Monads and Comonads in Intensional Semantics

Stephen Brookes Kathryn Van Stone

April 1993
CMU-CS-93-140

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Kleisli categories over monads have been used in denotational semantics to describe functional languages using various notions of computations as values. Kleisli categories over comonads have also been used to describe intensional semantics rather than extensional. This paper explores the possibilities of combining monads and comonads to obtain an intensional semantics using computations as values. We give three alternative ways to combine the two and explore which apply to known monads and comonads of interest. We will also look at various intensional semantics for an example programming language that uses monads for computations and compare them to the original extensional semantics.

This research was supported in part by National Science Foundation grant CCR-9006064.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA or the U.S. government.

Keywords: theory, denotational semantics, applicative languages, algebraic approaches to semantics

1. Introduction

Traditionally most denotational semantic interpretations of a functional programming language interpret a program as a function from environments to values. Since these semantics focus exclusively on input-output, or extensional behavior, they cannot easily be used to examine intensional properties of programs, such as order of evaluation or complexity. Also, since they typically use simple values, it is difficult to extend them to reason about programming languages with non-functional elements such as nondeterminism, error handling, and assignments.

When adding such features to a language, originally the denotational semantics was changed on a case-by-case basis, by adding new "values" to represent invalid results or by changing the structure of the values, such as using sets of values for non-deterministic results. Moggi in [14] showed that many of these techniques can be described uniformly using an algebraic structure in category theory called a *monad*. With a monad, one can develop a formal semantics using category theory for a variety of functional languages that also contain many non-functional elements.

There has been some work examining intensional properties for functional languages. Berry and Curien [5] developed a semantics using the cartesian closed category of concrete data structures and sequential algorithms, which include information on the order of evaluation as well as the final value. Brookes and Geva [6] looked at expanding Berry and Curien's results to parallel computation as well as sequential, and developed general notion of intensional semantics using *comonads*.

This paper explores the possibilities of combining both methods: using comonads to examine the intensional properties of programs whose extensional properties are modeled by means of a monad. Section 2 defines and gives notation for many of the concepts contained in this paper. It is assumed that the reader has a general knowledge of category theory and domains. Sections 3 and 4 define monads and comonads and give examples that have been useful for examining programs. The monads and comonads are typically defined in as general a fashion as possible, with examples given in specific categories, usually **Cont**, the category of Scott domains and continuous functions. Section 5 describes the formation of the Kleisli categories from comonads or monads. Sections 6 and 7 describe ways to combine comonads and monads (see [4]) and examines how they work with the examples given earlier. The last section then looks at a simple programming language from [14] and at various semantics using comonads and monads together.

2. Preliminary definitions and notational conventions

2.1. Category theory

There are many books containing the basic categorical concepts used in this paper. The technical report [15] is a good introduction, and [3] is aimed for computer scientists.

A category has *binary products* if for each pair of objects A and B there is an object $A \times B$ with projection morphisms $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$ such that for all morphisms $f : C \to A$ and $g : C \to B$, there is a unique morphism $\langle f, g \rangle$ from C to $A \times B$ satisfying $\pi_1 \circ \langle f, g \rangle = f$ and $\pi_2 \circ \langle f, g \rangle = g$. For all morphisms $f : A \to C$ and $g : B \to D$, we define $f \times g : A \times B \to C \times D$ to be $\langle f \circ \pi_1, g \circ \pi_2 \rangle$.

Similarly a category has *binary coproducts* if for every pair of objects A and B, there is an object $A + B$ and a pair of injection morphisms $\iota_1 : A \to A + B$ and $\iota_2 : B \to A + B$, such that for all morphisms $f : A \to C$

and $g : B \to C$, there is a unique morphism $[f, g]$ from $A+B$ to C satisfying $[f, g] \circ \iota_1 = f$ and $[f, g] \circ \iota_2 = g$. For all morphisms $f : A \to C$ and $g : B \to D$, we define $f + g : A + B \to C + D$ to be $[\iota_1 \circ f, \iota_2 \circ g]$.

An object $\mathbf{1}$ of a category is *terminal* if for every object A, there exists a unique morphism, denoted $!_A$, from A to $\mathbf{1}$.

A category is *cartesian closed* if it has binary products, a terminal object, and if for each pair of objects A and B there is an exponentiation object $[A \to B]$ and a morphism $\mathsf{app}_{A,B} : [A \to B] \times A \to C$ such that for all morphisms $f : C \times A \to B$, there is a unique morphism $\mathsf{curry}(f) : C \to [A \to B]$ satisfying $f = \mathsf{app}_{A,B} \circ (\mathsf{curry}(f) \times \mathsf{id}_A)$. Here, as usual, id_A denotes the identity morphism for A. Given a morphism $g : C \to [A \to B]$ we define $\mathsf{uncurry}(g) : C \times A \to B$ to be $\mathsf{app}_{A,B} \circ (g \times \mathsf{id}_A)$.

A cartesian closed category is *set-like* if its objects are sets, its morphisms are functions, its products are the usual cartesian products of sets, exponentiation objects $[A \to B]$ are subsets of the set of functions from A to B, and application is the standard function application.

2.2. Partial orders

The definitions for domains and cpo's in this section come primarily from [10]; the discussion on Plotkin orders it taken out of [9].

A *poset* is a set D with a partial order $\leq^D$, i.e. a binary relation that is reflexive, transitive, and anti-symmetric. A subset X of a poset is *directed* if every finite subset of X has an upper bound in X. A partial order $(D, \leq^D)$ is *directed complete* if for every directed set $X \subseteq D$, the least upper bound of X, denoted $\sqcup X$, exists in D. A partial order is *pointed* if it has a least element, usually denoted $\bot$. A *cpo* is a pointed directed complete partial order.

A subset X of a poset is *consistent* or *bounded* if it has an upper bound. A partial order is *bounded-complete* if every consistent set X has a least upper bound. In a bounded-complete poset, every pair of elements x, y with a lower bound has a greatest lower bound, written as $x \sqcap y$.

An element k in a cpo D is *compact* (also called *finite* or *isolated*) if for every directed set X such that $k \leq^D \sqcup X$, k is less than or equal to some element in X. The set of compact elements of D is written as $K(D)$. A cpo D is *algebraic* if for every $x \in D$, the set $\{k \in K(D) \mid k \leq^D x\}$ is directed and its least upper bound is x. A cpo D is *ω-algebraic* [9] if it is algebraic and the set $K(D)$ is countable. A *Scott domain* is a bounded-complete, ω-algebraic cpo. In this paper we will often use *domain* for Scott domain.

A *bc-domain* is an algebraic bounded-complete cpo. A bc-domain is *distributive* if for all consistent pairs of elements y and z and all elements x we have that $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$. An algebraic cpo has *property I* if for each element x there are only finitely many compact elements below it. A *dI-domain* is a distributive bc-domain that has property I.

A subset N of a poset A is *normal* if for every $x \in A$, the set $N \cap \{y \in A \mid y \leq^A x\}$ is directed. A poset A is a *Plotkin order* if for every finite subset X of A, there is a finite normal subset N of A with $X \subseteq N$. A *bifinite domain* is an ω-algebraic cpo D such that $K(D)$ is a Plotkin order.

A function f is *monotone* if $f(x) \leq f(y)$ whenever $x \leq y$. A function is *continuous* if it is monotone and preserves least upper bounds of directed sets. A function is *strict* if it maps least elements to least elements. A continuous function $f : A \to B$ is *stable* if for all $b \leq f(a)$ the set $\{a' \leq a \mid b \leq f(a')\}$ has a least element, represented as $M(f, a, b)$. Given two functions f_1 and f_2 from A to B, we define the *pointwise* ordering,

$\leq^{A\to B}$ such that $f_1 \leq^{A\to B} f_2$ if for all $x \in A, f_1(x) \leq^B f_2(x)$. We also define the *stable* ordering, $\leq_s^{A\to B}$ such that $f_1 \leq_s^{A\to B} f_2$ if $f_1 \leq^{A\to B} f_2$ and for all $b \leq f_1(a)$, $M(f_1, a, b) = M(f_2, a, b)$.

Given two posets D_1 and D_2, their *separated sum* $D_1 + D_2$ is a domain consisting of the set $\{\langle 1, x_1\rangle \mid x_1 \in D_1\} \cup \{\langle 2, x_2\rangle \mid x_2 \in D_2\} \cup \{\bot\}$ with the ordering $\leq^{D_1+D_2}$ defined by $\langle i, x\rangle \leq^{D_1+D_2} \langle i, x'\rangle$ whenever $x \leq^{D_i} x'$ ($i = 1$ or 2) and for all $y \in D_1 + D_2$, $\bot \leq^{D_1+D_2} y$. The *coalesced sum* $D_1 \oplus D_2$ of D_1 and D_2 is $(D_1 - \{\bot_{D_1}\}) + (D_2 - \{\bot_{D_2}\})$.

2.3. Specific categories

The categories used for the examples in this paper will generally be drawn from the following list:

Set: The category of sets and functions.

Cont: The category of Scott domains and continuous functions.

Cont$_S$: The category of Scott domains and strict continuous functions.

Bif: The category of bifinite domains and continuous functions. The only interest in this category is that bifinite domains are closed under the Plotkin power domain (see section 3.5), but Scott domains are not [9].

dI: The category of dI-domains and stable functions, stably ordered.

All the categories listed above are set-like, except for **Cont$_S$** which is not cartesian closed (since currying does not preserve strictness). Virtually all of the examples that use **Cont** can be applied to any cartesian closed category of cpo's and continuous functions, and most can be applied to any set-like category.

3. Monads

Definition 1 A *monad* on a category $\mathcal{C}$, $\langle T, \eta, \mu\rangle$ is a functor, $T : \mathcal{C} \to \mathcal{C}$, with two natural transformations, $\eta : I \xrightarrow{\bullet} T$ and $\mu : T^2 \xrightarrow{\bullet} T$, satisfying the following conditions for all objects A of $\mathcal{C}$:

- $\mu_A \circ \mu_{TA} = \mu_A \circ T\mu_A$
- $\mu_A \circ \eta_{TA} = \mu_A \circ T\eta_A = \mathsf{id}_{TA}$

An alternative form is a *Kleisli triple*, $\langle T, \eta, _^*\rangle$, where T is a function from objects to objects, for each object A, η_A is a morphism from A to TA, and for all morphisms $f : A \to TB$, f^* is a morphism from TA to TB, all satisfying the three conditions below:

- $f^* \circ \eta_A = f$
- $\eta_A^* = \mathsf{id}_{TA}$
- $(g^* \circ f)^* = g^* \circ f^*$.

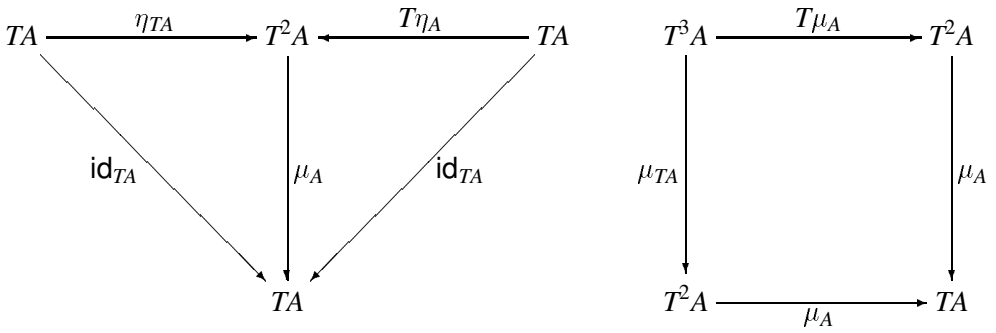

Figure 1: Identity and Associativity laws for a monad

The two forms are equivalent; for any a monad $\langle T, \eta, \mu \rangle$, there is a Kleisli triple $\langle T, \eta, _^* \rangle$ where T is the restriction of the functor T to objects, η is unchanged, and for all $f : A \to TB$, $f^* = \mu_B \circ Tf$. Also for all Kleisli triples $\langle T, \eta, _^* \rangle$, there is a monad $\langle T, \eta, \mu \rangle$, where $Tf = (\eta \circ f)^*$, and $\mu = \mathsf{id}^*_{TA}$.

3.1. Identity monad

An obvious monad is the identity monad $\langle I, \mathsf{id}, \mathsf{id} \rangle$, where I is the identity functor, and the natural transformations are identity morphisms.

3.2. Lifting

In the category **Cont**, there is a monad $\langle L, \mathsf{up}, \mathsf{down} \rangle$ where

- For all objects A, $LA = A \cup \{\bot_{LA}\}$, where $\bot_{LA} \notin A$ is a new least element,
- For all morphisms (continuous functions) $f : A \to B$, $Lf(x) = \begin{cases} \bot_{LB} & x = \bot_{LA} \\ f(x) & \text{otherwise} \end{cases}$
- For all objects A, $\mathsf{up}_A : A \to LA$ is the inclusion function
- For all objects A, $\mathsf{down}_A : L^2A \to LA$ is $\mathsf{down}_A(x) = \begin{cases} \bot_{LA} & x = \bot_{L^2A} \\ x & \text{otherwise} \end{cases}$

Lifting can be used (see [14]) to model partiality in programming languages, with the new bottom element representing divergence.

3.3. Coproducts, Disjoint Sums, and Separated Sums

In any category $\mathcal{C}$ with finite coproducts, and a distinguished object E, there is a monad $\langle E, \eta, \mu \rangle$ such that

- For all objects A, $EA = A + E$.

- For all $f : A \to B$, $Ef : A + E \to B + E$ is $f + \mathsf{id}_E$
- For all objects A, $\eta_A : A \to A + B$ is ι_1, the corresponding coproduct injection morphism.
- For all objects A, $\mu_A : (A + E) + E \to A + E$ is $[\mathsf{id}_{EA}, \iota_2]$.

This construction also defines a monad when using separated sums in **Cont**, even though they do not satisfy the requirements for a coproduct (the [–,–] constructor is not unique for separated sums).

Coproducts are typically (see [14] and [18]) used to model exceptional handling, where E represent a set of errors or exceptions.

As an example, in **Cont**, let E be the singleton set $\{\mathtt{err}\}$. Then the monad $\langle E, \eta, \mu \rangle$ looks like

- For all domains A, $EA = \{\langle 1, a\rangle \mid a \in A\} \cup \{\langle 2, \mathtt{err}\rangle\} \cup \{\bot\}$ with the standard separated sum ordering
- For all continuous functions $f : A \to B$

$$Ef\,(x) = \begin{cases} \langle 1, f\,(a)\rangle & x = \langle 1, a\rangle \\ \langle 2, \mathtt{err}\rangle & x = \langle 2, \mathtt{err}\rangle \\ \bot & x = \bot \end{cases}$$

- For all domains A, $\eta_A(a) = \langle 1, a\rangle$
- For all domains A, $\mu_A : (A + E) + E \to (A + E)$ is defined as

$$\mu_A(x) = \begin{cases} y & x = \langle 1, y\rangle \\ \langle 2, \mathtt{err}\rangle & x = \langle 2, \mathtt{err}\rangle \\ \bot & x = \bot \end{cases}$$

If we look at separated sums and lifting on *predomains* (domains that are not necessarily pointed), then lifting is a specific kind of separated sum, namely $LA = A + \{\}$. In particular any property that is true of all separated sums is also true of all lifted domains, and if a property fails to hold for lifted domains, it also fails to hold for separated sums.

3.4. Products

Let $\mathcal{C}$ be any category with binary products and a terminal object. Then an object X is a monoid in $\mathcal{C}$, if there exist morphisms $e : \mathbf{1} \to X$ and $m : X \times X \to X$ such that

- $(m \circ \mathsf{id}_X) \times e = \pi_1$,
- $m \circ (e \times \mathsf{id}_X) = \pi_2$, and
- $m \circ (m \times \mathsf{id}_X) \circ \alpha_{X,X,X} = m \circ \mathsf{id}_X \times m$, where $\alpha_{A,B,C}$ is the natural isomorphism from $A \times (B \times C)$ to $(A \times B) \times C$.

Given a monoid X, there is then a monad $\langle X, \eta, \mu \rangle$ defined as

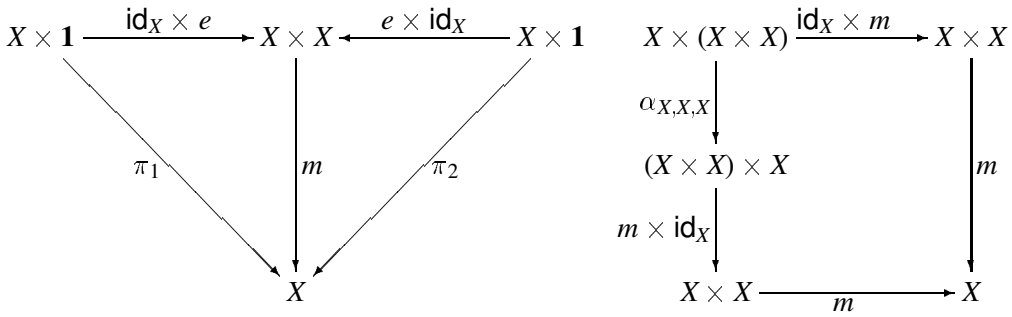

Figure 2: Identity and Associativity properties of a monoid

- For all objects A, $XA = A \times X$.
- For all $f : A \to B$, $Xf = f \times \mathsf{id}_X$.
- For all objects A, $\eta_A : A \to A \times X$ is $\langle \mathsf{id}_A, e \circ !_A \rangle$.
- For all objects A, $\mu_A : (A \times X) \times X \to A \times X$ is $(\mathsf{id}_A \times m) \circ \alpha^{-1}_{A,X,X}$, where α^{-1} is the natural isomorphism from $(_ \times _) \times _$ to $_ \times (_ \times _)$.

Products can be used to model a simple form of output processing ([18]) or to calculate resources (such as time or space) used in a program ([11]).

As an example, let $\mathcal{C}$ = **Cont** and let X = **VNat**, the set of natural numbers (plus ω to make it directed complete), ordered vertically with least element 0 and greatest element ω. Let e be the constant 0 function, let m be addition. Then **VNat** is a monoid, and the monad $\langle X, \eta, \mu \rangle$ looks like

- For all domains A, $XA = A \times$ **VNat**
- For all functions $f : A \to B$, $Xf \langle a, n \rangle = \langle f(a), n \rangle$.
- For all domains A, $\eta_A(a) = \langle a, 0 \rangle$
- For all domains A, $\mu_A \langle \langle a, n_1 \rangle, n_2 \rangle = \langle a, n_1 + n_2 \rangle$.

3.5. Power Sets and Power Domains

In the category **Set** the power set constructor forms a monad $\langle \mathcal{P}, \{-\}, \bigcup \rangle$ where

- For all sets A, $\mathcal{P}A$ is the power set of A,
- For all functions $f : A \to B$ and all $X \in \mathcal{P}A$, $\mathcal{P}f(X) = \{f(x) \mid x \in X\}$.
- $\{-\} : I \xrightarrow{\bullet} P$ forms singleton sets from elements
- $\bigcup : P^2 \xrightarrow{\bullet} P$ is set union

Thus we can model nondeterminism ([14]) by having the result of a program be a set of possible final values.

To do the same in **Cont**, however, we need to form domains out of power sets, in particular we need a partial order on the sets. Since we generally want to make use of the ordering in the underlying set, using set inclusion as the partial order is insufficient. There are three orderings typically used for power set domains (see [9]):

Hoare or lower ordering: $u \sqsubseteq^\flat v$ if $\forall x \in u, \exists y \in v$ such that $x \leq y$.

