\nu}_{\Sigma}(\mathbb{V}) \times T^{\nu}_{\Sigma}(\mathbb{V})$ expresses an *equation* $(e_1 = e_2)$ or *inequation* $(e_1 \leq e_2)$, denoting whether they are related in some way. A set of such (in)equations form a relation on algebraic expressions $\mathcal{R} \subseteq (T^{\nu}_{\Sigma}(\mathbb{V}))^2$, which can be specified by choosing a set of axioms $\mathcal{A} \subseteq (T^{\nu}_{\Sigma}(\mathbb{V}))^2$ appropriate to the effect, and closing this set under a couple of proof rules, e.g. *compositionality*:

$$\forall (e_1, e_2) \in \mathcal{R}, \forall f, g : \mathbb{V} \to T^{\nu}_{\Sigma}(\mathbb{V}). \quad (\forall x \in \mathbb{V}.\ (f(x), g(x)) \in \mathcal{R}) \implies (T^{\nu}_{\Sigma}(f)(e_1), T^{\nu}_{\Sigma}(g)(e_2)) \in \mathcal{R}\ .$$

An axiom we tend to assume for each effect example is the inequation $(\bot \leq x)$ where $x \in \mathbb{V}$. This reflects the fact that a diverging computation does not produce anything observable.

An EM-algebra $\alpha : T^{\nu}_{\Sigma}(A) \to A$ on some preorder specifies a relation $\mathcal{R}_{\alpha} \subseteq (T^{\nu}_{\Sigma}(\mathbb{V}))^2$ as follows:

$$\forall e_1, e_2 \in T^{\nu}_{\Sigma}(\mathbb{V}), \qquad (e_1, e_2) \in \mathcal{R}_{\alpha} \quad \iff \quad \forall f : \mathbb{V} \to A.\ \alpha(T^{\nu}_{\Sigma}(f)(e_1)) \leq \alpha(T^{\nu}_{\Sigma}(f)(e_2))\ .$$

If α is an EM-algebra, then $\mathcal{R}_{\alpha}$ is compositional. More on this comparison can be found in [31].

We say that an algebra α *complements* a set of axioms $\mathcal{A}$ if they generate the same algebraic relation. The word 'complement' expresses their opposing nature: whereas equations state equality between programs, algebras are used to find distinctions between programs (see Examples 7.2 and 7.6 for such a distinctions). An equation (e_1, e_2) holds for α if $(e_1, e_2) \in \mathcal{R}_{\alpha}$. This direct correspondence allows us to compare the method of combining effects defined in this paper with traditional methods for combining equational theories of effects [14,15].

Firstly, we observe that equations which holds for the individual effects still holds for the combination of effects. Note that for two effect signatures $\Sigma \subseteq \Sigma'$, $T^{\nu}_{\Sigma}(\mathbb{V}) \subseteq T^{\nu}_{\Sigma'}(\mathbb{V})$. For simplicity, we consider $\Sigma_1 + \Sigma_2 = \Sigma_1 \cup \Sigma_2$, hence it contains both Σ_1 and Σ_2.

Proposition 7.1 *Given two effect descriptions* $(\Sigma_1, A_1, \alpha_1)$ *and* $(\Sigma_2, A_2, \alpha_2)$, *let* $(\Sigma_{12}, A_{12}, \alpha_{12})$ *be their combination. Then both* $\mathcal{R}_{\alpha_1} \subseteq T^{\nu}_{\Sigma_1}(\mathbb{V})$ *and* $\mathcal{R}_{\alpha_2} \subseteq T^{\nu}_{\Sigma_1}(\mathbb{V})$ *are contained in* $\mathcal{R}_{\alpha_{12}} \subseteq T^{\nu}_{\Sigma_{12}}(\mathbb{V})$.