Smyth or upper ordering: $u \sqsubseteq^\sharp v$ if $\forall y \in v, \exists x \in u$ such that $x \leq y$.

Plotkin or convex ordering: $u \sqsubseteq^\natural v$ if $u \sqsubseteq^\flat v$ and $u \sqsubseteq^\sharp v$.

These orderings tend to form *preorders* instead of partial orders but, by a construction taken from [9], we can get back a partial order.

Definition 2 An *ideal* over a preorder A is a nonempty set $u \subseteq A$ that is directed and downwards closed. An ideal u is a *principal ideal* if $u = \downarrow a \stackrel{\text{def}}{=} \{b \mid b \leq a\}$ for some $a \in A$. $\mathbf{Idl}(A)$ is the poset of all ideals in A, ordered by inclusion.

Theorem 1 [9] *For all preorders A, $\mathbf{Idl}(A)$ is an algebraic cpo, with compact elements being the principal ideals.*

Thus for any algebraic cpo A we can form another algebraic cpo, the power domain

$$P^\dagger A = \mathbf{Idl}(\langle \mathcal{P}^*_{\text{fin}}(K(A)), \sqsubseteq^\dagger \rangle),$$

where $\dagger \in \{\flat, \sharp, \natural\}$ refers to any one of the three preorderings on powersets and $\mathcal{P}^*_{\text{fin}}(X)$ is the set of finite, non-empty subsets of X. Thus an element $s \in P^\dagger A$ is a (directed, downwards closed) collection of finite sets of compact elements of A.

Although theorem 1 gives us an algebraic cpo, what we are looking for is a Scott domain. If A is Scott domain, then so are the Smyth and Hoare power domains $P^\sharp A$ and $P^\flat A$. The Plotkin power domain $P^\natural A$, however, may not even be bounded-complete (for example, consider $P^\natural(\mathbf{Bool} \times \mathbf{Bool})$, where **Bool** is the standard flat truth value domain $\{\mathtt{tt}, \mathtt{ff}, \bot\}$). While the Plotkin power domain of a Scott domain is not necessarily a Scott domain, the Plotkin power domain of a bifinite domain does remain bifinite (see [9]).

All three power domains can be used to form monads $\langle P^\dagger, \{\!|-|\!\}^\dagger, \biguplus^\dagger \rangle$ as follows:

- For all domains A, and $a \in A$, Let $\{\!|-|\!\}^\dagger_A : A \to P^\dagger A$ be the function

$$\{\!|\, a \,|\!\}^\dagger_A = \{w \in \mathcal{P}^*_{\text{fin}}(K(A)) \mid \exists k \in K(A).k \leq^A a \text{ and } w \sqsubseteq^\dagger \{k\}\}$$

- For all functions $f : A \to B$, let $P^\dagger f : P^\dagger A \to P^\dagger B$ be defined as

$$P^\dagger f(s) = \bigcup_{\{a_1,\ldots,a_n\} \in s} \{\!|f(a_1)|\!\}^\dagger_B \uplus^\dagger_B \ldots \uplus^\dagger_B \{\!|f(a_n)|\!\}^\dagger_B$$

where for each domain A, $\uplus_A^\dagger$ is the binary (associative) function from $P^\dagger A \times P^\dagger A$ to $P^\dagger A$ defined by

$$s \uplus_A^\dagger t = \{w \in \mathcal{P}^*_{\text{fin}}(K(A)) \mid \exists u \in s.\exists v \in t.\, w \sqsubseteq^\dagger u \cup v\}$$

Combining the above definitions gives us

$$P^\dagger f\,(s) = \bigcup_{u \in s} \{w \in \mathcal{P}^*_{\text{fin}}(K(B)) \mid \forall a \in u.\exists k_a \in K(B).k_a \leq^B f\,(a) \text{ and } w \sqsubseteq^\dagger \{k_a \mid a \in u\}\}$$

- Given a domain A, $\biguplus_A^\dagger : P^\dagger(P^\dagger A) \to P^\dagger A$ is defined by

$$\biguplus^\dagger{}_A(\overline{s}) = \bigcup\{\bigcup\{u \mid \downarrow u \in \overline{u}\} \mid \overline{u} \in \overline{s}\}$$

If we only look at compact elements and if we have an $f : A \to B$ that preserves compactness, then the functions above simplify to

- $\{\!|\, k \,|\!\}_A^\dagger = \{w \in \mathcal{P}^*_{\text{fin}}(K(A)) \mid w \sqsubseteq^\dagger \{k\}\} = \downarrow\{k\}$.
- $(\downarrow u) \uplus_A^\dagger (\downarrow v) = \downarrow(u \cup v)$
- $P^\dagger f\,(\downarrow\{a_1, \ldots a_n\}) = \downarrow\{f\,(a_1)\} \uplus_B^\dagger \ldots \uplus_B^\dagger \downarrow\{f\,(a_1)\} = \downarrow\{f\,(a_1), \ldots f\,(a_n)\}$, i.e. $P^\dagger f\,(\downarrow u) = \downarrow \mathcal{P} f\,(u)$.
- $\biguplus_A^\dagger(\downarrow\overline{u}) = \bigcup_{\downarrow u \in \overline{u}} u$.

3.6. Exponentiation

Let $\mathcal{C}$ be a cartesian closed category and let V be any object of $\mathcal{C}$. Then there is a monad $\langle V^{\rightarrow}, \mathsf{const}, \mathsf{diag}\rangle$ where

- For all objects A, $V^{\rightarrow}A = [V \to A]$
- For all $f : A \to B$, $V^{\rightarrow}f : [V \to A] \to [V \to B]$ is $\mathsf{curry}(f \circ \mathsf{app}_{V,A})$
- For all objects A, $\mathsf{const}_A : A \to [V \to A]$ is $\mathsf{curry}(\pi_1)$
- For all objects A, $\mathsf{diag}_A : [V \to [V \to A]] \to [V \to A]$ is $\mathsf{curry}(\mathsf{app}_{V,A} \circ \langle \mathsf{app}_{V,[V\to A]}, \pi_2\rangle)$

In **Cont** the monad becomes

- For all domains A, $V^{\rightarrow}A$ is the domain of all continuous functions from V to A.
- For all continuous functions $f : A \to B$, $V^{\rightarrow}f\,(\lambda v.a_v) = \lambda v.f\,(a_v)$.
- For all domains A, $\mathsf{const}_A(a) = \lambda v.a$.
- For all domains A, $\mathsf{diag}_A(\lambda v_1.\lambda v_2.a_{v_1,v_2}) = \lambda v.a_{v,v}$

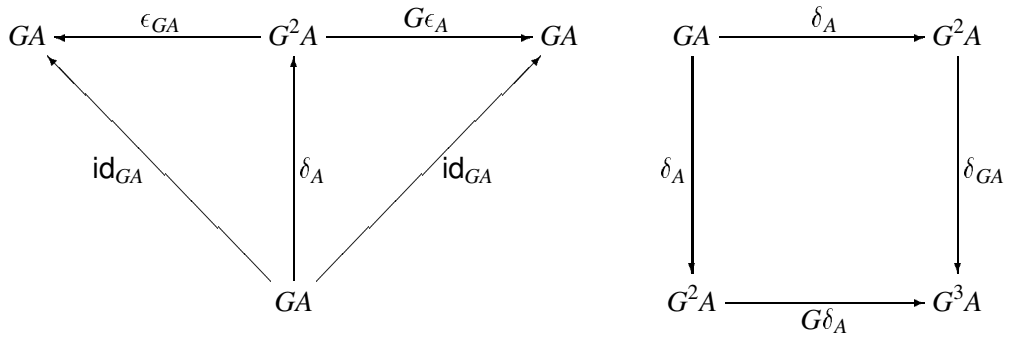

Figure 3: Conditions for a comonad

4. Comonads

A comonad in $\mathcal{C}$ is a monad in the opposite category $\mathcal{C}^{\mathrm{op}}$, namely, it is a functor $G : \mathcal{C} \to \mathcal{C}$, with two natural transformations $\epsilon : G \stackrel{\bullet}{\to} I$ and $\delta : G \stackrel{\bullet}{\to} G^2$ such that for all objects A of $\mathcal{C}$

- $\epsilon_{GA} \circ \delta_A = G\epsilon_A \circ \delta_A = \mathsf{id}_{GA}$
- $\delta_{GA} \circ \delta_A = G\delta_A \circ \delta_A$

A comonad also has a form similar to the Kleisli triple: the *Kleisli cotriple* $\langle G, \epsilon, _^+ \rangle$, where G is a function on objects, and for each object A, ϵ_A is a morphism from GA to A, and for each morphism $f : GA \to B$, f^+ is a morphism from GA to GB, all satisfying the following conditions:

- For all $f : GA \to B$, $\epsilon_B \circ f^+ = f$
- For all objects A, $\epsilon_A^+ = \mathsf{id}_{GA}$
- For all $f : GA \to B$ and $g : GB \to C$, $g^+ \circ f^+ = (g \circ f^+)^+$.

4.1. Side effects

Let $\mathcal{C}$ be any cartesian closed category and let S be an object in $\mathcal{C}$. Then there is a monad $\langle S, \eta, \mu \rangle$ where

- For all objects A, $SA = [S \to (A \times S)]$
- For all morphisms $f : A \to B$, $Sf : [S \to (A \times S)] \to [S \to (B \times S)]$ is $\mathsf{curry}((f \times \mathsf{id}_S) \circ \mathsf{app}_{S,A\times S})$
- For all objects A, $\eta_A : A \to [S \to (A \times S)]$ is $\mathsf{curry}(\mathsf{id}_{A\times S})$
- For all objects A, $\mu_A : [S \to ([S \to (A \times S)] \times S)] \to [S \to (A \times S)]$ is $\mathsf{curry}(\mathsf{app}_{S,A\times S} \circ \mathsf{app}_{S,SA\times S})$

In **Cont** the monad becomes:

- For all continuous functions $f : A \to B$, $Sf(\lambda s.\langle a_s, z_s\rangle) = \lambda s.\langle f(a_s), z_s\rangle$
- For all domains A, $\eta_A(a) = \lambda s.\langle a, s\rangle$.
- For all domains A, $\mu_A(\lambda s.\langle u_s, z_s\rangle) = \lambda s.u_s(z_s)$

This monad can be used to model side effects (see [14]), where S represents some internal state; a computation takes an initial state and returns a value plus a new state.

4.2. Product Comonad

Let $\mathcal{C}$ be a category with binary products and let X be any object of $\mathcal{C}$. Then there is a comonad $\langle X, \pi_1, \delta\rangle$ where

- For all objects A, $XA = A \times X$
- For all $f : A \to B$, $Xf = f \times \mathsf{id}_X$.
- For all A, $\pi_1 : A \times X \to A$ is the corresponding product projection morphism.
- For all objects A, $\delta_A : A \times X \to (A \times X) \times X$ is $\langle \mathsf{id}_{A\times X}, \pi_2\rangle$; in **Cont**, $\delta_A\langle a, x\rangle = \langle\langle a, x\rangle, x\rangle$.

4.2.1. Computation paths: Exponentiation comonad

Let $\mathcal{C}$ be a cartesian closed category, and let V be a monoid in $\mathcal{C}$ with identity element $e : 1 \to V$ and binary operator $m : V \times V \to V$. Then there is a comonad $\langle V, \mathsf{val}, \mathsf{pre}\rangle$ where

- For all objects A, $VA = [V \to A]$
- For all $f : A \to B$, $Vf : [V \to A] \to [V \to B]$ is $\mathsf{curry}(f \circ \mathsf{app}_{VA,V})$
- For all objects A, $\mathsf{val}_A : [V \to A] \to A$ is $\mathsf{app}_{VA,V} \circ \langle \mathsf{id}_{VA}, e \circ !_{VA}\rangle$
- For all objects A, $\mathsf{pre}_A : [V \to A] \to [V \to [V \to A]]$ is $\mathsf{curry}(\mathsf{curry}(\mathsf{app}_{VA,V} \circ (\mathsf{id}_A \times m) \circ \alpha^{-1}_{VA,V,V}))$, where $\alpha^{-1} : (_ \times _) \times _ \xrightarrow{\bullet} _ \times (_ \times _)$

To see the above more clearly, in **Cont** it becomes

- For all objects A, VA is the domain of continuous functions from V to A
- For all continuous functions $f : A \to B$, $Vf(\lambda v.a_v) = \lambda v.f(a_v)$.
- For all objects A, $\mathsf{val}_A(\lambda v.a_v) = a_e$.
- For all objects A, $\mathsf{pre}_A(\lambda v.a_v) = \lambda v_1.\lambda v_2.a_{m(v_1,v_2)}$.

One possible monoid in **Cont** has V = **VNat**, $e = \omega$, and m the greatest lower bound function. With this monoid the monad becomes the path comonad mentioned in [7], where VA can be interpreted as a non-decreasing sequence of elements of A, Vf maps f onto the elements of the sequence, val takes the value of a sequence at ω, which by continuity is also the least upper bound of the elements of the sequence, and pre returns the sequence of prefixes of a sequence.

4.2.2. Strictly increasing paths

When using the path comonad mentioned above, where V = **VNat**, etc., elements of VA can be thought of as sequences of construction steps used to build a data value. Since the sequence is not necessarily strictly increasing, there may be places in the sequence where the values remain unchanged. To get sequences where the values always increase, we create a variant of the comonad $\langle V, \mathsf{val}, \mathsf{pre}\rangle$ in **Cont** using strictly increasing paths. The resulting comonad $\langle V_S, \mathsf{sval}, \mathsf{spre}\rangle$ is defined as follows:

- For all domains A, let V_SA be the set of finite or infinite strictly increasing sequences in A. The finite sequences are represented as eventually constant (infinite) sequences. Therefore we have either sequences of the form $\langle a_n\rangle_{n=0}^{\omega}$, where for each n, $a_n <_A a_{n+1}$, or sequences of the form $a_0a_1 \ldots a_{N-1}a_N^{\omega}$ where $a_0 <_A a_1 <_A \ldots <_A a_N$. The ordering $\leq_{V_SA}$ is the least partial ordering such that

  $$a_0 \ldots a_{N-1}a_N^{\omega} \leq_{V_SA} a_0 \ldots a_{N-1}\mathbf{a}$$

  whenever $\mathbf{a} \in V_SA$ and $a_N \leq_A \mathbf{a}_0$. If we consider a sequence to be a (stable) function from **VNat** to A, then this ordering of sequences is closely related to the stable ordering on functions.

- For all continuous functions $f : A \to A'$, let V_Sf be the least continuous function such that for all $a, a_0, a_1 \in A$ such that $a_0 <_A a_1$ and for all $\mathbf{a} \in V_SA$,

  $$\begin{aligned} V_Sf\,(a^{\omega}) &= (f\,(a))^{\omega} \\ V_Sf\,(a_0a_1\mathbf{a}) &= \begin{cases} (f\,(a_0))(V_Sf\,(a_1\mathbf{a})) & f\,(a_0) \neq f\,(a_1) \\ V_Sf\,(a_1\mathbf{a}) & f\,(a_0) = f\,(a_1) \end{cases} \end{aligned}$$

  The definition is similar to the definition of V on functions except that any resulting duplications are removed.

- For all domains A, $\mathsf{sval}_A(\mathbf{a}) = \mathbf{a}_{\omega} = \sqcup_{n=0}^{\infty}\mathbf{a}_n$.

- For all domains A, $\mathsf{spre}_A(\mathbf{a}) = \left\langle\left\langle \mathbf{a}_{\min(i,j)}\right\rangle_{j=0}^{\omega}\right\rangle_{i=0}^{\omega}$ or equivalently, $\mathsf{spre}_A(\mathbf{a}) = \left\langle \mathbf{a}_0 \ldots \mathbf{a}_{i-1}\mathbf{a}_i^{\omega}\right\rangle_{i=0}^{\omega}$.

5. Kleisli Categories

Given a category $\mathcal{C}$ and a monad $\langle T, \eta, \mu\rangle$ there is a category $\mathcal{K}(T)$, the *Kleisli category* of T, defined as follows:

- The objects in $\mathcal{K}(T)$ are the objects in $\mathcal{C}$

- A morphism $f : A \xrightarrow{T} B$ in $\mathcal{K}(T)$ is a morphism $f : A \to TB$ in $\mathcal{C}$

- For all objects A in $\mathcal{K}(T)$, the identity morphism on A is $\eta_A : A \to TA$.
- For all pairs of morphisms $f : A \xrightarrow{T} B$ and $g : B \xrightarrow{T} C$, their composition is $g \overset{T}{\circ} f = \mu_C \circ Tg \circ f$.

Similarly, for a comonad $\langle G, \epsilon, \delta \rangle$, there is a Kleisli category $\mathcal{K}(G)$, where

- The objects in $\mathcal{K}(G)$ are the objects in $\mathcal{C}$
- A morphism $f : A \xrightarrow{G} B$ in $\mathcal{K}(G)$ is a morphism $f : GA \to B$ in $\mathcal{C}$
- For all objects A in $\mathcal{K}(G)$, its identity arrow is $\epsilon_A : GA \to A$.
- For all pairs of morphisms $f : A \xrightarrow{G} B$ and $g : B \xrightarrow{G} C$, their composition is $g \overset{G}{\circ} f = g \circ Gf \circ \delta_A$.

[7] and [14] (and section 6.1) show how to obtain semantic interpretations of languages using Kleisli categories. In particular the meaning of a functional program becomes a morphism in the Kleisli category, and the Kleisli category composition rules are used instead of composition in the original category.

6. Double Kleisli Categories

The Kleisli category of a monad is generally used to represent a semantic interpretation of a functional language with added non-functional features. The Kleisli category of a comonad is generally used to look at intensional semantic interpretations. In order to combine the two, we would like to form from a monad $\langle T, \eta, \mu \rangle$ and a comonad $\langle G, \epsilon, \delta \rangle$ in $\mathcal{C}$ a category $\mathcal{K}(G, T)$ similar to the Kleisli categories, with objects in $\mathcal{C}$, and with morphisms of the form $f : GA \to TB$.

In order to guarantee that we actually obtain a category, however, we need some additional conditions, adapted from [4]:

Definition 3 Given a monad $\langle T, \eta, \mu \rangle$ and a comonad $\langle G, \epsilon, \delta \rangle$, a *distributive law of T over G* is a natural transformation $\sigma : GT \xrightarrow{\bullet} TG$ such that the following four identities hold:

- $\sigma_A \circ G\eta_A = \eta_{GA}$
- $T\epsilon_A \circ \sigma_A = \epsilon_{TA}$
- $\sigma_A \circ G\mu_A = \mu_{GA} \circ T\sigma_A \circ \sigma_{TA}$
- $T\delta_A \circ \sigma_A = \sigma_{GA} \circ G\sigma_A \circ \delta_{TA}$

σ is *distributive with* η whenever the first identity holds. Similarly σ is *distributive with* ϵ when the second identity holds, and so on.

For similar definitions, see [4], which related two monads, or [1], which related two comonads.

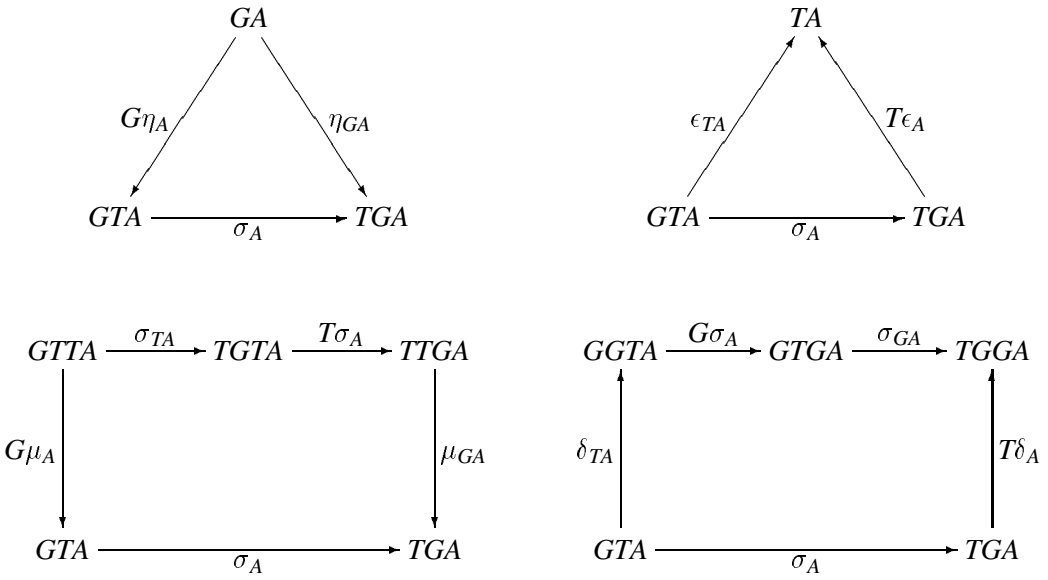

Figure 4: Distributive laws

Theorem 2 *Given a monad $\langle T, \eta, \mu\rangle$ and a comonad $\langle G, \epsilon, \delta\rangle$ in a category $\mathcal{C}$ plus a distributive law σ of T over G, there is a category $\mathcal{K}(G,T)$ defined as follows:*

- *The objects of $\mathcal{K}(G,T)$ are the objects of $\mathcal{C}$.*
- *A morphism $f : A \overset{G,T}{\to} B$ in $\mathcal{K}(G,T)$ is a morphism $f : GA \to TB$ in $\mathcal{C}$.*
- *For each object A, the identity morphism $\mathsf{id}_A^{G,T}$ on A is $\eta_A \circ \epsilon_A : GA \to TA$.*
- *For every pair of morphisms $f : A \overset{G,T}{\to} B$ and $g : B \overset{G,T}{\to} C$, their composition, $g \overset{G,T}{\circ} f$, is $\mu_C \circ Tg \circ \sigma_B \circ Gf \circ \delta_A$.*

Proof. The proof involves some straightforward diagram chasing, shown in figures 5 and 6. Figure 5 contains a diagram showing that $\mathsf{id}_B^{G,T} \circ f = f \circ \mathsf{id}_A^{G,T} = f$. The upper part of the diagram represents the expansion of $\mathsf{id}_B^{G,T} \circ f$, $\mu_B \circ T\eta_B \circ T\epsilon_B \circ \sigma_B \circ Gf \circ \delta_A$, and the lower part of the diagram represents the expansion of $f \circ \mathsf{id}_A^{G,T}$, $\mu_B \circ Tf \circ \sigma_A \circ G\eta_A \circ T\epsilon_A \circ \delta_A$; in the center is f.

Figure 6 contains a diagram showing that $h \overset{G,T}{\circ} (g \overset{G,T}{\circ} f) = (h \overset{G,T}{\circ} g) \overset{G,T}{\circ} f$, The upper path represents the expansion of $h \overset{G,T}{\circ} (g \overset{G,T}{\circ} f)$ and the lower path represents the expansion of $(h \overset{G,T}{\circ} g) \overset{G,T}{\circ} f$. □

6.1. Examples

We now check all the possible combinations of comonads and monads from our lists of examples to see whether the distributivity condition holds. A summary of the results is in table 1 on page 26.

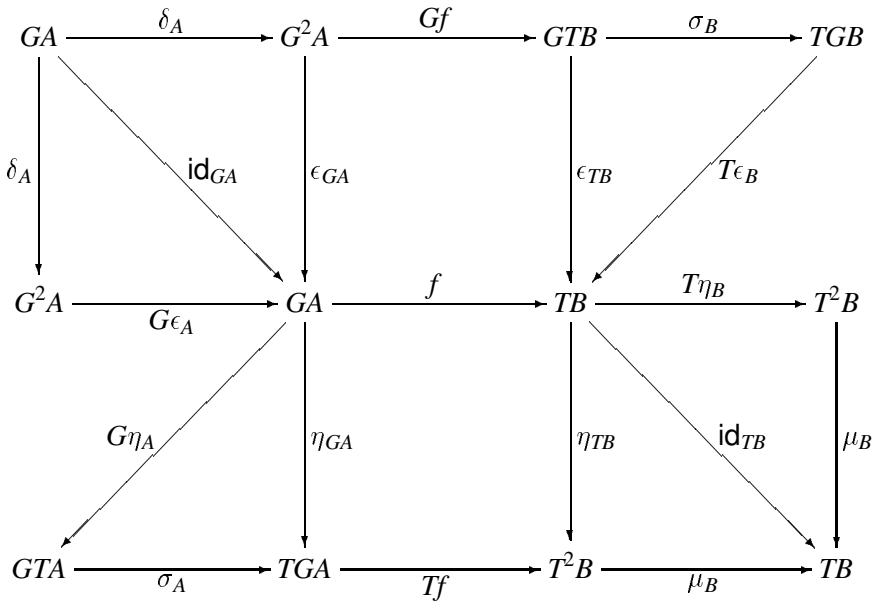

Figure 5: Proof of the properties of identity in $\mathcal{K}(G,T)$.

6.1.1. The Product comonad

A distributive law σ of any monad T over the product comonad X is a natural transformation from $T_ \times X$ to $T(_ \times X)$. For certain types of monads such a distributive law is guaranteed to exist:

Definition 4 Let $\mathcal{C}$ be any category with finite products. A monad $\langle T, \eta, \mu \rangle$ is *strong* if there exists a natural transformation $\tau : T_ \times _ \stackrel{\bullet}{\to} T(_ \times _)$, called a *tensorial strength*, satisfying the following conditions:

- $T\pi_1 \circ \tau_{A,\mathbf{1}} = \pi_1$.
- $\tau_{A,B} \circ (\eta_A \times \mathsf{id}_B) = \eta_{A\times B}$.
- $\mu_{A\times B} \circ T\tau_{A,B} \circ \tau_{TA,B} = \tau_{A,B} \circ (\mu_A \times \mathsf{id}_B)$.
- $\tau_{A\times B,C} \circ (\tau_{A,B} \times \mathsf{id}_C) \circ \alpha_{TA,B,C} = T\alpha_{A,B,C} \circ \tau_{A,B\times C}$, where again $\alpha : _ \times (_ \times _) \stackrel{\bullet}{\to} (_ \times _) \times _$

This definition of strength is equivalent to the definition in [14], which uses a natural transformation $t : _ \times T_ \stackrel{\bullet}{\to} T(_ \times _)$. It is clear that given such a t we can get τ with $\tau_{A,B} = T\langle \pi_2, \pi_1 \rangle \circ t_{B,A} \circ \langle \pi_2, \pi_1 \rangle$ and similarly we can get t from τ.

Unsurprisingly, if $\langle T, \eta, \mu \rangle$ is strong then there is a distributive law σ of T over the product comonad X, with $\sigma_A = \tau_{A,X}$. Of course, determining the existence of strength is in general as difficult as determining the existence of a distributive law directly. For set-like categories, however, there is a simple test for strength:

Theorem 3 *Let $\mathcal{C}$ be a set-like category, and let $\langle T, \eta, \mu \rangle$ be a monad over $\mathcal{C}$ such that for each pair of objects A and B the function $\mathbf{t}_{A,B} = \lambda f\,.Tf : [A \to B] \to [TA \to TB]$ is a morphism in $\mathcal{C}$. Then $\langle T, \eta, \mu \rangle$ is strong.*

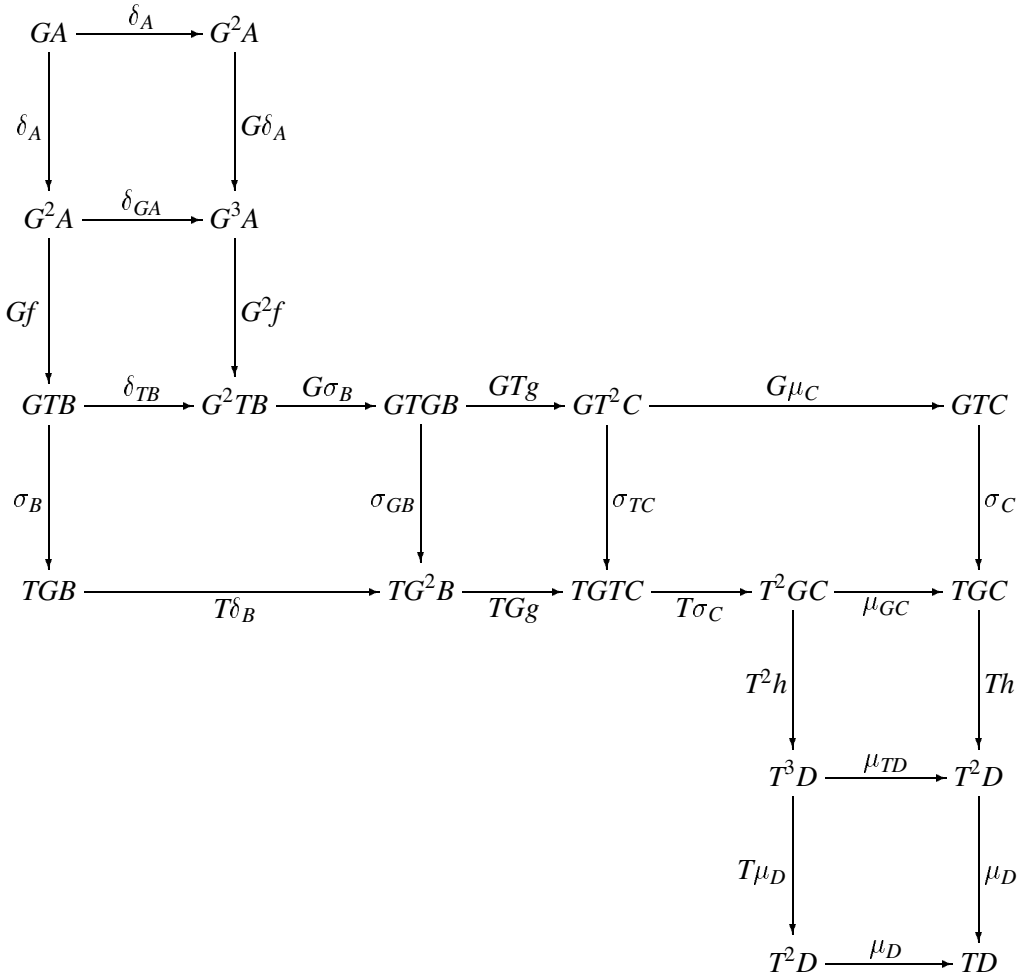

Figure 6: Proof of associativity of composition in $\mathcal{K}(G,T)$

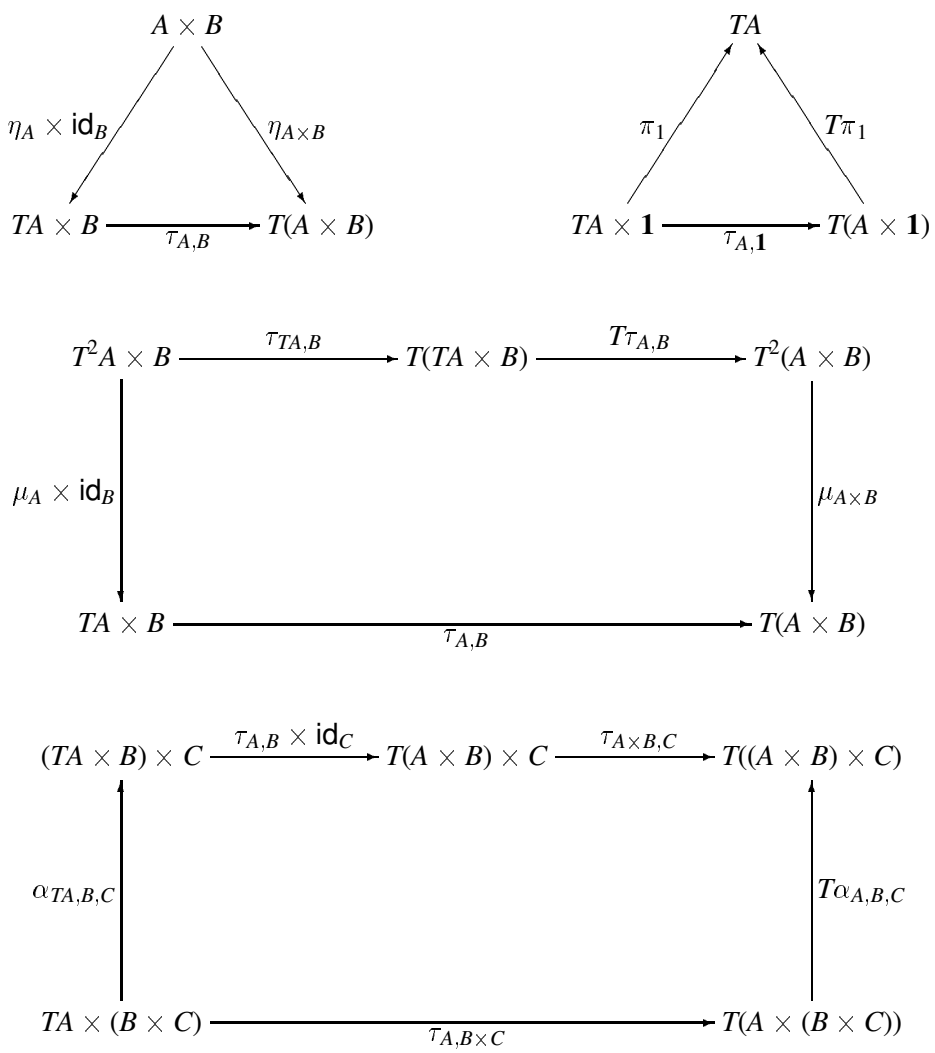

Figure 7: Requirements for τ to be a tensorial strength

Proof. Given such a category, let $\tau_{A,B} = \mathsf{uncurry}(\mathbf{t}_{A,A\times B} \circ \mathsf{curry}(\langle \pi_2, \pi_1 \rangle)) \circ \langle \pi_2, \pi_1 \rangle = \lambda\langle \mathbf{a}, b\rangle.T(\lambda a.\langle a, b\rangle)\mathbf{a}$. For naturality, given a function $f : A \to B$ and a function $g : C \to D$, we have that

$$\begin{aligned}
\tau_{C,D} \circ (Tf \times g)\langle \mathbf{a}, c\rangle &= \tau_{C,D}\langle Tf\,(\mathbf{a}), g(c)\rangle \\
&= T(\lambda b.\langle b, g(c)\rangle)Tf\,(\mathbf{a}) \\
&= T(\lambda b.\langle b, g(c)\rangle \circ \lambda a.f\,(a))\mathbf{a} \\
&= T(\lambda a.\langle f\,(a), g(c)\rangle)\mathbf{a} \\
&= T((f \times g) \circ \lambda a.\langle a, c\rangle)\mathbf{a} \\
&= T(f \times g) \circ \tau_{A,B}\langle \mathbf{a}, c\rangle
\end{aligned}$$

The other conditions can be shown in a similarly straightforward manner. □

All of the monad examples given are strong in **Cont** [14]. The resulting σ turn out to be:

Lifting: $\sigma_A : LA \times X \to L(A \times X)$ is defined by $\sigma_A\langle \perp, x\rangle = \perp$ and for $a \in A$, $\sigma_A\langle a, x\rangle = \langle a, x\rangle$.

Separated Sums: $\sigma_A : (A+E) \times X \to (A \times X)+E$ is defined by $\sigma_A\langle \iota_1(a), x\rangle = \iota_1(\langle a, x\rangle)$, $\sigma_A\langle \iota_2(e), x\rangle = \iota_2(e)$, and $\sigma_A\langle \perp, x\rangle = \perp$.

Products: $\sigma_A : (A \times X') \times X \to (A \times X) \times X'$ is defined by $\sigma_A\langle\langle a, x'\rangle, x\rangle = \langle\langle a, x\rangle, x'\rangle$, where X denotes the object from the product comonad, and X' denotes the object from the product monad.

The product monad is strong in any category with products, where $\sigma_A = \langle \pi_1 \times \mathsf{id}_X, \pi_2 \circ \pi_1\rangle$ is the natural isomorphism from $(A \times X') \times X$ to $(A \times X) \times X'$.

Side effects: $\sigma_A : [S \to (A \times S)] \times X \to [S \to ((A \times X) \times S)]$ is defined by $\sigma_A\langle \lambda s.\langle a_s, z_s\rangle, x\rangle = \lambda s.\langle\langle a_s, x\rangle, z_s\rangle$.

More generally, in a cartesian closed category we have $\sigma_A = \mathsf{curry}(\beta_{A,S,X} \circ (\mathsf{app}_{SA,S} \times \mathsf{id}_X) \circ \beta_{SA,X,S})$, where $\beta_{A,B,C}$ is the natural isomorphism from $(A \times B) \times C$ to $(A \times C) \times B$.

Power Sets: (In **Set**) $\sigma_A : \mathcal{P}A \times X \to \mathcal{P}(A \times X)$ is defined by $\sigma_A\langle u, x\rangle = \{\langle a, x\rangle \mid x \in u\}$.

Power Domains: $\sigma^\dagger_A : P^\dagger A \times X \to P^\dagger(A \times X)$ (for $\dagger \in \{\flat, \sharp, \natural\}$) is defined by

$$\sigma^\dagger_A\langle s, x\rangle = \bigcup_{u \in s}\{w \mid \forall a \in u.\exists k_a \leq \langle a, x\rangle.w \sqsubseteq^\dagger \{k_a \mid a \in u\}\}$$

For compact elements $\langle \downarrow u, k\rangle$, $\sigma^\dagger_A\langle \downarrow u, k\rangle = \downarrow\{\langle a, k\rangle \mid a \in u\}$.

Exponentiation: $\sigma_A : [V \to A] \times X \to [V \to (A \times X)]$ is defined by $\sigma_A\langle \lambda v.a_v, x\rangle = \lambda v.\langle a_v, x\rangle$. A similar definition holds for strict paths.

For all cartesian closed categories the exponentiation monad is strong, with

$$\sigma_A = \mathsf{curry}((\mathsf{app}_{A,V} \times \mathsf{id}_X) \circ \beta_{VA,X,V})$$

6.1.2. Exponentiation as a monad

A comonad $\langle G, \epsilon, \delta\rangle$ is strong if there exists a natural transformation $\tau : G_- \times _ \xrightarrow{\bullet} G(_ \times _)$ that satisfies analogous conditions to the ones required for a strong monad (see figure 8). For all such comonads, there exists a distributive law σ of the exponentiation monad $V^{\rightarrowtail}$ over G, where $\sigma_A = \mathsf{curry}(G\mathsf{app}_{X,A} \circ \tau_{V \rightarrowtail A,X})$. The proof is analogous to the proof for the product monad.