Proof. Consider $t \in T^{\nu}_{\Sigma_1}((A \otimes B)^{\mathcal{P}})$, hence t only has internal nodes from Σ_1. For $b \in B$, we define the function $f_b : (A \otimes B)^{\mathcal{P}} \to A$ by $S \mapsto \bigvee\{a \in A \mid (a, b) \in S\}$. We prove that $\widehat{\alpha_{12}}(t) = \bigvee\{u_{A,B}(\widehat{\alpha_1}(T^{\nu}_{\Sigma_1}(f_b)(t)), b) \mid b \in B\}$. We start with an induction over finite trees, using the local functions. First the base cases.

If $t = \bot$, then $\alpha_{12}(t) = \mathbf{F}_{A,B} = \bigvee\{u_{A,B}(\mathbf{F}_A, b) \mid b \in B\} = \bigvee\{u_{A,B}(\alpha_1(T^{\mu}_{\Sigma_1}(f_b)(t)), b) \mid b \in B\}$.

If $t = \langle S \rangle$, then $\alpha_{12}(t) = S = \bigvee\{u_{A,B}(a, b) \mid (a, b) \in S\} = \bigvee\{u_{A,B}(f_b(S), b) \mid b \in B\} = \bigvee\{u_{A,B}(\alpha_1(T^{\mu}_{\Sigma_1}(f_b)(t)), b) \mid b \in B\}$.

If $t = \mathsf{op}\langle t_1, \ldots, t_n \rangle$, then

$$\begin{aligned}
\alpha_{12}(t) = {} & \alpha_{12,\mathsf{op}}(\alpha_{12}(t_1), \dots, \alpha_{12}(t_n)) \\
= {} & \alpha_{12,\mathsf{op}}(\bigvee\{u_{A,B}(\alpha_1(T^{\mu}_{\Sigma_1}(f_b)(t_1)), b) \mid b \in B\}, \dots, \bigvee\{u_{A,B}(\alpha_1(T^{\mu}_{\Sigma_1}(f_b)(t_n)), b) \mid b \in B\}) \\
= {} & \bigvee\{\alpha_{12,\mathsf{op}}(u_{A,B}(\alpha_1(T^{\mu}_{\Sigma_1}(f_{b_1})(t_1)), b_1), \dots, u_{A,B}(\alpha_1(T^{\mu}_{\Sigma_1}(f_{b_n})(t_n)), b_n)) \mid b_1, \dots, b_n \in B\} \\
= {} & \bigvee\{u_{A,B}(\alpha_{1,\mathsf{op}}(a_1, \dots, a_n), \textstyle\bigwedge_i b'_i) \mid \forall i \in I.\ b_i \in B, (a_i, b'_i) \in u_{A,B}(\alpha_1(T^{\mu}_{\Sigma_1}(f_{b_i})(t_i)), b_i)\} \\
= {} & \bigvee\{u_{A,B}(\alpha_{1,\mathsf{op}}(a_1, \dots, a_n), b) \mid \forall i, b_i \in B, b \in B, (a_i, b) \in u_{A,B}(\alpha_1(T^{\mu}_{\Sigma_1}(f_{b_i})(t_i)), b_i)\} \\
= {} & \bigvee\{u_{A,B}(\alpha_{1,\mathsf{op}}(a_1, \dots, a_n), b) \mid \forall i, a_i \in A, b \in B, a_i \leq \alpha_1(T^{\mu}_{\Sigma_1}(f_b)(t_i))\} \\
= {} & \bigvee\{u_{A,B}(\alpha_{1,\mathsf{op}}(\alpha_1(T^{\mu}_{\Sigma_1}(f_b)(t_1)), \dots, \alpha_1(T^{\mu}_{\Sigma_1}(f_b)(t_n))), b) \mid b \in B\} \\
= {} & \bigvee\{u_{A,B}(\alpha_1(T^{\mu}_{\Sigma_1}(f_b)(t_1)), b) \mid b \in B\} \quad .
\end{aligned}$$