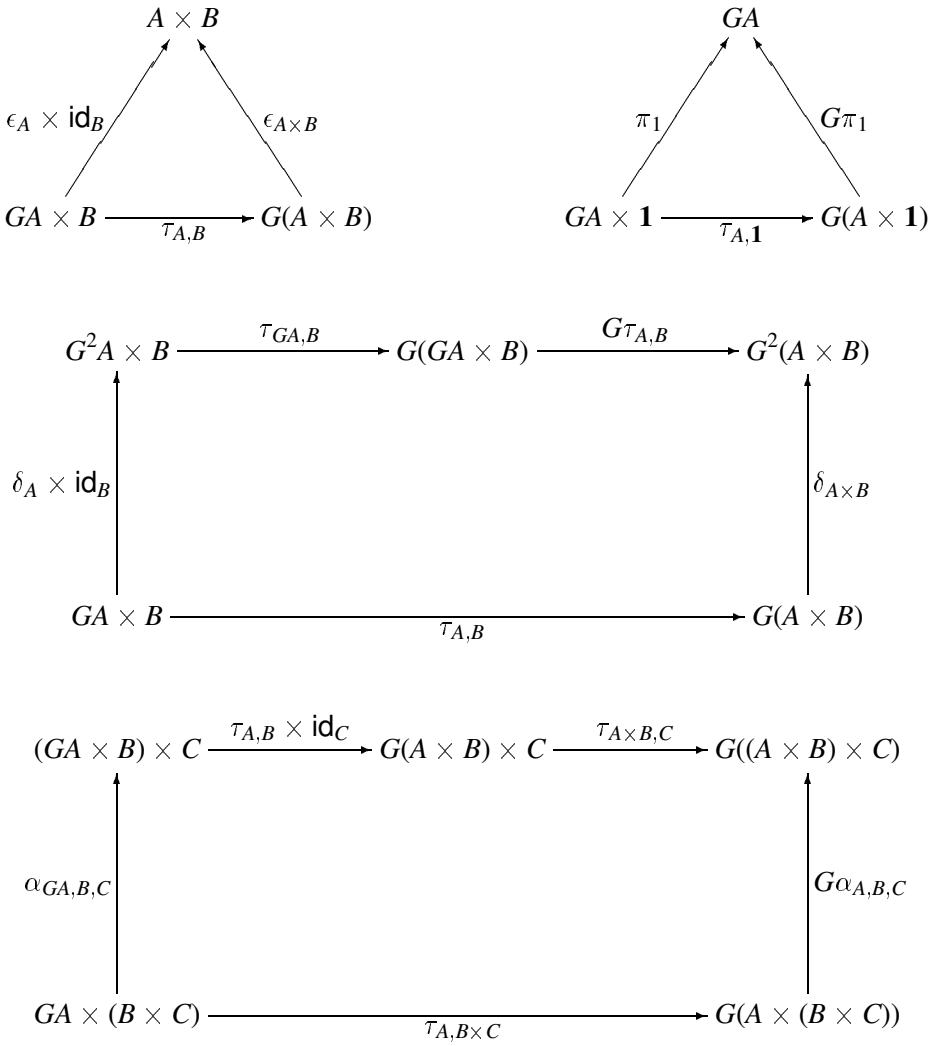

Figure 8: Requirements for a strong comonad

As for strong monads, for set-like categories whenever the function $\mathbf{g}_{A,B} = \lambda f\,.Gf$ is a morphism in that category, then G is strong. The resulting σ will be

$$\mathsf{curry}(\mathsf{uncurry}(\mathbf{g}_{V^{-A,A}} \circ \mathsf{curry}(\mathsf{app}_{X,A} \circ \langle \pi_2, \pi_1 \rangle)) \circ \langle \pi_2, \pi_1 \rangle)$$

i.e. for $g \in G[X \to A]$, $\sigma(g) = \lambda x.G(\lambda \mathbf{a}.\mathbf{a}(x))g$.

The comonad $\langle V, \mathsf{val}, \mathsf{pre} \rangle$ is strong in both **Cont** and **dI**, with the resulting σ being defined by $\sigma(\lambda v.\lambda x.a_{v,x}) = \lambda x.\lambda v.a_{v,x}$. The comonad $\langle V_S, \mathsf{sval}, \mathsf{spre} \rangle$, however is not strong in **Cont**, for the function $\lambda f\,.V_S f$ is not monotone when functions are ordered pointwise (and given a tensorial strength τ, we can derive as a morphism $\lambda f\,.V_S f$, with $\lambda f\,.V_S f = \mathsf{curry}(V_S \mathsf{app}_{A,B} \circ V_S \langle \pi_2, \pi_1 \rangle \circ \tau_{A,[A\to B]} \circ \langle \pi_2, \pi_1 \rangle)$). The product comonad is strong and the σ resulting from this construction is identical to the σ mentioned in the previous section.

6.1.3. Product as a monad

For all comonads $\langle G, \epsilon, \delta \rangle$ there is a distributive law $\sigma_A = \langle G\pi_1, \epsilon_X \circ G\pi_2 \rangle$ $(= \langle G\pi_1, \pi_2 \circ \epsilon_{A\times X} \rangle)$ of the product monad X over G. For example there is a distributive law σ of X over V, where for standard cartesian closed categories

$$\sigma_A = \langle \mathsf{curry}(\pi_1 \circ \mathsf{app}_{V(A\times X),V}), \pi_2 \circ \mathsf{app}_{V(A\times X),V} \circ \langle \mathsf{id}_{V(A\times X)}, e \circ !_{V(A\times X)} \rangle \rangle$$

or, in **Cont**,

$$\sigma_A(\lambda v.\langle a_v, x_v \rangle) = \langle \lambda v.a_v, x_e \rangle$$

For the strictly increasing paths comonad σ is the same except that duplicates are removed.

6.1.4. Lifting with the path comonads

If we look at the lifting monad $\langle L, \mathsf{up}, \mathsf{down} \rangle$ with the path comonad $\langle V, \mathsf{val}, \mathsf{pre} \rangle$, using the monoid **VNat** ($e = \omega$ and $m = \min$) in the category **Cont**, we see that an element of VLA can have one of three forms: it can be a path $\mathbf{a} \in VA$, the element $\perp^\omega$, or of the form $\perp^n \mathbf{a}$ for some path $\mathbf{a} \in VA$ and some $n > 0$. An element of LVA is either a path $\mathbf{a} \in VA$ or $\perp$.

In order for a natural transformation σ to be distributive with up we must have that if $\mathbf{a} \in VA$, $\sigma_A(\mathbf{a}) = \sigma_A(V\mathsf{up}_A(\mathbf{a})) = \mathsf{up}_{VA}(\mathbf{a}) = \mathbf{a}$. There is no value, however, that can be assigned to $\sigma_A(\perp\mathbf{a})$ that satisfies all distributivity requirements without violating monotonicity or naturality. For example, let us look at the domain $A = \mathbf{VNat}$ and the element $\perp 0^\omega$. Since $\sigma_{\mathbf{VNat}}(0^\omega) = 0^\omega$ and $\perp 0^\omega \leq 0^\omega$, by monotonicity of $\sigma_{\mathbf{VNat}}$ we must have that $\sigma_{\mathbf{VNat}}(\perp 0^\omega) \leq 0^\omega$ which in $LV\mathbf{VNat}$ means that either $\sigma_{\mathbf{VNat}}(\perp 0^\omega) = 0^\omega$ or $\sigma_{\mathbf{VNat}}(\perp 0^\omega) = \perp$.

If we let $\sigma_{\mathbf{VNat}}(\perp 0^\omega) = \perp$, then σ is not distributive with val, since $L\mathsf{val}_{\mathbf{VNat}}(\sigma_{\mathbf{VNat}}(\perp 0^\omega)) = L\mathsf{val}_{\mathbf{VNat}}(\perp) = \perp$ and $\mathsf{val}_{L\mathbf{VNat}}(\perp 0^\omega) = 0$.

So we must have that $\sigma(\perp 0^\omega) = 0^\omega$. Let $f : \mathbf{VNat} \to \mathbf{VNat}$ be the constant 1 function. Therefore by naturality of σ we have that

$$\begin{aligned} \sigma_{\mathbf{VNat}}(\perp) &= \sigma_{\mathbf{VNat}}(VLf\,(\perp 0^\omega)) \\ &= LVf\,(\sigma_{\mathbf{VNat}}(\perp 0^\omega)) \\ &= 1^\omega \end{aligned}$$

However we already know that $\sigma_{\mathbf{VNat}}(01^\omega) = 01^\omega$. Thus we have that $\perp 1^\omega \leq 01^\omega$, but $\sigma_{\mathbf{VNat}}(\perp 1^\omega) = 1^\omega > \sigma_{\mathbf{VNat}}(01^\omega)$. So σ fails to be monotone.

This proof generalizes to any non-trivial monoid V but not to the strictly increasing path comonad or to exponentiation in the category **dI**. In fact the natural transformation given by

$$\sigma_A(\mathbf{a}) = \begin{cases} \perp & \mathbf{a} = \perp^\omega \\ \mathbf{a}' & \mathbf{a} = \perp^n \mathbf{a}' \text{ for some } n > 0 \\ \mathbf{a} & \text{otherwise} \end{cases}$$

is a distributive law of L over V (V = **VNat**) in the category **dI** (σ_A is stable for all dI domains A). When we limit n to 1, σ is a distributive law of L over V_S.

6.1.5. Coproducts

There is no distributive law σ of the separated sum monad E over the exponentiation comonad V in the category **Cont**, for reasons similar to the ones above showing that there is no distributive law of the lifting monad L over V. Again there is again a distributive law σ of E over V_S and a distributive law σ of E over V in **dI** (for the monoid **VNat**), namely

$$\sigma_A(\mathbf{a}) = \begin{cases} \bot & \mathbf{a} = \bot^\omega \\ \iota_1(\langle a_i \rangle_{i=0}^\omega) & \exists n \geq 0.\mathbf{a} = \bot^n \langle \iota_1(a_i) \rangle_{i=0}^\omega \\ \epsilon(\mathbf{a}) & \text{otherwise} \end{cases}$$

This particular function depends not only on the structure of the comonad, but also on the structure of the monoid and thus does not generalize easily.

For categories with coproducts and one extra (somewhat strong) condition we do get distributivity:

Theorem 4 *Let $\mathcal{C}$ be a category with binary coproducts, $\langle G, \epsilon, \delta \rangle$ be any comonad over $\mathcal{C}$, and E be some object. Suppose that for each object A, the object $GA + GE$ is naturally isomorphic to $G(A + E)$, with one half of the isomorphism being $\phi_A = [G\iota_{1A}, G\iota_{2E}]$. Then there is a distributive law σ of the coproduct monad E over G, namely*

$$\sigma_A = (\mathsf{id}_A + \epsilon_E) \circ \phi_A^{-1}.$$

Although the above condition does not hold with the comonad V in **Cont**, it does hold for V in the category $\mathbf{Cont}_S$ of domains and strict continuous functions (using coalesced sums). For set-like categories, the above condition will hold primarily when the structure of $G(A + E)$ uniformly consists of either elements of A or elements of E.

6.1.6. Power Sets and Power Domains with the Path comonads

In the category **Set** and for all monoids V, there is a distributive law σ of the power set monad $\mathcal{P}$ over the exponentiation comonad V, with $\sigma_A : [V \to \mathcal{P}A] \to \mathcal{P}[V \to A]$ defined by

$$\sigma_A(\mathbf{X}) = \{\mathbf{a} \in VA \mid \forall v \in V.\mathbf{a}(v) \in \mathbf{X}(v)\}$$

The equivalent function in **Cont** for the Hoare or Smyth power domains, and in **Bif** for the Plotkin power domain is

$$\sigma_A(\mathbf{s}) = \{w \in \mathcal{P}^*_{\text{fin}}(K(VA)) \mid \forall v \in V.\{\mathbf{a}(v) \mid \mathbf{a} \in w\} \in \mathbf{s}(v)\}$$

For the Hoare or Smyth power domains, however, σ is not natural, and for the Plotkin power domain, it is not even monotone. In fact we can show that for all three power domains any choice of σ will fail at least one of the conditions needed for distributivity.

Theorem 5 *In the category* **Cont***, let V be the monoid* **VNat** *with unit* ω *and binary operator* min*. Then there is no distributive law of the Hoare power domain monad* $\langle P^\flat, \{\!|-|\!\}^\flat, \biguplus^\flat \rangle$ *over the path comonad* $\langle V, \mathsf{val}, \mathsf{pre} \rangle$.

Proof. Assume that there exists an indexed collection of morphisms σ such that for each domain A, $\sigma_A : VP^\flat A \to P^\flat VA$ and that σ is distributive with $\{\!|-|\!\}^\flat$, i.e. for all objects A, $\sigma_A \circ V\{\!|-|\!\}^\flat_A = \{\!|-|\!\}^\flat_{VA}$. We will show that there are domains A and B, an $\mathbf{s} \in VP^\flat A$, and a continuous function $f : A \to B$ such that σ is not natural.

Let A be the flat truth value domain **Bool**. Let $B = \mathbf{3}$, the domain of three points, $\{0, 1, 2\}$, with $0 < 1 < 2$. Let f : **Bool** $\to \mathbf{3}$ be the function with $f(\bot) = 0, f(\mathtt{tt}) = 1, f(\mathtt{ff}) = 2$. It is clear that f is monotone on a finite domain and therefore is also continuous. Let $\mathbf{s} = (\downarrow\{\mathtt{tt}\})(\downarrow\{\mathtt{tt}, \mathtt{ff}\})^\omega$. Then

$$
\begin{aligned}
\sigma_B \circ VP^\flat f\, \mathbf{s} &= \sigma_B(VP^\flat f\,((\downarrow\{\mathtt{tt}\})(\downarrow\{\mathtt{tt},\mathtt{ff}\})^\omega)) \\
&= \sigma_B(P^\flat f\,(\downarrow\{\mathtt{tt}\})P^\flat f\,(\downarrow\{\mathtt{tt},\mathtt{ff}\}^\omega)) \\
&= \sigma_B((\downarrow\{f(\mathtt{tt})\})(\downarrow\{f(\mathtt{tt}), f(\mathtt{ff})\})^\omega) && \text{using the definition of } P^\flat \\
& && \text{on compact elements} \\
&= \sigma_B((\downarrow\{1\})(\downarrow\{1,2\})^\omega) \\
&= \sigma_B((\downarrow\{1\})(\downarrow\{2\})^\omega) && \downarrow\{2\} = \downarrow\{1,2\} \\
&= \sigma_B(\{\!|\,1\,|\!\}^\flat_{\mathbf{3}} \{\!|\,2\,|\!\}^{\flat\,\omega}_{\mathbf{3}}) \\
&= \sigma_B \circ V\{\!|-|\!\}^\flat_{\mathbf{3}}(12^\omega) \\
&= \{\!|\,12^\omega\,|\!\}^\flat_{V\mathbf{3}} && \text{by assumption} \\
&= \downarrow\{12^\omega\}
\end{aligned}
$$

Now suppose that for some $s \in P^\flat V\mathbf{Bool}$ we have that $P^\flat Vf\,(s) = \downarrow\{12^\omega\}$. Given that all elements of $V\mathbf{3}$ are compact, we have that

$$
\begin{aligned}
\downarrow\{12^\omega\} &= P^\flat Vf\,(s) \\
&= \textstyle\bigcup_{u \in s} \downarrow\{Vf\,(x) \mid x \in u\} \\
&= \{w \in \mathcal{P}^*_{\mathrm{fin}}(V\mathbf{3}) \mid \exists u \in s.\, w \sqsubseteq^\flat \{Vf\,(x) \mid x \in u\}\}
\end{aligned}
$$

Thus we know that for every $u \in s$, $\{Vf\,(x) \mid x \in u\} \in P^\flat Vf\,(s)$, i.e. $\{Vf\,(x) \mid x \in u\} \sqsubseteq^\flat \{12^\omega\}$. This means that for all $\mathbf{b} \in u$, $Vf\,(\mathbf{b}) \leq 12^\omega$. We also know that since $\{12^\omega\} \in \downarrow\{12^\omega\} = P^\flat Vf\,(s)$, there must be an u such that $\{12^\omega\} \sqsubseteq^\flat \{Vf\,(x) \mid x \in u\}$, which means that there exists some $\mathbf{b} \in u$ with $12^\omega \leq Vf\,(\mathbf{b})$. Since we also have that $Vf\,(\mathbf{b}) \leq 12^\omega$, $Vf\,(\mathbf{b})$ must equal 12^ω.

However the only sequence $\mathbf{b}$ in **Bool** such that $Vf\,(\mathbf{b}) = 12^\omega$ is $\mathbf{b} = \mathtt{tt}\ \mathtt{ff}^\omega$, which is not an element of $V\mathbf{Bool}$. Thus there is no value $\sigma_{\mathbf{Bool}}$ can give to $\mathbf{s}$ to make $\sigma_B \circ VP^\flat f\,(\mathbf{s})$ equal to $P^\flat Vf\,(\sigma_{\mathbf{Bool}}(\mathbf{s}))$, so σ is not natural. □

Since the proof used valid V_S paths and did not use specific details of the ordering of the paths, we also have shown that

Corollary 6 *In the category* **Cont***, there is no distributive law of the Hoare monad* $\langle P^\flat, \{\!|-|\!\}^\flat, \biguplus^\flat \rangle$ *over the strict paths comonad* $\langle V_S, \mathsf{sval}, \mathsf{spre} \rangle$.

Also, by a straightforward extension of the proof given above, we can show that there is no distributive law for any monoid domain V that contains two points v_1 and v_2 with $v_1 <^V v_2$, i.e. any non-trivial monoid domain.

Theorem 7 *In the category* **Cont***, let V be the monoid* **VNat** *as in Theorem 5. Then there is no distributive law of the Smyth power domain* $\langle P^\sharp, \{\!|-|\!\}^\sharp, \uplus^\sharp \rangle$ *over the path comonad* $\langle V, \mathsf{val}, \mathsf{pre} \rangle$.

Proof. Assume that there exists an indexed collection of functions σ such that for each domain A, $\sigma_A : VP^\sharp A \to P^\sharp VA$. We will show that the σ's cannot simultaneously satisfy monotonicity, naturality, and distributivity with $\{\!|-|\!\}^\sharp$.