Which is what we wanted to prove. For an infinite tree t approximated by finite trees $t_1, t_2, t_3, \dots$:
$\widehat{\alpha_{12}}(t) \;=\; \widehat{\alpha_{12}}(\bigvee_i t_i) \;=\; \bigvee_i \alpha_{12}(t_i) \;=\; \bigvee_i \bigvee\{u_{A,B}(\alpha_1(T^{\mu}_{\Sigma_1}(f_b)(t_i)), b) \mid b \in B\}$ $=\; \bigvee\{u_{A,B}(\bigvee_i \alpha_1(T^{\mu}_{\Sigma_1}(f_b)(t_i)), b) \mid b \in B\} \;=\; \bigvee\{u_{A,B}(\widehat{\alpha_1}(T^{\nu}_{\Sigma_1}(f_b)(\bigvee_i t_i)), b) \mid b \in B\}$.

Consider an equation $(e_1, e_2) \in (T^{\nu}_{\Sigma_1}(\mathbb{V}))$ which holds for α_1. Then, given $f : \mathbb{V} \to (A \otimes B)$, we know by the above result that $\widehat{\alpha_{12}}(T^{\nu}_{\Sigma_1}(f)(e_1)) = \widehat{\alpha_{12}}(T^{\nu}_{\Sigma_1}(f)(e_2))$, hence the equation holds for α_{12}. □

In particular, if α_1 complements a set of axioms $\mathcal{A}$, then all equations from $\mathcal{A}$ still hold for α_{12}. We conclude that equations are preserved by our method of combining effect descriptions.

7.1 Comparing methods of combining effects

Given two sets of axioms $\mathcal{A}_1 \subseteq (T^{\nu}_{\Sigma_1}(\mathbb{V}))^2$ and $\mathcal{A}_2 \subseteq (T^{\nu}_{\Sigma_2}(\mathbb{V}))^2$, then we define the axioms of the *equational sum* [14,15] of effects to be $\mathcal{A}_1 \cup \mathcal{A}_2$. In some cases, the interpretation of a combination of effects defined in this paper corresponds to the equational sum:

Example 7.2 [Cost with Error] We combine the effects of cost in Example 3.11, and error in Example 3.13, using Example 6.5. The resulting combination of effects coincides with the sum of equational theories. We can see that for any $q \in \mathbb{Q}_{>0}$, the effect operation cost_q does not distribute over raise. This is because $\mathsf{cost}_q(\mathsf{raise}())$ and $\mathsf{raise}()$ are distinguished by the combined algebra $(\mathsf{Tal} * \mathsf{Err})$:

$$(\mathsf{Tal} * \mathsf{Err})(\mathsf{raise}())(0) \;=\; \diamond \;\neq\; \mathbf{F} \;=\; (\mathsf{Tal} * \mathsf{Err})(\mathsf{cost}_q(\mathsf{raise}()))(0) \;\; .$$

Informally, if we have no resources (the 0 argument), then $\mathsf{raise}()$ will yield an error (the $\diamond$), whilst $\mathsf{cost}_q(\mathsf{raise}())$ will stall (the $\perp$) as it requests a resource we do not have. We can use this as a basis to prove that our combined algebra complements the sum of equational theories..

In other cases, extra axioms are needed to describe the interaction between the effects we want to combine. Such axioms, which contain effect operations from both theories, are called *interaction laws.* The most common interaction law is a *commutativity law* [14,15].

Definition 7.3 Given two effect operations op_1 and op_2 of arity n and m respectively, the *commutativity* law between op_1 and op_2 is given by:

$$\mathsf{op}_1(\lambda i.\ \mathsf{op}_2(\lambda j.\ v_{i,j})) \;=\; \mathsf{op}_2(\lambda j.\ \mathsf{op}_1(\lambda i.\ v_{i,j})) \;\; ,$$

where we use a distinct variable $v_{i,j} \in \mathbb{V}$ for each $1 \leq i \leq n$ and $1 \leq j \leq m$.[5]

Let $\mathsf{commut}(\Sigma_1, \Sigma_2) \subseteq T^{\nu}_{\Sigma_1+\Sigma_2}(\mathbb{V})$ be the set containing the commutativity law for each pair $\mathsf{op}_1 \in \Sigma_1$ and $\mathsf{op}_2 \in \Sigma_2$. Given two sets of axioms $\mathcal{A}_1 \subseteq (T^{\nu}_{\Sigma_1}(\mathbb{V}))^2$ and $\mathcal{A}_2 \subseteq (T^{\nu}_{\Sigma_2}(\mathbb{V}))^2$, then we define the axioms of the *equational tensor* [15] of effects to be $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathsf{commut}(\Sigma_1, \Sigma_2)$. Note that these will still be closed under operations like compositionality, when used to generate an algebraic relation.

In some cases, the description of a combination of effects as defined in this paper corresponds to the equational tensor:

Example 7.4 [Probability with Global store] We combine the effects of probability in Example 3.9, and global store in Example 3.10, using Example 6.4. We will show that the relation $\mathcal{R}_\alpha$ induced by the combined algebra $\alpha = \mathsf{Exp} * \mathsf{Wp}$ contains all the commutativity laws. From Example 6.4, we know that the combined truth space is given by $A = (\mathsf{S} \to [0,1])$. Now, por commutes with lookup_l since, given $a, b, c, d \in A$:

$$\begin{aligned}
\alpha(\mathsf{por}(\mathsf{lookup}_l(a,b), \mathsf{lookup}_l(c,d))) &= \lambda s.(\alpha(\mathsf{lookup}_l(a,b))(s) + \alpha(\mathsf{lookup}_l(c,d))(s))/2 \\
&= \lambda s. \begin{cases} (a(s)+c(s))/2 & \text{if } s(l) = \mathbf{T} \\ (b(s)+d(s))/2 & \text{if } s(l) = \mathbf{F} \end{cases} \\
&= \alpha(\mathsf{lookup}_l(\mathsf{por}(a,c), \mathsf{por}(b,d))) \quad .
\end{aligned}$$

The operation por commutes with $\mathsf{update}_{l,\mathbf{T}}$ since, given $a, b \in A$:

$$\begin{aligned}
\alpha(\mathsf{por}(\mathsf{update}_{l,\mathbf{T}}(a), \mathsf{update}_{l,\mathbf{T}}(b))) &= \lambda s.(\alpha(\mathsf{update}_{l,\mathbf{T}}(a))(s) + \alpha(\mathsf{update}_{l,\mathbf{T}}(b))(s))/2 \\
&= \lambda s.(a(s[l := \mathbf{T}]) + b(s[l := \mathbf{T}]))/2 \\
&= \lambda s.\alpha(\mathsf{por}(a,b))(s[l := \mathbf{T}]) \\
&= \alpha(\mathsf{update}_{l,\mathbf{T}}(\mathsf{por}(a,b))) \quad .
\end{aligned}$$

Similarly, the operation por commutes with $\mathsf{update}_{l,\mathbf{F}}$. Hence, $\mathcal{R}_\alpha$ contains $\mathsf{commut}(\Sigma_{prob}, \Sigma_{global})$, the commutativity laws. We conclude that α complements the tensor of equational theories.

In some famous cases, the equational sum coincides with the equational tensor. This is because the commutativity laws are already present in the original sets of axioms.