Let $\mathbf{2}$ be the two point set $\{\bot, \top\}$ with $\bot < \top$. Define $f_0, f_1 : \mathbf{Bool} \to \mathbf{2}$ as follows:

$$\begin{aligned} f_0(\bot) &= \bot \\ f_0(\mathtt{tt}) &= \top \\ f_0(\mathtt{ff}) &= \top \end{aligned}$$

$$\begin{aligned} f_1(\bot) &= \bot \\ f_1(\mathtt{tt}) &= \bot \\ f_1(\mathtt{ff}) &= \top \end{aligned}$$

Let $\mathbf{s} = (\downarrow\{\mathtt{tt}, \mathtt{ff}\})(\downarrow\{\mathtt{ff}\})^\omega \in VP^\sharp\mathbf{Bool}$. If we assume that σ is distributive with $\{\!|-|\!\}^\sharp$, i.e. for all objects A, $\sigma_A \circ V\{\!|-|\!\}^\sharp_A = \{\!|-|\!\}^\sharp_{VA}$, we get that

$$\begin{aligned} \sigma_{\mathbf{2}} \circ VP^\sharp f_0 \mathbf{s} &= \sigma_{\mathbf{2}}(VP^\sharp f_0((\downarrow\{\mathtt{tt}, \mathtt{ff}\})(\downarrow\{\mathtt{ff}\})^\omega)) \\ &= \sigma_{\mathbf{2}}((\downarrow\{\top\})^\omega) \\ &= \sigma_{\mathbf{2}}((\{\!|\top|\!\})^\sharp)^\omega) \\ &= \{\!|\top^\omega|\!\}^\sharp && \text{by assumption} \\ &= \downarrow\{\top^\omega\} \end{aligned}$$

Given any $s \in P^\sharp V\mathbf{Bool}$, we see that

$$\begin{aligned} P^\sharp Vf_0(s) &= \bigcup_{u \in s} \uplus^\sharp_{x \in u} \{\!|Vf(x)|\!\}^\sharp_{\mathbf{2}} \\ &= \bigcup_{u \in s} \downarrow\{Vf(t) \mid t \in u\} && \text{since all elements of } V\mathbf{2} \\ & && \text{are compact} \end{aligned}$$

Thus in order for $P^\sharp Vf_0(s)$ to be equal to $\downarrow\{\top^\omega\}$ there must be a $u \in s$ with $\{\top^\omega\} \sqsubseteq^\sharp \{Vf(x) \mid x \in u\}$, i.e. for all $\mathbf{t} \in u$, $\top^\omega \leq^{V\mathbf{2}} Vf(\mathbf{t})$. Since $\top^\omega$ is the maximum element of $V\mathbf{2}$, this is the same as saying that for all $\mathbf{t} \in u$, $Vf(\mathbf{t}) = \top^\omega$.

Now the elements of $V\mathbf{Bool}$ are all of the form $\bot^\omega$, $\bot^n\mathtt{tt}^\omega$, and $\bot^n\mathtt{ff}^\omega$, where n is a non-negative integer. If we apply Vf_0 to all of these we find that

$$\begin{aligned} Vf_0(\bot^\omega) &= \bot^\omega \\ Vf_0(\bot^n\mathtt{tt}^\omega) &= \bot^n\top^\omega \\ Vf_0(\bot^n\mathtt{ff}^\omega) &= \bot^n\top^\omega \end{aligned}$$

Therefore for all s with $P^\sharp Vf_0(s) = \downarrow\{\top^\omega\}$, either $\{\mathtt{tt}^\omega\}$, $\{\mathtt{ff}^\omega\}$, or $\{\mathtt{tt}^\omega, \mathtt{ff}^\omega\}$ must be in s. Since those three sets are maximal in the Smyth preorder, s must therefore be either $\downarrow\{\mathtt{tt}^\omega\}$, $\downarrow\{\mathtt{ff}^\omega\}$, or $\downarrow\{\mathtt{tt}^\omega, \mathtt{ff}^\omega\}$.

If we assume σ is natural we have that $P^\sharp Vf_0(\sigma_{\mathbf{Bool}}(\mathbf{s})) = \sigma_{\mathbf{2}}(VP^\sharp f_0(\mathbf{s}))$, but we know that $\sigma_{\mathbf{2}}(VP^\sharp f_0(\mathbf{s}))$ equals $\downarrow\{\top^\omega\}$. Therefore $\sigma_{\mathbf{Bool}}(\mathbf{s})$ must be either $\downarrow\{\mathtt{tt}^\omega\}$, $\downarrow\{\mathtt{ff}^\omega\}$, or $\downarrow\{\mathtt{tt}^\omega, \mathtt{ff}^\omega\}$

Now with f_1 we see that

$$\begin{aligned}
\sigma_{\mathbf{2}} \circ VP^\sharp f_1(\mathbf{s}) &= \sigma_{\mathbf{2}}(VP^\sharp f_1((\downarrow\{\mathtt{tt}, \mathtt{ff}\})(\downarrow\{\mathtt{ff}\})^\omega)) \\
&= \sigma_{\mathbf{2}}((\downarrow\{\bot, \top\})(\downarrow\{\top\})^\omega) \\
&= \sigma_{\mathbf{2}}((\downarrow\{\bot\})(\downarrow\{\top\})^\omega) && (\downarrow\{\bot\} = \downarrow\{\bot, \top\}) \\
&= \sigma_{\mathbf{2}}((\{\!|\bot|\!\}^\sharp)(\{\!|\top|\!\}^\sharp)^\omega) \\
&= \{\!|\bot\top^\omega|\!\}^\sharp_{V\mathbf{2}} && \text{assuming distributivity with } \{\!|-|\!\}^\sharp \\
&= \downarrow\{\bot\top^\omega\}
\end{aligned}$$

To get naturality with f_1 we need $P^\sharp Vf_1(\sigma_{\mathbf{Bool}}(\mathbf{s}))$ to equal $\downarrow\{\bot\top^\omega\}$ for one of the possible values for $\sigma_{\mathbf{Bool}}(\mathbf{s})$. However

$$\begin{aligned}
P^\sharp Vf_1(\downarrow\{\mathtt{tt}^\omega\}) &= \downarrow\{\top^\omega\} \\
Vf_1(\downarrow\{\mathtt{ff}^\omega\}) &= \downarrow\{\bot^\omega\}, \text{and} \\
P^\sharp Vf_1(\downarrow\{\mathtt{ff}^\omega, \mathtt{tt}^\omega\}) &= \downarrow\{\bot^\omega, \top^\omega\}
\end{aligned}$$

none of which are equal to $\downarrow\{\bot\top^\omega\}$. □

Corollary 8 *In the category* **Cont***, there is no distributive law of the Smyth monad* $\langle P^\sharp, \{\!|-|\!\}^\sharp, \biguplus^\sharp\rangle$ *over the strict paths comonad* $\langle V_S, \mathsf{sval}, \mathsf{spre}\rangle$.

Proof. The proof for theorem 7 can be adapted as a proof of this corollary, since the proof used only strict paths and while the proof did use the path ordering non-trivially, all statements using that ordering (namely that $\top^\omega$, $\mathtt{tt}^\omega$, and $\mathtt{ff}^\omega$ are maximal in their respective domains), hold for the strict path ordering as well. □

Even though there is no distributive law σ, for the Hoare and Smyth domains there are lax distributive laws $\varsigma^\flat$ and $\varsigma^\sharp$, in the sense that for all morphisms $f : A \to B$, $\mathbf{s} \in VP^\sharp A$ and $\mathbf{s}' \in VP^\flat A$, $\varsigma^\sharp \circ VP^\sharp f(\mathbf{s}) \leq P^\sharp Vf \circ \varsigma^\sharp(\mathbf{s})$ and $\varsigma^\flat \circ VP^\flat f(\mathbf{s}') \geq P^\flat Vf \circ \varsigma^\flat(\mathbf{s}')$ and ς satisfies all the other requirements. Here ς, defined as

$$\varsigma^\dagger_A(\mathbf{s}) = \{w \in \mathcal{P}^*_{\mathrm{fin}}(VA) \mid \forall v \in V.\{\mathbf{a}(v) \mid \mathbf{a} \in w\} \in \mathbf{s}(v)\} \qquad (\dagger \in \{\flat, \sharp\})$$

is the generalization of the distributive law for sets and exponentiation mentioned at the beginning of this section. Unfortunately, the inequality in the naturality of ς translates to an inequality in the associativity law for the double Kleisli construction, so we cannot form a Kleisli category this way.

Theorem 9 *In the category* **Bif***, let V be the monoid* **VNat** *as in Theorem 5. Then there is no distributive law of the Plotkin monad* $\langle P^\natural, \{\!|-|\!\}^\natural, \biguplus^\natural\rangle$ *over the path comonad* $\langle V, \mathsf{val}, \mathsf{pre}\rangle$.

Proof. Assume that there exists a natural transformation $\sigma : VP^\natural \xrightarrow{\bullet} P^\natural V$ and that σ is distributive with both $\{\!|-|\!\}^\natural$ and val. We will show that there exists a bifinite domain A and elements $\mathbf{s}_1$ and $\mathbf{s}_2$ in $VP^\natural A$ such that $\mathbf{s}_1 \leq \mathbf{s}_2$ but $\sigma_A(\mathbf{s}_1) \not\leq \sigma_A(\mathbf{s}_2)$.

Let A be the bifinite domain consisting of the set $\{\bot, a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2\}$ with the ordering $\leq^A$ as shown in figure 9. Let $\mathbf{s}_1 = (\downarrow\{a_1, b_1\})(\downarrow\{a_2, b_2\})^\omega$, and $\mathbf{s}_2 = (\downarrow\{c_1, d_1\})(\downarrow\{c_2, d_2\})^\omega$. Clearly $\mathbf{s}_1 \leq \mathbf{s}_2$. We will show that $\sigma_A(\mathbf{s}_1) \not\leq \sigma_A(\mathbf{s}_2)$.

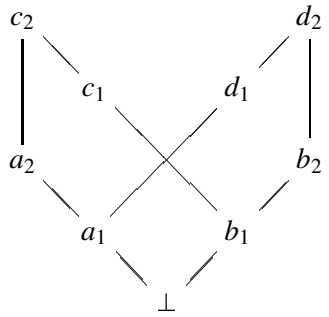

Figure 9: The domain A

Let $f : A \to \mathbf{VNat}$ be defined as the least continuous function such that:

$$\begin{aligned} f(a_i) = f(b_i) &= i \qquad i = 1 \text{ or } 2 \\ f(c_i) = f(d_i) &= i + 2 \quad i = 1 \text{ or } 2 \end{aligned}$$

Figure 10 shows the values of f for each value in A.

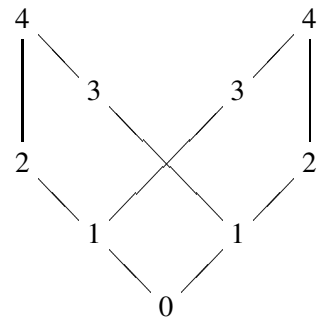

Figure 10: Definition of f on A

We then have that:

$$\begin{aligned} \sigma_{\mathbf{VNat}} \circ VP^\sharp f(\mathbf{s}_1) &= \sigma_{\mathbf{VNat}}(VP^\sharp f((\downarrow\{a_1, b_1\})(\downarrow\{a_2, b_2\})^\omega)) \\ &= \sigma_{\mathbf{VNat}}((\downarrow\{1\})(\downarrow\{2\})^\omega) \\ &= \sigma_{\mathbf{VNat}}(\{\!|\, 1 \,|\!\}^\sharp \{\!|\, 2 \,|\!\}^{\sharp\omega}) \\ &= \{\!|\, 12^\omega \,|\!\}^\sharp_{V\mathbf{VNat}} && \text{assuming distributivity with } \{\!|-|\!\}^\sharp \\ &= \downarrow\{12^\omega\} \end{aligned}$$

Suppose there exists an $s \in P^\sharp VA$ such that $P^\sharp Vf(s) = \downarrow\{12^\omega\}$. Since the range of Vf consists only of

compact elements, we have that

$$\begin{aligned}\downarrow\{12^\omega\} &= P^\natural Vf\,(s)\\ &= \textstyle\bigcup_{u\in s} \downarrow\{Vf\,(\mathbf{a}) \mid \mathbf{a} \in u\}\end{aligned}$$

so for each $u \in s$ we must have that $\{Vf\,(\mathbf{a}) \mid \mathbf{a} \in u\} \sqsubseteq^\natural \{12^\omega\}$, i.e. for all $\mathbf{a} \in u$, $\mathbf{a} \leq 12^\omega$. Since the a_i's and the b_i's are incomparable with each other in A, this means that there exists a $u_0 \subseteq \{a_1a_2^\omega, b_1b_2^\omega\}$ with $u \sqsubseteq^\natural u_0$. This implies that $s = \downarrow u_0$. By the assumption of naturality for σ we have $\downarrow\{12^\omega\} = \sigma_{\mathbf{VNat}} \circ VP^\natural f\,\mathbf{s}_1 = P^\natural Vf \circ \sigma_A \mathbf{s}_1$ so $\sigma_A\mathbf{s}_1 = \downarrow u_0$. If we then use the assumption of distributivity with val we find that

$$\begin{aligned}P^\natural \mathsf{val}_{\mathbf{VNat}}(\downarrow u_0) &= P^\natural \mathsf{val}_{\mathbf{VNat}} \circ \sigma_A \mathbf{s}_1\\ &= \mathsf{val}_{P^\natural \mathbf{VNat}} \mathbf{s}_1\\ &= \downarrow\{a_2, b_2\}\end{aligned}$$

The only value of u_0 that satisfies the above equations is $u_0 = \{a_1a_2^\omega, b_1b_2^\omega\} = \sigma_A\mathbf{s}_1$.

By a similar argument we can also show that $\sigma_A\mathbf{s}_2 = \{c_1c_2^\omega, d_1d_2^\omega\}$. However, $\{a_1a_2^\omega, b_1b_2^\omega\}$ and $\{c_1c_2^\omega, d_1d_2^\omega\}$ are incomparable, so σ_A is not monotone. □

This proof generalizes to any monoid with two distinct and related elements; however, unlike the proofs for $P^\sharp$ and $P^\flat$, it does not apply to the strict path comonad V_S, since $\mathbf{s}_1$ and $\mathbf{s}_2$ are incomparable in V_SA. Moreover, the σ given at the beginning of the section, is also not a distributive law over V_S since it not distributive with $\{\!|-|\!\}^\natural$. We do not currently know if such a distributive law exists.

6.1.7. Side Effects

If we have a comonad $\langle G, \epsilon, \delta\rangle$ that is both strong and satisfies the equation

$$\tau_{A,B} \circ (\mathsf{id}_{GA} \times \epsilon_B) \circ \langle G\pi_1, G\pi_2\rangle = \mathsf{id}_{G(A\times B)}$$

then there will be a distributive law σ of the side effect monad $\langle S, \eta, \mu\rangle$ over G, with

$$\sigma_A = \mathsf{curry}((\mathsf{id}_{GA} \times \epsilon_S) \circ \langle G\pi_1, G\pi_2\rangle \circ G\mathsf{app}_{S,A\times S} \circ \tau_{SA,S})$$

This condition is rather stringent, essentially requiring that information lost when applying ϵ to GB is "stored" in GA so that it can later be recovered by τ. Even though this condition holds for the product comonad, for the exponentiation comonad (and its variant, the strict path comonad), it is equivalent to requiring that the monoid V is isomorphic to $\mathbf{1}$. In fact the above definition for σ, which in **Cont** turns out to be

$$\sigma_A(\lambda v.\lambda s.\langle a_v(s), z_v(s)\rangle) = \lambda s.\langle \lambda v.a_v(s), z_e(s)\rangle,$$

does not distribute with μ for non-trivial monoids. It is highly unlikely that any distributive law exists for S over any non-trivial exponentiation comonads.

7. Non-distributive double Kleisli categories

An alternative method to using both the comonad and the monad simultaneously is to lift the comonad into the Kleisli category of the monad. One way is to let $\hat{GA} = GA$ and then coerce the morphism part of the

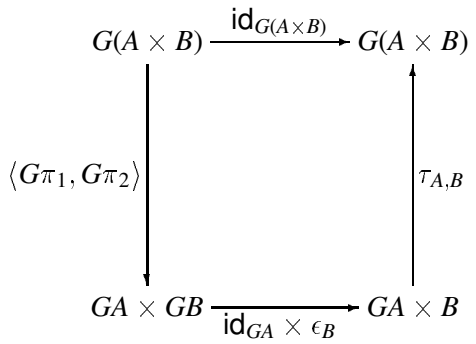

Figure 11: The condition $\tau_{A,B} \circ (\mathsf{id}_{GA} \times \epsilon_B) \circ \langle G\pi_1, G\pi_2 \rangle = \mathsf{id}_{G(A\times B)}$

	Comonads				
Monads	Products	General Exponentiation	Paths in **Cont** (V = **VNat**)	Strict Paths or Paths in **dI**	Other comonad
Lifting	yes	—	no	yes	—
Exceptions	yes	see sec. 6.1.5	no	yes	if $G(A + E) \cong GA + GE$
Products	yes	yes	yes	yes	yes
Side Effects	yes	see section 6.1.7			
Power sets	yes	yes	—	—	—
$P^\flat$	yes	—	no	no	—
$P^\sharp$	yes	—	no	no	—
$P^\natural$	yes	—	no	see sec. 6.1.6	—
Exponentiation	yes	if strong	yes	**dI** only	if strong
Other monad	if strong	—	—	—	—

Table 1: Distributive laws of various monads over comonads

functor and the natural transformations into the Kleisli category $\mathcal{K}(T)$ (see figure 12). In order to get a valid comonad, however, we need a distributive law σ of G over T as before and in fact the resulting Kleisli category $\mathcal{K}(\hat{G})$ is identical to the doubly lifted Kleisli category $\mathcal{K}(G,T)$. A similar result holds if we try to lift the monad in to the Kleisli category of the comonad as shown in figure 12 (For similar constructions, see [4] and [1]).

Since there are several combinations of monad and comonads that do not have distributive laws, we need to find a way of combining them without one. One possibility is to lift the comonad using $G'A = GTA$:

Theorem 10 *Let $\langle T, \eta, _^* \rangle$ be a Kleisli triple and $\langle G, \epsilon, _^+ \rangle$ be a Kleisli cotriple on a category $\mathcal{C}$. Then there*

$$\begin{array}{ll} \hat{G}A = GA & \hat{T}A = TA \\ \hat{G}f = \sigma_A \circ Gf : GA \to TGA & \hat{T}f = Tf \circ \sigma_A : GTA \to TA \\ \hat{\epsilon}_A = \eta_A \circ \epsilon_A : GA \to TA & \hat{\eta}_A = \eta_A \circ \epsilon_A : GA \to TA \\ \hat{\delta}_A = \eta_{G^2A} \circ \delta_A : GA \to TG^2A & \hat{\mu}_A = \mu_A \circ \epsilon_{T^2A} : GT^2A \to TA \end{array}$$

Figure 12: Lifting the comonad or monad using a distributive law

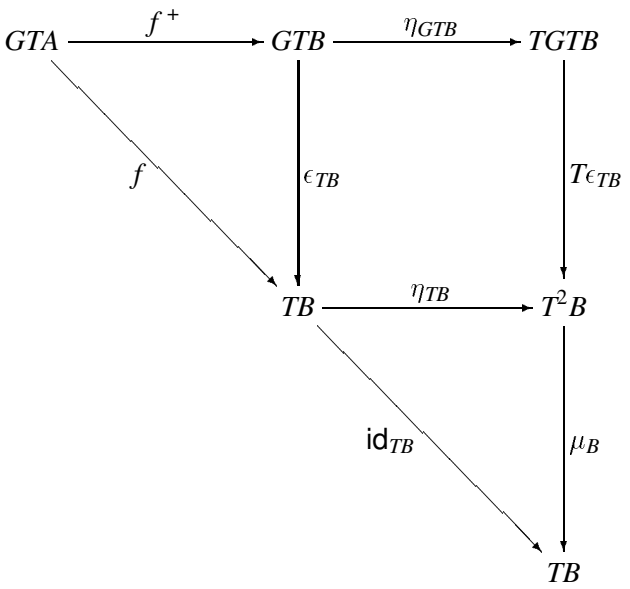

Figure 13: Proof that $\epsilon'_B \overset{T}{\circ} f^{+'} = f$.

exists a Kleisli cotriple $\langle G', \epsilon', _^{+'} \rangle$ *on the Kleisli category* $\mathcal{K}(T)$ *defined as follows:*

- *For all objects A,* $G'A$ *is the object* GTA*.*
- *For all objects A,* $\epsilon'_A : GTA \to TA$ *is* ϵ_{TA}*.*
- *For all morphisms* $f : GTA \to TB$*,* $f^{+'} : GTA \to TGTB$ *is the morphism* $\eta_{GTB} \circ f^+$*.*

Proof. The proof involves using straightforward diagram chasing to show that the three conditions specified in section 4 are satisfied. Figure 13 contains a diagram proving that for all $f : GTA \to TB$, $\epsilon'_B \overset{T}{\circ} f^{+'} = f$ □

Given the above, we can construct a lifted comonad $\langle G', \epsilon', \delta' \rangle$ from the lifted Kleisli cotriple $\langle G', \epsilon', _^{+'} \rangle$. It can easily be shown by simplifying the definition given in section 4 that the resulting comonad is defined as follows:

- For all objects A, $G'A$ is the object GTA
- For all $f : A \to TB$, $G'f : GTA \to TGTB$ is $\eta_{GTB} \circ G(\mu_B \circ Tf)$
- For all objects A, $\epsilon'_A : GTA \to TA$ is ϵ_{TA}
- For all objects A, $\delta'_A : GTA \to TGTGTA$ ($= T(G')^2A$) is $\eta_{GTGTA} \circ G\eta_{GTA} \circ \delta_{TA}$

We can now construct the Kleisli category $\mathcal{K}(G')$ from $\mathcal{K}(T)$. Again it is straightforward to show that the resulting category has the form

- Objects of $\mathcal{K}(G')$ are the same as for $\mathcal{C}$.