Example 7.5 [Nondeterminism with Error] We combine the effects of nondeterminism in Example 3.12, and error in Example 3.13. As seen in Proposition 7.1, the relation $\mathcal{R}_\alpha$ induced by the combined interpretation $\alpha = (\mathsf{Pos} * \mathsf{Err})$ contains the original axioms of the theory. One such axiom is that of idempotency, that $\mathsf{nor}(x,x) = x$. The commutativity law between nor and raise is given by $\mathsf{nor}(\mathsf{raise}, \mathsf{raise}) = \mathsf{raise}$, which can be proven using idempotency, substituting raise for x. Hence the combination of effects complements both the sum and the tensor of equational theories

[5] The choice of variables is unimportant, since the resulting (induced) relation on $T^{\nu}_{\Sigma}(\mathbb{V})$ will be closed under substitution. Note also that the definition of commutativity law can be easily adapted to talk about operations with countable arity.

(since they are identical).

Lastly, there is an instance in which the method of combining effects neither corresponds to the equational sum, nor with the equational tensor:

Example 7.6 [Probability with Nondeterminism] We combine the effects of probability in Example 3.9, and nondeterminism in Example 3.10, using Example 6.3. The combined truth space A is given by ordered pairs (a, a') of elements $a, a' \in [0,1]$. Let $\alpha = (\mathsf{Exp} * \mathsf{Pos})$. We first show that this algebra does not complement the sum of equational theories, since the interaction law $\mathsf{por}(x, \mathsf{nor}(y,z)) = \mathsf{nor}(\mathsf{por}(x,y), \mathsf{por}(x,z))$ holds for α. Take $(a,a'), (b,b'), (c,c') \in A$:

$$\begin{aligned}\alpha(\mathsf{por}((a,a'), \mathsf{nor}((b,b'),(c,c')))) &= ((a + (b \wedge c))/2,\ (a' + (b' \vee c')/2)) \\ &= ((a+b)/2 \wedge (a+c)/2,\ (a'+b')/2 \vee (a'+c')/2) \\ &= \alpha(\mathsf{nor}(\mathsf{por}((a,a'),(b,b')), \mathsf{por}((a,a'),(b,b'))))\quad .\end{aligned}$$

However, the commutativity law $\mathsf{por}(\mathsf{nor}(x,y), \mathsf{nor}(z,w)) = \mathsf{nor}(\mathsf{por}(x,z), \mathsf{por}(y,w))$ does not holds for α, since for $x = (0,0)$, $y = w = (1/4, 1/4)$, and $z = (1,1)$:

$$\begin{aligned}\alpha(\mathsf{por}(\mathsf{nor}(x,y), \mathsf{nor}(z,w))) &= (((0 \wedge 1/4) + (1 \wedge 1/4))/2,\ ((0 \vee 1/4) + (1 \vee 1/4))/2) \\ &= (1/8, 5/8) \neq (1/4, 1/2) \\ &= ((0+1)/2 \wedge (1/4+1/4)/2, (0+1)/2 \vee (1/4+1/4)/2) \\ &= \alpha(\mathsf{nor}(\mathsf{por}(x,z), \mathsf{por}(y,w)))\quad .\end{aligned}$$

Hence, the combined interpretation α neither complements the sum, nor complements the tensor of equational theories. Instead, it complements a natural theory for this combination of effects, as e.g. described in [18,19].

8 Conclusions and related work

In this paper, we looked at Eilenberg-Moore algebras over the tree monad, whose carrier sets are given by complete lattices. These are used in [29,30] to formulate behavioural equivalence, in particular to define *quantitative modalities*: lifting quantitative predicates on values to quantitative predicates on computations. These quantitative modalities are a generalisation of Boolean modalities, given by subsets of $T^{\nu}_{\Sigma}(\{*\})$, and used in [27,22] to specify program equivalence. In turn, those modalities are based on the notion of *observation* from [17].

Eilenberg-Moore algebras describing weakest preconditions, as done in our example for global store, are commonly used for describing effectful programs, see e.g. [12]. In [2,20], weakest precondition semantics are given in terms of *Dijkstra monads*. It would be interesting to see whether the theory developed here has practical applications to the formalism developed in those papers.