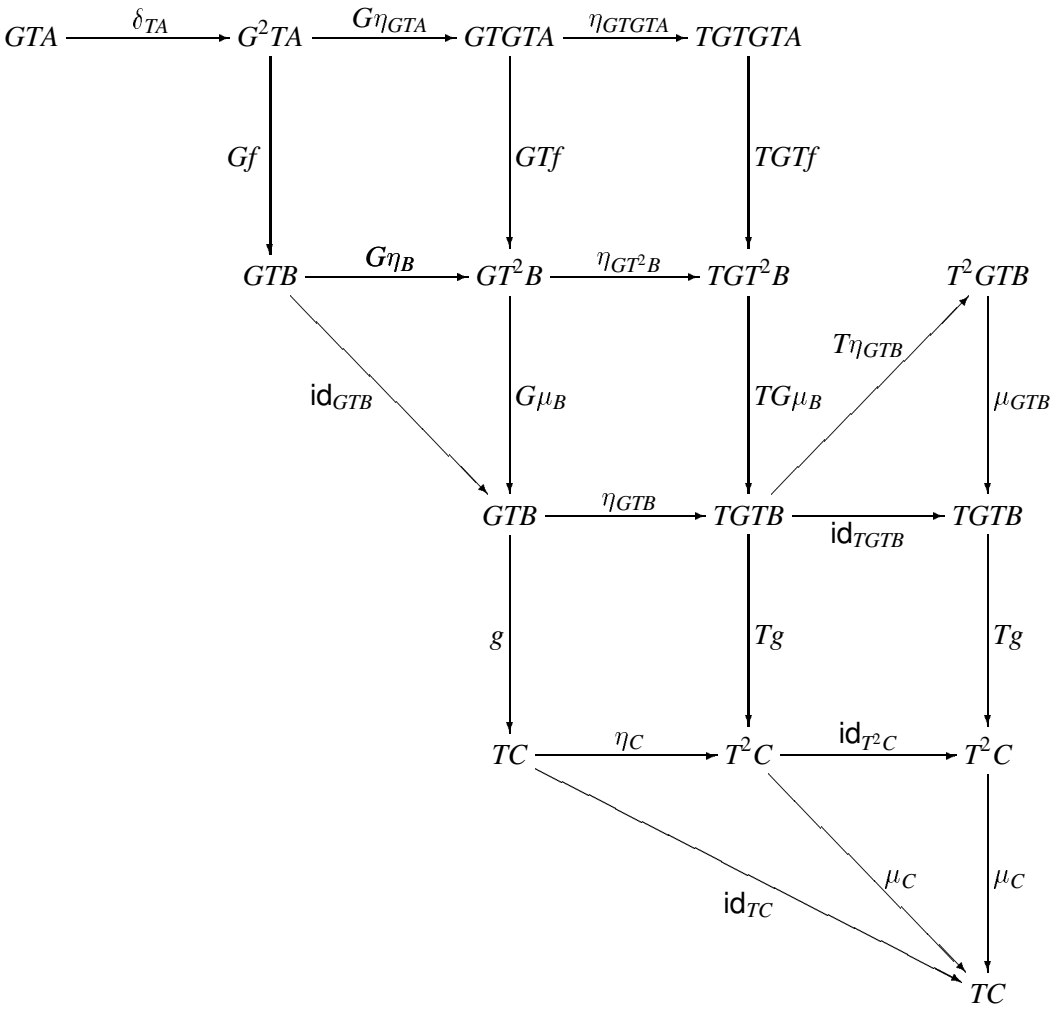

Figure 14: Proof that $g \overset{T}{\circ} G'f \overset{T}{\circ} \delta'_A = g \circ Gf \circ \delta_{TA}$. The upper part of the diagram represents $g \overset{T}{\circ} (G'f \overset{T}{\circ} \delta'_A) = \mu_C \circ Tg \circ \mu_{GTB} \circ T\eta_{GTB} \circ TG\mu_B \circ TGTf \circ \eta_{GTGTA} \circ G\eta_{GTA} \circ \delta_{TA}$.

- A morphism $f : A \overset{G'}{\to} B$ is a morphism $f : GTA \to TB$ in $\mathcal{C}$.
- For all objects A, the identity morphism on A, $\mathsf{id}_A^{G'}$, is $\epsilon'_A = \epsilon_{TA}$.
- For all $f : A \overset{G'}{\to} B$ and $g : B \overset{G'}{\to} C$, their composition $g \overset{G'}{\circ} f$ is $g \overset{T}{\circ} G'f \overset{T}{\circ} \delta'_A = g \circ Gf \circ \delta_{TA}$.

Figure 14 contains a diagram proving that $g \overset{T}{\circ} G'f \overset{T}{\circ} \delta'_A = g \circ Gf \circ \delta_{TA}$.

Using similar definitions, we can also construct a lifted monad $\langle T'A, \eta'_A, \mu'_A \rangle$ over the Kleisli category $\mathcal{K}(G)$ and then form the Kleisli category $\mathcal{K}(T')$. It is straightforward to show that they have the following forms:

- For all objects A, $T'A$ is the object TGA

$$\vdash \rho \text{ type} \qquad \frac{\vdash \tau \text{ type}}{\vdash T\tau \text{ type}}$$

$$\frac{\vdash \tau \text{ type}}{x : \tau \vdash x : \tau} \qquad \frac{x : \tau \vdash e : \mathrm{dom}(p)}{x : \tau \vdash p(e) : \mathrm{ran}(p)}$$

$$\frac{x : \tau \vdash e : \tau'}{x : \tau \vdash [e] : T\tau'} \qquad \frac{x : \tau \vdash e : T\tau'}{x : \tau \vdash \mu(e) : \tau'}$$

$$\frac{x : \tau \vdash e_1 : \tau_1 \quad x_1 : \tau_1 \vdash e_2 : \tau_2}{x : \tau \vdash \texttt{let } x_1 \Leftarrow e_1 \texttt{ in } e_2 : \tau_2}$$

Figure 15: Typing rules for simple programming language

- For all $f : GA \to B$, $T'f : GTGA \to TGB$ is $T(Gf \circ \delta_A) \circ \epsilon_{TGA}$
- For all objects A, $\eta'_A : GA \to TGA$ is η_{GA}
- For all objects A, $\mu'_A : GTGTGA (= G(T')^2A) \to TGA$ is $\mu_{GA} \circ T\epsilon_{TGA} \circ \epsilon_{TGTGA}$

and

- Objects in $\mathcal{K}(T')$ are objects in $\mathcal{C}$.
- A morphism $f : A \xrightarrow{T'} B$ is a morphism $f : GA \to TGB$ in $\mathcal{C}$.
- For all objects A, the identity arrow on A, $\mathsf{id}^{T'}_A$ is $\eta'_A = \eta_{GA}$.
- For all $f : A \xrightarrow{T'} B$ and $g : B \xrightarrow{T'} C$ their composition $g \overset{T'}{\circ} f$ is $\mu_{GC} \circ Tg \circ f$

8. Examples with monadic languages

To apply some of these constructions, let us look at the "simple programming language" described in [14], with the following syntax:

$$\begin{aligned} \tau &::= \rho \mid T\rho \\ e &::= x \mid p(e) \mid [e] \mid \mu(e) \mid \texttt{let } x \Leftarrow e_1 \texttt{ in } e_2 \end{aligned}$$

Here ρ ranges over a set of atomic types, x over a set of variables, and p over a set of constant function symbols. Note that expressions contain exactly one free variable.

The typing rules for this language are listed in figure 15, consisting of definitions for the judgements $\vdash \tau$ type (denoting a valid type) and $x : \tau \vdash e : \tau'$. For each constant p we assume a given domain type $\mathrm{dom}(p)$ and a range type $\mathrm{ran}(p)$.

$$\frac{x:\tau \vdash e:\tau'}{x:\tau \vdash e \equiv_{\tau'} e} \qquad \frac{x:\tau \vdash e_1 \equiv_{\tau'} e_2 \quad x:\tau \vdash e_2 \equiv_{\tau'} e_3}{x:\tau \vdash e_1 \equiv_{\tau'} e_3}$$

$$\frac{x:\tau \vdash e_1 \equiv_{\tau'} e_2}{x:\tau \vdash e_2 \equiv_{\tau'} e_1} \qquad \frac{x:\tau \vdash e_1 \equiv_{\mathrm{dom}(p)} e_2}{x:\tau \vdash p(e_1) \equiv_{\mathrm{ran}(p)} p(e_2)}$$

$$\frac{x:\tau \vdash e_1 \equiv_{\tau'} e_2}{x:\tau \vdash [e_1] \equiv_{T\tau'} [e_2]} \qquad \frac{x:\tau \vdash e_1 \equiv_{T\tau'} e_2}{x:\tau \vdash \mu(e_1) \equiv_{\tau'} \mu(e_2)}$$

$$\frac{x:\tau \vdash e:\tau'}{x:\tau \vdash \mu([e]) \equiv_{\tau'} e} \qquad \frac{x:\tau \vdash e_1 \equiv_{\tau_1} e_1' \quad x_1:\tau_1 \vdash e_2 \equiv_{\tau_2} e_2'}{x:\tau \vdash \texttt{let}\ x_1 \Leftarrow e_1\ \texttt{in}\ e_2 \equiv_{\tau_2} \texttt{let}\ x_1 \Leftarrow e_1'\ \texttt{in}\ e_2'}$$

$$\frac{x:\tau \vdash e\downarrow_{T\tau'}}{x:\tau \vdash [\mu(e)] \equiv_{T\tau'} e} \qquad \frac{x:\tau \vdash e:\mathrm{dom}(p)}{x:\tau \vdash p(e) \equiv_{\mathrm{ran}(p)} \texttt{let}\ x_1 \Leftarrow e\ \texttt{in}\ p(x_1)}$$

$$\frac{x:\tau \vdash e:\tau'}{x:\tau \vdash \texttt{let}\ x_1 \Leftarrow e\ \texttt{in}\ x_1 \equiv_{\tau'} e} \qquad \frac{x:\tau \vdash e_1\downarrow_{\tau_1} \quad x_1:\tau_1 \vdash e_2:\tau_2}{x:\tau \vdash \texttt{let}\ x_1 \Leftarrow e_1\ \texttt{in}\ e_2 \equiv_{\tau_2} [e_1/x_1]e_2}$$

$$\frac{x:\tau \vdash e_1:\tau_1 \qquad x_1:\tau_1 \vdash e_2:\tau_2 \qquad x_2:\tau_2 \vdash e_3:\tau_3}{x:\tau \vdash \texttt{let}\ x_2 \Leftarrow (\texttt{let}\ x_1 \Leftarrow e_1\ \texttt{in}\ e_2)\ \texttt{in}\ e_3 \equiv_{\tau_3} \texttt{let}\ x_1 \Leftarrow e_1\ \texttt{in}\ (\texttt{let}\ x_2 \Leftarrow e_2\ \texttt{in}\ e_3)}$$

Figure 16: Equivalence rules for simple programming language

$$\frac{\vdash \tau\ \text{type}}{x:\tau \vdash x\downarrow_{\tau}} \qquad \frac{x:\tau \vdash e_1 \equiv_{\tau'} e_2 \quad x:\tau \vdash e_1\downarrow_{\tau'}}{x:\tau \vdash e_2\downarrow_{\tau'}} \qquad \frac{x:\tau \vdash e:\tau'}{x:\tau \vdash [e]\downarrow_{T\tau'}}$$

$$\frac{x:\tau \vdash e\downarrow_{\tau_1} \quad x_1:\tau_1 \vdash \phi}{x:\tau \vdash [e/x_1]\phi}$$ where ϕ is one of $e_1 \equiv_{\tau'} e_2$, $e_1:\tau'$, or $e_1\downarrow_{\tau'}$.

Figure 17: Existence assertion rules for simple programming language

The operational behavior of the language is described by an equivalence relation $x : \tau \vdash e \equiv_{\tau'} e'$ (see figure 16) and an existence relation $x : \tau \vdash e \downarrow_{\tau'}$ (see figure 17). The existence relation determines whether or not an expression can be considered to have the form $[e]$.

The denotation semantic functions considered for this language will all interpret types as objects and typing judgments as morphisms in some category. A semantic function is *sound* if

1. whenever $x : \tau \vdash e_1 \equiv_{\tau'} e_2$, the meaning of $x : \tau \vdash e_1 : \tau'$ is equal to the meaning of $x : \tau \vdash e_2 : \tau'$
2. whenever $x : \tau \vdash e \downarrow_{\tau'}$, there exists a unique morphism h from the meaning of τ to the meaning of τ' such that the meaning of $x : \tau \vdash e : \tau'$ equals $\eta \circ h$.

8.1. Extensional semantics

For a category $\mathcal{C}$ with a monad $\langle T, \eta, \mu \rangle$ we define the extensional denotational semantics, taken from [14], for this language as follows:

- For each base type ρ, let A_ρ be some object in $\mathcal{C}$.
- If $\tau = T\tau'$, let $A_\tau = TA_{\tau'}$.
- For each p, let $[\![p]\!]$ be a morphism from $A_{\mathrm{dom}(p)}$ to $TA_{\mathrm{ran}(p)}$.
- For each judgment $x : \tau \vdash e : \tau'$, let $\mathcal{M}[\![x : \tau \vdash e : \tau']\!]$ be a morphism from A_τ to $TA_{\tau'}$ (i.e. a morphism from A_τ to $A_{\tau'}$ in the Kleisli category), defined inductively as follows:

$$
\begin{aligned}
\mathcal{M}[\![x : \tau \vdash x : \tau]\!] &= \eta_{A_\tau} \\
&= \mathsf{id}^T_{A_\tau} \\
\mathcal{M}[\![x : \tau \vdash p(e) : \mathrm{ran}(p)]\!] &= \mu_{A_{\mathrm{ran}(p)}} \circ T[\![p]\!] \circ \mathcal{M}[\![x : \tau \vdash e : \mathrm{dom}(p)]\!] \\
&= T[\![p]\!] \overset{T}{\circ} \mathcal{M}[\![x : \tau \vdash e : \mathrm{dom}(p)]\!] \\
\mathcal{M}[\![x : \tau \vdash [e] : T\tau']\!] &= \eta_{TA_{\tau'}} \circ \mathcal{M}[\![x : \tau \vdash e : \tau']\!] \\
\mathcal{M}[\![x : \tau \vdash \mu(e) : \tau']\!] &= \mu_{A_{\tau'}} \circ \mathcal{M}[\![x : \tau \vdash e : T\tau']\!] \\
\mathcal{M}[\![x : \tau \vdash \texttt{let } x_1 \Leftarrow e_1 \texttt{ in } e_2 : \tau_2]\!] & \\
&= \mu_{A_{\tau_2}} \circ T\mathcal{M}[\![x_1 : \tau_1 \vdash e_2 : \tau_2]\!] \circ \mathcal{M}[\![x : \tau \vdash e_1 : \tau_1]\!] \\
&= \mathcal{M}[\![x_1 : \tau_1 \vdash e_2 : \tau_2]\!] \overset{T}{\circ} \mathcal{M}[\![x : \tau \vdash e_1 : \tau_1]\!]
\end{aligned}
$$

Moggi in [14] showed that if the monad satisfies the *mono-requirement*, namely if η_A is a monomorphism for every object A, the above semantics is sound. It is easy to see [14] that all the monads given in this paper satisfy the mono-requirement.