This paper only features a selection of examples, though a lot more effects, like Input/Output, can be given an effect description. It is also possible to implement the jump effect, which is given an algebraic description in [10]. Moreover, though we only looked at a couple of combinations of effects, we can use our method for combining effect descriptions to give a description of any combination of the effects featured in this paper. This includes combining effects with themselves, e.g. nondeterminism with nondeterminism would create a description of a game between two nondeterministic schedulers.

One interesting combination we did not look at specifically is the combination of global store and error. It turns out, that this combination coincides with the tensor of equational theories. Informally, this describes the situation in which the global store is inaccessible after an error has been raised. It is also possible to define an algebra complementing the sum of equational theories, as defined in [29,30]. We can tweak interpretations of combinations of effects in other ways too. E.g., when combining probability with cost, we can associate a cost to each probabilistic choice. This would induce a logic capable of testing the average time till convergence.

We focussed on effect operation with finite arity in this paper. As mentioned before, it is also possible to extend the results to include operations with infinite arity. This for instance allows us to describe global store for storing natural numbers. However, not all examples with infinite arity can be incorporated. Take for instance countable nondeterminism. It is not possible to inductively check must-termination over countable choice. So, though it is possible to define local ilinear functions to describe this (e.g. $\bigwedge$), the induced algebra cannot appropriately detect must-termination (since $\bigwedge'$ as defined in (1) is the constant false function). In general, when looking at local functions with infinite arity, one needs to be careful to check if they are ω-continuous.

Lastly, the algebras used in this paper seem suitable for defining quantitative relations, e.g. metrics [11,21], on functional programming languages with effects. For metrics, *quantales* tend to be used as space of distances. Since completely distributive lattices form quantales, there is a possible link between such metrics and the algebras used in this paper. It could be possible to develop a formalism for using algebras for defining quantitative relations, and use the method for combining effects to combine such quantitative relations. This would be an interesting avenue for future research.

Acknowledgements:
I would like to thank both Alex Simpson and Danel Ahman for the helpful tips and discussions, and the anonymous reviewers for their insightful feedback and suggestions.

References

[1] Abramsky, S., *The lazy λ-calculus*, Research Topics in Functional Programming (1990), pp. 65–117.

[2] Ahman, D., C. Hritcu, K. Maillard, G. Martínez, G. Plotkin, J. Protzenko, A. Rastogi and N. Swamy, *Dijkstra monads for free*, in: *44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)*, 2017, pp. 515–529.

[3] Battenfeld, I., K. Keimel and T. Streicher, *Observationally-induced algebras in domain theory*, Logical Methods in Computer Science **10(3:18)** (2014), pp. 1–26.

[4] Battenfeld, I. and M. Schröder, *Observationally-induced effect monads: Upper and lower powerspace constructions*, Electron. Notes Theor. Comput. Sci. **276** (2011), p. 105–119.

[5] Crubillé, R. and U. Dal Lago, *On probabilistic applicative bisimulation and call-by-value λ-calculi*, Lecture Notes in Computer Science **8410** (2014).

[6] Dal Lago, U., F. Gavazzo and P. B. Levy, *Effectful applicative bisimilarity: Monads, relators, and the Howe's method*, Logic in Computer Science (2017), pp. 1–12.

[7] Davey, B. A. and H. A. Priestley, "Introduction to Lattices and Order," Cambridge University Press, 2002, 2nd edition.

[8] Eklund, P., J. Gutiérrez García, U. Höhle and J. Kortelainen, "Semigroups in Complete Lattices: Quantales, Modules and Related Topics," 2018.

[9] Erné, M. and J. Picado, *Tensor products and relation quantales*, Algebra Univers. **78** (2017), p. 461–487.

[10] Fiore, M. and S. Staton, *Substitution, jumps, and algebraic effects*, in: *Proceedings of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, CSL-LICS '14, 2014, pp. 41:1–41:10.

[11] Gavazzo, F., *Quantitative behavioural reasoning for higher-order effectful programs: Applicative distances*, in: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018*, 2018, pp. 452–461.