8.2. Intensional semantics

Suppose we have a category $\mathcal{C}$ with both a monad $\langle T, \eta, \mu \rangle$ and a comonad $\langle G, \epsilon, \delta \rangle$. It is easy to show that if a monad $\langle T, \eta, \mu \rangle$ satisfies the mono-requirement then so do the lifted monads $\langle \hat{T}, \hat{\eta}, \hat{\mu} \rangle$ and $\langle T', \eta', \mu' \rangle$. We can then use the extensional semantics above to define an intensional semantics by starting with the Kleisli category $\mathcal{K}(G)$ instead of $\mathcal{C}$ and using a lifted monad instead of the original. Thus we get the following two intensional semantics:

$$
\begin{aligned}
\mathcal{M}[\![x:\tau \vdash x:\tau]\!]_{G,T} &= \hat{\eta}_{A_\tau^{G,T}} \\
&= \eta_{A_\tau^{G,T}} \circ \epsilon_{A_\tau^{G,T}} \\
&= \mathsf{id}_{A_\tau^{G,T}}^{G,T} \\
\mathcal{M}[\![x:\tau \vdash p(e):\mathrm{ran}(p)]\!]_{G,T} &= \hat{\mu}_{A_{\mathrm{ran}(p)}^{G,T}} \overset{G}{\circ} \hat{T}[\![p]\!]_{G,T} \overset{G}{\circ} \mathcal{M}[\![x:\tau \vdash e:\mathrm{dom}(p)]\!]_{G,T} \\
&= \mu_{\mathrm{ran}(p)} \circ T[\![p]\!]_{G,T} \circ \sigma_{A_{\mathrm{dom}(p)}^{G,T}} \circ G\mathcal{M}[\![x:\tau \vdash e:\mathrm{dom}(p)]\!]_{G,T} \circ \delta_{A_\tau^{G,T}} \\
&= [\![p]\!]_{G,T} \overset{G,T}{\circ} \mathcal{M}[\![x:\tau \vdash e:\mathrm{dom}(p)]\!]_{G,T} \\
\mathcal{M}[\![x:\tau \vdash [e]:T\tau']\!]_{G,T} &= \hat{\eta}_{TA_{\tau'}^{G,T}} \overset{G}{\circ} \mathcal{M}[\![x:\tau \vdash e:\tau']\!]_{G,T} \\
&= \eta_{TA_{\tau'}^{G,T}} \circ \mathcal{M}[\![x:\tau \vdash e:\tau']\!]_{G,T} \\
\mathcal{M}[\![x:\tau \vdash \mu(e):\tau']\!]_{G,T} &= \hat{\mu}_{A_{\tau'}^{G,T}} \overset{G}{\circ} \mathcal{M}[\![x:\tau \vdash e:T\tau']\!]_{G,T} \\
&= \mu_{A_{\tau'}^{G,T}} \circ \mathcal{M}[\![x:\tau \vdash e:T\tau']\!]_{G,T} \\
\mathcal{M}[\![x:\tau \vdash \texttt{let } x_1 \Leftarrow e_1 \texttt{ in } e_2:\tau_2]\!]_{G,T} & \\
&= \hat{\mu}_{A_{\tau_2}^{G,T}} \overset{G}{\circ} \hat{T}\mathcal{M}[\![x_1:\tau_1 \vdash e_2:\tau_2]\!]_{G,T} \overset{G}{\circ} \mathcal{M}[\![x:\tau \vdash e_:\tau_1]\!]_{G,T} \\
&= \mu_{A_{\tau_2}^{G,T}} \circ T\mathcal{M}[\![x_1:\tau_1 \vdash e_2:\tau_2]\!]_{G,T} \circ \sigma_{A_{\tau_1}^{G,T}} \circ G\mathcal{M}[\![x:\tau \vdash e_1:\tau_1]\!]_{G,T} \circ \delta_{A_\tau^{G,T}} \\
&= \mathcal{M}[\![x_1:\tau_1 \vdash e_2:\tau_2]\!]_{G,T} \overset{G,T}{\circ} \mathcal{M}[\![x:\tau \vdash e_1:\tau_1]\!]_{G,T}
\end{aligned}
$$

Figure 18: Denotational semantics using the monad $\langle \hat{T}, \hat{\eta}, \hat{\mu} \rangle$

1. Using the lifted monad $\langle \hat{T}, \hat{\eta}, \hat{\mu} \rangle$ (thus assuming the existence of a distributive law) we define

   - For each base type ρ, $A_\rho^{G,T} = A_\rho$, and for type $\tau = T\tau'$, $A_\tau^{G,T} = \hat{T}A_{\tau'}^{G,T} = TA_{\tau'}^{G,T}$.
   - For each constant p, let $[\![p]\!]_{G,T}$ be a morphism from $GA_{\mathrm{dom}(p)}^{G,T}$ to $TA_{\mathrm{ran}(p)}^{G,T}$.
   - For each type judgement $x:\tau \vdash e:\tau'$, let $\mathcal{M}[\![x:\tau \vdash e:\tau']\!]_{G,T} : GA_\tau^{G,T} \to TA_{\tau'}^{G,T}$ be defined as shown in figure 18.

2. Using the lifted monad $\langle T', \eta', \mu' \rangle$ we define

   - For each base type ρ, let $A_\rho^{T'} = A_\rho$, and for $\tau = T\tau'$, let $A_\tau^{T'} = T'A_{\tau'}^{T'} = TGA_{\tau'}^{T'}$.
   - For each constant p, let $[\![p]\!]_{T'}$ be a morphism from $GA_{\mathrm{dom}(p)}^{T'}$ to $TGA_{\mathrm{ran}(p)}^{T'}$.
   - For each type judgement $x:\tau \vdash e:\tau'$, let $\mathcal{M}[\![x:\tau \vdash e:\tau']\!]_{T'} : GA_\tau^{T'} \to TGA_{\tau'}^{T'}$ be defined as show in figure 19.

These two semantics are clearly sound; they are essentially the extensional semantics using different base categories and monads, and all such semantics are sound whenever the monad satisfies the mono-requirement.

One more semantic function can be derived for this language by taking the original semantics and directly adding a comonad to get a semantic definition in the other non-distributive Kleisli category, where a morphism from A to B is a morphism from GTA to TB in the original category.

- For each base type ρ, let $A_\rho^{G'} = A_\rho$, and for $\tau = T\tau'$, let $A_\tau^{G'} = TA_{\tau'}^{G'}$.

$$
\begin{aligned}
\mathcal{M}[\![x:\tau \vdash x:\tau]\!]_{T'} &= \eta'_{A^{T'}_{\tau}} \\
&= \eta_{GA^{T'}_{\tau}} \\
&= \mathsf{id}^{T'}_{A^{T'}_{\tau}} \\
\mathcal{M}[\![x:\tau \vdash p(e):\mathrm{ran}(p)]\!]_{T'} &= \mu'_{A^{T'}_{\mathrm{ran}(p)}} \overset{G}{\circ} T'[\![p]\!]_{T'} \overset{G}{\circ} \mathcal{M}[\![x:\tau \vdash e:\mathrm{dom}(p)]\!]_{T'} \\
&= \mu_{GA^{T'}_{\mathrm{ran}(p)}} \circ T[\![p]\!]_{T'} \circ \mathcal{M}[\![x:\tau \vdash e:\mathrm{dom}(p)]\!]_{T'} \\
&= [\![p]\!]_{T'} \overset{T'}{\circ} \mathcal{M}[\![x:\tau \vdash e:\mathrm{dom}(p)]\!]_{T'} \\
\mathcal{M}[\![x:\tau \vdash [e]:T\tau']\!]_{T'} &= \eta'_{T'A^{T'}_{\tau'}} \overset{G}{\circ} \mathcal{M}[\![x:\tau \vdash e:\tau']\!]_{T'} \\
&= \eta_{GTGA^{T'}_{\tau'}} \circ G\mathcal{M}[\![x:\tau \vdash e:\tau']\!]_{T'} \circ \delta_{A^{T'}_{\tau}} \\
\mathcal{M}[\![x:\tau \vdash \mu(e):\tau']\!]_{T'} &= \mu'_{A^{T'}_{\tau'}} \overset{G}{\circ} \mathcal{M}[\![x:\tau \vdash e:T\tau']\!]_{T'} \\
&= \mu_{GA^{T'}_{\tau'}} \circ T\epsilon_{TGA^{T'}_{\tau'}} \circ \mathcal{M}[\![x:\tau \vdash e:T\tau']\!]_{T'} \\
\mathcal{M}[\![x:\tau \vdash \texttt{let } x_1 \Leftarrow e_1 \texttt{ in } e_2:\tau_2]\!]_{T'} & \\
&= \mu'_{A^{T'}_{\tau_2}} \overset{G}{\circ} T'\mathcal{M}[\![x_1:\tau_1 \vdash e_2:\tau_2]\!]_{T'} \overset{G}{\circ} \mathcal{M}[\![x:\tau \vdash e_1:\tau_1]\!]_{T'} \\
&= \mu_{GA^{T'}_{\tau_2}} \circ T\mathcal{M}[\![x_1:\tau_1 \vdash e_2:\tau_2]\!]_{T'} \circ \mathcal{M}[\![x:\tau \vdash e_1:\tau_1]\!]_{T'} \\
&= \mathcal{M}[\![x_1:\tau_1 \vdash e_2:\tau_2]\!]_{T'} \overset{T'}{\circ} \mathcal{M}[\![x:\tau \vdash e_1:\tau_1]\!]_{T'}
\end{aligned}
$$

Figure 19: Denotational semantics using the monad $\langle T', \eta', \mu' \rangle$

- For each constant p, let $[\![p]\!]_{G'}$ be a morphism from $GTA^{G'}_{\mathrm{dom}(p)}$ to $TA^{G'}_{\mathrm{ran}(p)}$
- For each judgment $x:\tau \vdash e:\tau'$, let $\mathcal{M}[\![x:\tau \vdash e:\tau']\!]_{G'} : GTA^{G'}_{\tau} \to TA^{G'}_{\tau'}$ be defined as shown in figure 20

Unfortunately, this semantics is not sound ($x:\tau \vdash x\downarrow_{\tau}$ is false). This is because we lack control over the monad part of *GTA*. If instead we restrict *GTA* to the range of $G\eta_A$ we can get the following "semi-soundness" property:

Theorem 11 *For all expressions e, e_1, and e_2 the following two properties hold:*

1. *If $x:\tau \vdash e_1 \equiv_{\tau'} e_2$ then $\mathcal{M}[\![x:\tau \vdash e_1:\tau']\!]_{G'} \circ G\eta_{A^{G'}_{\tau}} = \mathcal{M}[\![x:\tau \vdash e_2:\tau']\!]_{G'} \circ G\eta_{A^{G'}_{\tau}}$.*
2. *If $x:\tau \vdash e\downarrow_{\tau'}$ then there exists an $h : GA^{G'}_{\tau} \to A^{G'}_{\tau'}$ such that $\mathcal{M}[\![x:\tau \vdash e:\tau']\!]_{G'} \circ G\eta_{A^{G'}_{\tau}} = \eta_{A^{G'}_{\tau'}} \circ h$.*

Proof. By structural induction on e, using as a substitution lemma

$$\mathcal{M}[\![x:\tau \vdash [e/x]e':\tau'']\!]_{G'} = \mathcal{M}[\![x:\tau' \vdash e':\tau'']\!]_{G'} \circ G\mathcal{M}[\![x:\tau \vdash e:\tau']\!]_{G'} \circ \delta_{TA^{G'}_{\tau}}$$

whenever $x:\tau \vdash e:\tau'$ and $x:\tau' \vdash e':\tau''$. □

$$
\begin{array}{ll}
\mathcal{M}[\![x:\tau \vdash x:\tau]\!]_{G'} & = \epsilon_{TA^{G'}_{\tau}} \\
 & = \mathsf{id}^{G'}_{A^{G'}_{\tau}} \\
\mathcal{M}[\![x:\tau \vdash p(e):\mathrm{ran}(p)]\!]_{G'} & = [\![p]\!]_{G'} \circ G\mathcal{M}[\![x:\tau \vdash e:\mathrm{dom}(p)]\!]_{G'} \circ \delta_{TA^{G'}_{\tau}} \\
 & = [\![p]\!]_{G'} \overset{G'}{\circ} \delta_{TA^{G'}_{\tau}} \\
\mathcal{M}[\![x:\tau \vdash [e]:T\tau']\!]_{G'} & = \eta_{TA^{G'}_{\tau'}} \circ \mathcal{M}[\![x:\tau \vdash e:\tau']\!]_{G'} \\
\mathcal{M}[\![x:\tau \vdash \mu(e):\tau']\!]_{G'} & = \mu_{A^{G'}_{\tau'}} \circ \mathcal{M}[\![x:\tau \vdash e:T\tau']\!]_{G'} \\
\mathcal{M}[\![x:\tau \vdash \texttt{let}\ \ x_1 \Leftarrow e_1\ \ \texttt{in}\ \ e_2:\tau_2]\!]_{G'} & \\
 & = \mathcal{M}[\![x_1:\tau_1 \vdash e_2:\tau_2]\!]_{G'} \circ G\mathcal{M}[\![x:\tau \vdash e_1:\tau_1]\!]_{G'} \circ \delta_{TA^{G'}_{\tau}} \\
 & = \mathcal{M}[\![x_1:\tau_1 \vdash e_2:\tau_2]\!]_{G'} \overset{G'}{\circ} \mathcal{M}[\![x:\tau \vdash e_1:\tau_1]\!]_{G'}
\end{array}
$$

Figure 20: Denotational semantics using the comonad $\langle G', \epsilon', \delta' \rangle$

8.3. Relating intensional and extensional semantics

For all comonads $\langle G, \epsilon, \delta \rangle$ there is an (adjoint) pair of functors $F : \mathcal{C} \to \mathcal{K}(G)$ and $U : \mathcal{K}(G) \to \mathcal{C}$ relating the Kleisli category $\mathcal{K}(G)$ to the underlying category, where

- For all objects A, $FA = A$
- For all $f : A \to B$, $Ff = \epsilon_A \circ Gf = f \circ \epsilon_A$
- For all objects A, $UA = GA$
- For all $a : GA \to B$, $Ua = Ga \circ \delta_A$.

(see [2] and [6]). There is also a similar pair of functors, of course, for monads.

For some comonads there is a much closer relation:

Definition 5 A *computational* comonad $\langle G, \epsilon, \delta, \gamma \rangle$[6] is a comonad $\langle G, \epsilon, \delta \rangle$ plus a natural transformation $\gamma : I \overset{\bullet}{\to} G$ such that

- $\epsilon_A \circ \gamma_A = \mathsf{id}_A$
- $\delta_A \circ \gamma_A = G\gamma_A \circ \gamma_A$ (= $\delta_A \circ \gamma_{GA}$ by naturality)

Whenever $\langle G, \epsilon, \delta, \gamma \rangle$ is a computational comonad, then there is another functor, $H : \mathcal{K}(G) \to \mathcal{C}$ defined by

- $HA = A$
- For $a : GA \to A$, $Ha = a \circ \gamma_A$.

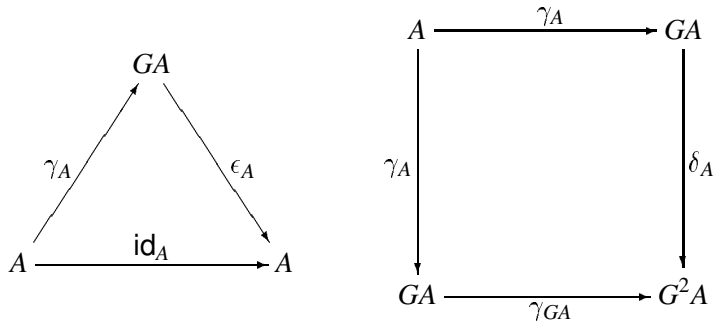

Figure 21: Properties of γ for a computational comonad

From this we can define a family of relations $\sim_{A,B}$: if $a : GA \to B$ and $f : A \to B$ then $a \sim_{A,B} f$ whenever $f = Ha = a \circ \gamma_A$ (the subscripts will be dropped when A and B are understood). We can also define two functions alg and fun as the morphism parts of the functors F and H respectively. When we use comonads to represent intensional semantic behavior, fun takes an extensional function and give a default intensional algorithm, and alg takes an algorithm and returns its extensional behavior (for further details see [7]).

All of the comonads given in the paper, if we assume one very general extra condition, are computational, with γ defined as follows:

Product comonad $\langle X, \pi_1, \delta \rangle$: Assume that there exists at least one morphism from $\mathbf{1}$ to X. Then for all objects A, and for all morphisms $x : \mathbf{1} \to X$, $\gamma_A^x = \langle \mathsf{id}_A, x \circ !_A \rangle$. For **Cont**, given $x \in X$ and for all objects A, $\gamma_A^x = \lambda a.\langle a, x \rangle$.

Exponentiation comonad $\langle V, \mathsf{val}, \mathsf{pre} \rangle$: For all objects A, let $\gamma_A = \mathsf{curry}(\pi_1)$. For **Cont**, $\gamma_A(a) = \lambda v.a$.

Strict path comonad $\langle V_S, \mathsf{sval}, \mathsf{spre} \rangle$: For all objects A, $\gamma_A = \lambda a.a^\omega$, essentially the same definition as the one for exponentiation.

8.4. Relating $\mathcal{K}(G,T)$ to $\mathcal{K}(T)$

Theorem 12 *Let $\langle G, \epsilon, \delta, \gamma \rangle$ be a computational comonad on $\mathcal{C}$, $\langle T, \eta, \mu \rangle$ be a monad on a category $\mathcal{C}$, and σ be a distributive law of T over G. If for every object A, $\sigma_A \circ \gamma_{TA} = T\gamma_A$ (figure 22) then the lifted comonad $\langle \hat{G}, \hat{\epsilon}, \hat{\delta}, \hat{\gamma} \rangle$ is also computational, where for each object A, $\hat{\gamma}_A = \eta_{GA} \circ \gamma_A$.*

Proof. By straightforward diagram chasing. Figure 23 contains the diagram proving naturality of $\hat{\gamma}$. The upper path represents the expansion of $\hat{\gamma}_B \overset{T}{\circ} f$, the lower path represents the expansion of $\hat{G}f \overset{T}{\circ} \hat{\gamma}_A$. □

From the results of the previous theorem we can construct a lifted pair of functors $(\hat{F}, \hat{H})$, and a family of relations $\hat{\sim}_{A,B}$ relating the Kleisli category $\mathcal{K}(T)$ to the doubly-lifted category $\mathcal{K}(G,T)$, as follows:

- For all objects A, $\hat{F}A = \hat{H}A = A$
- For all $f : A \to TB$, $\hat{F}f = f \overset{T}{\circ} \hat{\epsilon_A} = f \circ \epsilon_A$

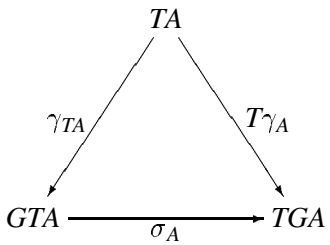

Figure 22: Condition needed to lift computational comonad

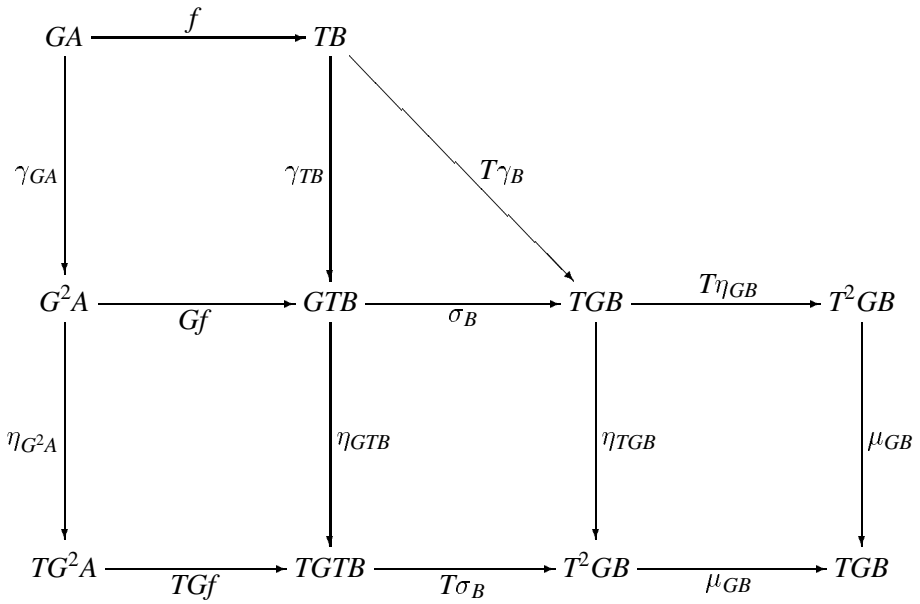

Figure 23: Proof that $\hat{\gamma}_B \overset{T}{\circ} f = \hat{G}f \overset{T}{\circ} \hat{\gamma}_A$

- For all $a : GA \to TB$, $\hat{Ha} = a \overset{G}{\circ} \hat{\gamma_A} = a \circ \gamma_A$.
- For all $a : GA \to TB$ and $f : A \to TB$, $a \hat{\sim} f$ whenever $f = \hat{Ha} = a \circ \gamma_A$.

There are also functions alg and fun as before, with $\mathsf{alg}(f) = f \circ \epsilon_A$, and $\mathsf{fun}(a) = a \circ \gamma_A$. Note that all of these relations are the same as for the standard computational comonad, but for different categories.