[12] Hasuo, I., *Generic weakest precondition semantics from monads enriched with order*, Theoretical Computer Science **604** (2015), pp. 2–29.

[13] Heckmann, R., *Abstract valuations: A novel representation of plotkin power domain and vietoris hyperspace*, Electr. Notes Theor. Comput. Sci. **6** (1997), pp. 160–173.

[14] Hyland, M., G. Plotkin and J. Power, "Combining Computational Effects: Commutativity and Sum," Springer US, Boston, MA, 2002 pp. 474–484.

[15] Hyland, M., G. Plotkin and J. Power, *Combining effects: Sum and tensor*, Theoretical Computer Science **357** (2006), pp. 70 – 99.

[16] Jacobs, B., *Semantics of weakening and contraction*, Annals of Pure and Applied Logic **69** (1994), pp. 73 – 106.

[17] Johann, P., A. Simpson and J. Voigtländer, *A generic operational metatheory for algebraic effects*, in: *Proceedings of Logic in Computer Science (LICS'10)*, 2010, pp. 209–218.

[18] Keimel, K. and G. D. Plotkin, *Mixed powerdomains for probability and nondeterminism*, Logical Methods in Computer Science **13** (2017).

[19] Lopez, A. and A. Simpson, *Basic operational preorders for algebraic effects in general, and for combined probability and nondeterminism in particular*, in: *27th EACSL Annual Conference on Computer Science Logic, CSL 2018*, 2018, pp. 29:1–29:17.

[20] Maillard, K., D. Ahman, R. Atkey, G. Martínez, C. Hritcu, E. Rivas and E. Tanter, *Dijkstra monads for all*, in: *24th ACM SIGPLAN International Conference on Functional Programming (ICFP)*, 2019.

[21] Mardare, R., P. Panangaden and G. D. Plotkin, *Quantitative algebraic reasoning*, in: *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16*, 2016, pp. 700–709.

[22] Matache, C. and S. Staton, *A sound and complete logic for algebraic effects*, in: *Lecture Notes in Computer Science*, Springer International Publishing, 2019 pp. 382–399.

[23] Plotkin, G. D., *Domains* (1983), course notes.

[24] Plotkin, G. D. and J. Power, *Adequacy for algebraic effects*, Foundations of Software Science and Computation Structures (2001), pp. 1–24.

[25] Plotkin, G. D. and J. Power, *Notions of computation determine monads*, in: *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures*, 2002, pp. 342–356.

[26] Sigmon, K., *Cancellative medial means are arithmetic*, Duke Math. J. **37** (1970), pp. 439–445.

[27] Simpson, A. and N. Voorneveld, *Behavioural equivalence via modalities for algebraic effects*, ACM Trans. Program. Lang. Syst. **42** (2020), 45 pages. Journal version of ESOP 2018 conference paper.

[28] Sünderhauf, P., *Tensor products and powerspaces in quantitative domain theory*, Electronic Notes in Theoretical Computer Science **6** (1997), pp. 327 – 347, MFPS XIII, Mathematical Foundations of Progamming Semantics, Thirteenth Annual Conference.

[29] Voorneveld, N., *Quantitative logics for equivalence of effectful programs*, Electronic Notes in Theoretical Computer Science **347** (2019), pp. 281 – 301, proceedings of the Thirty-Fifth Conference on the Mathematical Foundations of Programming Semantics.

[30] Voorneveld, N., "Equality between programs with effects," Ph.D. thesis, University of Ljubljana (2020).

[31] Voorneveld, N., *From equations to distinctions: Two interpretations of effectful computations*, in: Proceedings Eighth Workshop on *Mathematically Structured Functional Programming*, Electronic Proceedings in Theoretical Computer Science **317** (2020), pp. 1–17.

[32] Wille, R., *Tensorial decomposition of concept lattices*, Order **2** (1985), p. 81–95.