All the comonad/monad combinations from table 1 that have distributive laws also satisfy the condition of figure 22 except for the power set monad over the exponentiation comonad. It can easily be seen why it fails. If we let $V = \{1, 2\}$ ($e = 2$ and $m = \max$) and $A = \{a, b\}$, then

$$\begin{aligned}\sigma_A \circ \gamma_{PA}(\{a, b\}) &= \sigma_A([1 \mapsto \{a, b\}, 2 \mapsto \{a, b\}]) \\ &= \{[1 \mapsto a, 2 \mapsto a], [1 \mapsto a, 2 \mapsto b], [1 \mapsto b, 2 \mapsto b]\}\end{aligned}$$

but $P\gamma_A(\{a, b\}) = \{\gamma_A(a), \gamma_A(b)\} = \{[1 \mapsto a, 2 \mapsto a], [1 \mapsto b, 2 \mapsto b]\}$.

Note, however that the lax distributive laws $\varsigma^\dagger$ for the Hoare and Smyth power domains do satisfy the requirement of figure 22. Essentially the closure properties of the two power domains fill in the elements missing when using the plain power set.

8.5. Relating $\mathcal{K}(T)$ and $\mathcal{K}(G')$

If we look at the lifted comonad G', which did not require a distributive law, the obvious candidate for $\gamma' : A \to TGTA$ is $\gamma'_A = \eta_{GTA} \circ \gamma_{TA} \circ \eta_A$. This γ', however, is not natural. Nevertheless we can relate both $\mathcal{K}(T)$ and $\mathcal{K}(G')$ to $\mathcal{C}$ (actually, the subcategory of $\mathcal{C}$ generated by the range of T). To do this we define a pair of functors (U_T, H_G), with $U_T : \mathcal{K}(T) \to \mathcal{C}$ and $H_T : \mathcal{K}(G') \to \mathcal{C}$, defined by

- For all objects A, $U_T A = H_G A = TA$
- For all $f : A \to TB$, $U_T f = \mu_B \circ Tf$
- For all $a : GTA \to TB$, $H_T a = a \circ \gamma_{TA}$.

Note that U_T is the monad equivalent of the functor U mentioned at the beginning of this section, and H_T is the functor H restricted to objects in the range of T. Thus if we let F_T to be the functor F restricted to objects in the range of T, we then have that $H_T F_T = I$ and thus $H_T F_T U_T = U_T$.

From these functors we can define a family of relations $\sim^G_{A,B}$ as follows: given a morphism $f : A \to TB$ in the Kleisli category and a morphism $a : GTA \to TB$ in the lifted Kleisli category, $a \sim^G_{A,B} f$ if and only if $H_T a = U_T f$, i.e. if $a \circ \gamma_{TA} = \mu_B \circ Tf$

For $f : A \to TB$ and $a : GTA \to TB$, the relation $\sim^G_{A,B}$ has the following properties:

- $F_T U_T f = \mu_B \circ Tf \circ \epsilon_{TA} \sim^G f$. Thus we can define a function $\mathsf{alg}^G(f) = \mu_B \circ Tf \circ \epsilon_{TA}$ giving us a default intensional morphism for each extensional one.
- It is possible that there is no morphism f such that $a \sim^G f$. For example in the category **Set**, let $E = \{\mathrm{err}_1, \mathrm{err}_2\}$, let B and X be sets, and let $x \in X$. Then using the computational product comonad

$\langle X, \pi_1, \delta, \gamma^x \rangle$, where for $b \in B$, $\gamma^x_B(b) = \langle b, x \rangle$, and the coproduct monad $\langle E, \iota_1, \mu \rangle$, we define the function $a : (B + E) \times X \to B + E$ to be

$$a(\langle \overline{b}, x \rangle) = \begin{cases} \iota_1(b) & \overline{b} = \iota_1(b) \\ \iota_2(\mathrm{err}_2) & \overline{b} = \iota_2(e) \end{cases}$$

Then for all $f : B \to TB$, $\mu_B \circ Ef\,(\iota_2(\mathrm{err}_1)) = \langle \mathsf{id}_B, \iota_2 \rangle \circ (f + \mathsf{id}_E)(\iota_2(\mathrm{err}_1)) = \iota_2(\mathrm{err}_1)$, but $a(\gamma^x(\iota_2(\mathrm{err}_1))) = \iota_2(\mathrm{err}_2)$, so $a \not\sim^G f$.

We can find similar examples for most other non-trivial monads. In fact if for every a there exists an f such that $a \sim^G f$, then $\eta_{TA} = T\eta_A$ (to see this let $a = \eta_{TA} \circ \epsilon_{TA}$) implying that T^2A is isomorphic to TA.

- If $a \sim^G f$, then $f = a \circ \gamma_{TA} \circ \eta_A$. Thus we can define a function $\mathsf{fun}^G(a) = a \circ \gamma_{TA} \circ \eta_A$, giving us the extensional part of a. Although this function is defined for all a, from the item above we know that it may not be true that $a \sim^G \mathsf{fun}^G(a)$.
- $\mathsf{fun}^G(\mathsf{alg}^G(f)) = f$ (although it may not be true that $\mathsf{alg}^G(\mathsf{fun}^G(a)) = a$).
- For all $f' : B \to TC$ and $a' : GTB \to TC$, if $a \sim^G f$ and $a' \sim^G f'$, then $a' \overset{T}{\circ} a \sim^G f' \overset{G}{\circ} f$. This means that $\mathsf{alg}^G(f' \overset{T}{\circ} f) = \mathsf{alg}^G(f') \overset{G}{\circ} \mathsf{alg}^G(f)$, and when $a \sim^G \mathsf{fun}^G(a)$ and $a' \sim^G \mathsf{fun}^G(a')$, then $\mathsf{fun}^G(a' \overset{G}{\circ} a) = \mathsf{fun}^G(a') \overset{T}{\circ} \mathsf{fun}^G(a)$ (and also $a' \overset{G}{\circ} a \sim^G \mathsf{fun}^G(a' \overset{G}{\circ} a)$).

8.6. Relating $\mathcal{K}(T)$ and $\mathcal{K}(T')$

These two categories are not related via a Kleisli construction (actually $\mathcal{K}(T')$ is (isomorphic to) a full subcategory of $\mathcal{K}(T)$); there is thus no point in looking for a version of γ. We can, however, form a family of relations as we did in the previous section. For any pair of morphisms $f : A \to TB$ and $a : GA \to TGB$, let $a \sim^T_{A,B} f$ if and only if $a \circ \gamma_A = T\gamma_B \circ f$. Note that, unlike the last section, the constructions $T\gamma_B \circ f$ and $a \circ \gamma_A$ are not parts of functors, since the resulting morphisms are of the form $A \to TGA$ and do not compose properly to form a category.

The family of relations $\sim^T$ has similar properties to $\sim^G$: for all $f : A \to TB$ and $a : GA \to TGB$

- $T\gamma_B \circ f \circ \epsilon_A \sim^T f$, so again we can define a function $\mathsf{alg}^T(f) = T\gamma_B \circ f \circ \epsilon_A$ on morphisms in $\mathcal{K}(T)$.
- Again there are morphisms a such that there is no f with $a \sim^T f$. For example, in **Set**, let $\langle X, \pi_1, \delta, \gamma^1 \rangle$ be the product comonad from from the set $\{1, 2\}$ with $\gamma^1_A = \lambda a.\langle a, 1 \rangle$ for any set A, and let B be any set. Let the monad be the power set monad $\mathcal{P}$. Let $a = \lambda \langle b, n \rangle.\{\langle b, 2 \rangle\}$. Then for all morphisms $f : B \to \mathcal{P}B$, $\mathcal{P}\gamma^1_B \circ f\,(b) = \{\langle b', 1 \rangle \mid b' \in f\,(b)\}$, but $a \circ \gamma_B(b) = a(\langle b, 1 \rangle) = \{\langle b, 2 \rangle\}$. In order to guarantee that there is always an f such that $a \sim^G f$, we must have that $\gamma_A \circ \epsilon_A = \mathsf{id}_{GA}$, which implies that GA is isomorphic to A.
- If $a \sim^T f$ then $f = T\epsilon_B \circ a \circ \gamma_A$. Thus again we can define a function $\mathsf{fun}^T(a) = T\epsilon_B \circ a \circ \gamma_A$, and again $\mathsf{fun}^T(\mathsf{alg}^T(f)) = f$.
- For all $f' : B \to TC$ and $a' : GB \to TGC$, if $a \sim^T f$ and $a' \sim^T f'$, then $a' \overset{T}{\circ} a \sim^T f' \overset{T}{\circ} f$ (when the composition is well defined). Thus we know that $\mathsf{alg}^T(f' \overset{T}{\circ} f) = \mathsf{alg}^T(f') \overset{T}{\circ} \mathsf{alg}^T(f)$ and $\mathsf{fun}^T(a' \overset{T}{\circ} a) = \mathsf{fun}^T(a') \overset{T}{\circ} \mathsf{fun}^T(a)$ whenever a' and a have the property that $a' \sim^T \mathsf{fun}^T(a')$ and $a \sim^T \mathsf{fun}^T(a)$.

8.7. Relating the intensional and extensional semantics

Theorem 13 *For all computational comonads $\langle G, \epsilon, \delta, \gamma\rangle$ and all monads $\langle T, \eta, \mu\rangle$ that satisfy the mono-requirement, whenever $[\![p]\!]_{G,T} \sim [\![p]\!]$ for all constants p, it follows that*

$$\begin{aligned} A_\tau^{G,T} &= A_\tau, \text{ and} \\ \mathcal{M}[\![x : \tau \vdash e : \tau']\!]_{G,T} &\sim \mathcal{M}[\![x : \tau \vdash e : \tau']\!], \text{ i.e.} \\ \mathcal{M}[\![x : \tau \vdash e : \tau']\!] &= \mathcal{M}[\![x : \tau \vdash e : \tau']\!]_{G,T} \circ \gamma_{A_\tau} \end{aligned}$$

Proof. That $A_\tau^{G,T} = A_\tau$ is obvious, since they have the same definition. The rest is shown by straightforward induction over the structure of e. □

For the $\mathcal{M}[\![]\!]_{T'}$ function, we cannot simply state that $\mathcal{M}[\![x : \tau \vdash e : \tau']\!]_{T'} \sim^T \mathcal{M}[\![x : \tau \vdash e : \tau']\!]$ since the types do not match ($A_\tau^{T'} \neq A_\tau$). All we need to do, however, is add enough γ's after the evaluation of $\mathcal{M}[\![x : \tau \vdash e : \tau']\!]$ to match up the types and then we do get an equivalence. This is made precise below:

Definition 6 For all types τ, let $\Gamma[\![\tau]\!] : A_\tau \to A_\tau^{T'}$ be defined inductively by $\Gamma[\![\rho]\!] = \mathsf{id}_{A_\rho}$ and $\Gamma[\![T\tau]\!] = T\gamma_{A_\tau^{T'}} \circ T\Gamma[\![\tau]\!]$.

Theorem 14 *For all computational comonads $\langle G, \epsilon, \delta, \gamma\rangle$ and any monad $\langle T, \eta, \mu\rangle$ that satisfies the mono-requirement, $\Gamma[\![T\tau']\!] \circ \mathcal{M}[\![x : \tau \vdash e : \tau']\!] = \mathcal{M}[\![x : \tau \vdash e : \tau']\!]_{T'} \circ \gamma_{A_\tau^{T'}} \circ \Gamma[\![\tau]\!]$ whenever for all constants p, $\Gamma[\![T\mathrm{ran}(p)]\!] \circ [\![p]\!] = [\![p]\!]_{T'} \circ \gamma_{A^{T'}_{\mathrm{dom}(p)}} \circ \Gamma[\![\mathrm{dom}(p)]\!]$.*

Proof. By straightforward induction on e. □

For the third semantics, $\mathcal{M}[\![]\!]_{G'}$, the relationship $\mathcal{M}[\![x : \tau \vdash e : \tau']\!]_{G'} \sim^G \mathcal{M}[\![x : \tau \vdash e : \tau']\!]$ does not hold when $e = [e']$, except where $\eta_{TA} = T\eta_A$. Since $\mu_A \circ \eta_{TA} = \mu_A \circ T\eta_A$, with certain restrictions we can relate the two semantics:

Definition 7 For all $n \geq 0$, let $\mu_A^n : T^{n+1}A \to TA$ be defined inductively as follows: $\mu_A^0 = \mathsf{id}_{TA}$ and for $n > 0$, $\mu^{n+1} = \mu_A^n \circ \mu_{T^{n-1}A}$.

Definition 8 A morphism $f : T^{n+1}A \to T^{n+1}B$ is *simple* if there exists an $f' : A \to TB$, called a *simplification* of f, such that $\mu_B^n \circ f = \mu_B \circ Tf' \circ \mu_A^m$.

Note that for $a : GTA \to TB$, if there exists an f such that $f \sim^G a$, then $a \circ \gamma_{TA}$ is simple and f is a simplification, i.e. any algorithm that has been built from a morphism $f : A \to TB$ in a natural way will be simple. Also note that for all morphisms $f : A \to TB$, $n > 0$ and $m \geq 0$, f is a simplification of $\mu_B \circ T^n f \circ \mu_A^m$.

An example (in **Set**) of a morphism that is not simple is the set complement function $f : \mathcal{P}A \to \mathcal{P}A$, $fX = A - X$. To see this, suppose that there existed a simplification $f' : A \to \mathcal{P}A$ of f. Then $f = \bigcup_A \circ \mathcal{P}f'$, i.e. for any set $X \subseteq A$,

$$fX = \bigcup\{f'(x) \mid x \in X\} = \{a \in A \mid a \notin X\}.$$

This implies that for all $X \subseteq A$, and all $x \in X, f'(x) \notin X$. By setting $X = A$ we can easily see that no such f' can exist.

Theorem 15 *Let $\langle G, \epsilon, \delta, \gamma \rangle$ be a computational comonad and $\langle T, \eta, \mu \rangle$ be a monad over a category $\mathcal{C}$ that satisfies the mono-requirement. Suppose that for every constant p, both $\mu_{A_{\mathrm{ran}(p)}} \circ T[\![p]\!]$ and $[\![p]\!]_{G'} \circ \gamma_{TA_{\mathrm{dom}(p)}}$ are simple, with equal simplifications. Then for all $x : \tau \vdash e : \tau'$,*

$$\mu^{n+1}_{A_{\rho'}} \circ T\mathcal{M}[\![x : \tau \vdash e : \tau']\!] = \mu^{n}_{A_{\rho'}} \circ \mathcal{M}[\![x : \tau \vdash e : \tau']\!]_{G'} \circ \gamma_{T^{m+1}A^{G'}_{\rho}}$$

where $\tau = T^m\rho$ and $\tau' = T^m\rho'$ for base types ρ and ρ'

Proof. It can be shown by straightforward structural induction that for all $x : T^m\rho \vdash e : T^n\rho'$, $\mu_{T^nA_{\rho'}} \circ \mathcal{M}[\![x : T^m\rho \vdash e : T^n\rho']\!]$ and $\mathcal{M}[\![x : T^m\rho \vdash e : T^n\rho']\!]_{G'} \circ \gamma_{T^{m+1}{}_{\rho'}}$ are both simple and that $\mathcal{M}[\![x : \rho \vdash \overline{e} : \rho']\!]$ is a simplification for both, where $\overline{e}$ is formed from e by removing all μ's and []'s and converting any constants p to new constants $\overline{p}$, whose meaning is a simplification of $\mu_{A_{\mathrm{ran}(p)}} \circ T[\![p]\!]$ and $[\![p]\!]_{G'} \circ \gamma_{TA_{\mathrm{dom}(p)}}$ (by assumption they have equal simplifications). The theorem then follows directly. □

From the proof it is not difficult to see that the two semantics were related by effectively removing all explicit references in the language to the μ and the [] construct. For more monads, however, there is no need to use the [] construct to get a meaningful language. For example typical language constructs that use the power set and power domain monads, such as parallel composition and nondeterministic choice, require only simple morphisms to be meaningful, even when adding intensional behavior. Thus we still have a useful relationship between the intensional and extensional semantics as we had for the other semantic interpretations, stated in theorems 13 and 14.

9. Conclusions and Future Work

The combination of a comonad and a monad using a distributive law provides an elegant method for obtaining an intensional semantics from a monadic extensional semantics. It relates as well to the extensional monadic semantics as the standard intensional semantics does to the plain extensional semantics. Unfortunately, there are monads and comonads currently of interest in computer science that do not have distributive laws. Thus we discussed alternative ways to combine comonads and monads without a distributive law and have explored the more complex relationship between extensional and intensional interpretations obtained this way.

There are still other monads and comonads of interest to computer science that were not explored in this paper, such as the monad representing continuations. It may be that later on some combinations will be found that will be interesting enough to explore further. It should also be interesting to explore the uses and limitations of the lax distributive law of the Hoare and Smyth power domain monads over the exponentiation comonads. Also the language given in this paper was extremely simple and not particularly useful in itself; it mostly exists so we can study monads without worrying about products; most of the Kleisli categories built over monads do not have products. There are many other more complex languages that use monads and are worth studying.

References

[1] M. Barr. Composite cotriples and derived functors. In *Seminar on Triples and Categorial Homology*

Theory, Lecture notes in Mathematics. Springer-Verlag, 1969.

[2] M. Barr and C. Wells. *Toposes, Triples and Theories*. Springer-Verlag, 1985.

[3] M. Barr and C. Wells. *Category Theory for Computing Science*. Prentice-Hall International, 1990.

[4] J. Beck. Distributive laws. In *Seminar on Triples and Categorial Homology Theory*, Lecture notes in Mathematics. Springer-Verlag, 1969.

[5] G. Berry and P. L. Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20:265–321, 1982.

[6] S. Brookes and S. Geva. A cartesian closed category of parallel algorithms between Scott domains. Technical Report CMU-CS-91-159, Carnegie Mellon University, School of Computer Science, 1991. Submitted for publication.

[7] S. Brookes and S. Geva. Computational comonads and intensional semantics. In M. P. Fourman, P. T. Johnstone, and A. M. Pitts, editors, *Categories in Computer Science*, London Mathematical Society Lecture Notes, pages 1–44. Cambridge University Press, 1992.

[8] S. Brookes and S. Geva. Towards a theory of parallel algorithms on concrete data structures. *Theoretical Computer Science*, 101:177–221, July 1992.

[9] C. Gunter and D. Scott. Semantic domains. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*. MIT Press/Elsevier, 1990.

[10] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. Foundations of computing. MIT Press, 1992.

[11] Douglas J. Gurr. *Semantic Frameworks for Complexity*. PhD thesis, University of Edinburgh, January 1991.

[12] R Heckmann. Power domain constructions. *Science of Computer Programming*, 17:77–117, December 1991.

[13] S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.

[14] E. Moggi. Notions of computation and monads. *Information and Computation*, 1991.

[15] Benjamin C. Pierce. *Basic Category Theory for Computer Scientists*. MIT Press, 1991.

[16] G. D. Plotkin. A powerdomain construction. In *Society for Industrial and Applied Mathematics Journal on Computing*, volume 5, pages 452–587. Society for Industrial and Applied Mathematics, September 1976.

[17] M. B. Smyth. Power domains. *Journal of Computer and System Sciences*, 16(1):23–36, February 1978.

[18] P. Wadler. The essence of functional programming. In *Conference Record of the 19th ACM Symposium on Principles of Programming Languages*, pages 1–13. Association for Computing Machinery, 1992.