Initial Algebras of Terms, with binding and algebraic structure

Bart Jacobs[1] and Alexandra Silva[1]

Institute for Computing and Information Sciences, Radboud University Nijmegen
{bart,alexandra}@cs.ru.nl

July 15, 2013

Abstract. One of the many results which makes Joachim Lambek famous is: an initial algebra of an endofunctor is an isomorphism. This fixed point result is often referred to as "Lambek's Lemma". In this paper, we illustrate the power of initiality by exploiting it in categories of algebra-valued presheaves $\mathcal{EM}(T)^{\mathbb{N}}$, for a monad T on **Sets**. The use of presheaves to obtain certain calculi of expressions (with variable binding) was introduced by Fiore, Plotkin, and Turi. They used set-valued presheaves, whereas here the presheaves take values in a category $\mathcal{EM}(T)$ of Eilenberg-Moore algebras. This generalisation allows us to develop a theory where more structured calculi can be obtained. The use of algebras means also that we work in a linear context and need a separate operation ! for replication, for instance to describe strength for an endofunctor on $\mathcal{EM}(T)$. We apply the resulting theory to give systematic descriptions of non-trivial calculi: we introduce non-deterministic and weighted lambda terms and expressions for automata as initial algebras, and we formalise relevant equations diagrammatically.

1 Introduction

In [22] Joachim Lambek proved a basic result that is now known as "Lambek's Lemma". It says: an initial algebra $F(A) \to A$ of an endofunctor $F\colon \mathbf{C} \to \mathbf{C}$ is an isomorphism. The proof is an elegant, elementary exercise in diagrammatic reasoning. The functor F can be seen as an abstraction of a signature, describing the arities of operations. The initial algebra, if it exists, is then the algebra whose carrier is constructed inductively, using these operations for the formation of terms. This "initial algebra semantics" forms the basis of the modern perspective on expressions (terms, formulas) defined by operations, in logic and in computer science. An early reference is [11]. A map going out of an initial algebra, obtained by initiality, provides denotational semantics of expressions. By construction this semantics is "compositional", which is a way of saying that it is a homomorphism of algebras.

A more recent development is to use the dual approach, involving coalgebras $X \to G(X)$, see [28,17]; they capture the operational semantics of languages, describing the behaviour in terms of elementary steps. By duality, Lambek's Lemma also applies in this setting: a final coalgebra is an isomorphism. One of the highlights of the categorical approach to language semantics is the combined description of both denotational and operational semantics in terms of algebras and coalgebras of a functors (typically connected via distributive laws, see [19] for an overview).

This paper contains the first part of a study elaborating such a combined approach, using algebra-valued presheaves. It concentrates on obtaining structured terms as initial algebras. Historically, the first examples of initial algebras appeared in the category **Sets**, of sets and functions. But soon it became clear that initiality (or finality) in more complicated categories gives rise to a richer theory, involving additional language structure or stronger initiality principles.

- Induction with parameters (also called recursion) can be captured via initiality in "simple" slice categories, see *e.g.* [14]; similarly, initiality in (ordinary) slice categories gives a dependent form of induction, see [1].
- Expressions with variable binding operations like $\lambda x.\, xx$ in the lambda calculus can be described via initiality in presheaf categories, as shown in [8].
- This approach is extended to more complicated expressions, like in the π-calculus process language, see [10,9].
- The basics of Zermelo-Fraenkel set theory can be developed in a category with a suitably rich collection of map, see [18].
- Final coalgebras in Kleisli categories of a monad give rise to so-called trace semantics, see [12].

This paper contains an extension of the second point: the theory developed in [8] uses set-valued presheaves $\mathbb{N} \rightarrow \mathbf{Sets}$. In contrast, here we use algebra-valued presheaves $\mathbb{N} \rightarrow \mathcal{EM}(T)$, where $\mathcal{EM}(T)$ is the category of Eilenberg-Moore algebras of a monad T on **Sets**. The concrete examples elaborated at the end of the paper involve (1) the finite power set monad $\mathcal{P}_{\text{fin}}$, with algebras $\mathcal{EM}(\mathcal{P}_{\text{fin}}) = \mathbf{JSL}$, the category of join semilattices, and (2) the multiset monad $\mathcal{M}_S$ for a semiring S, with semimodules over S as algebras. The initial algebras in the category of presheaves over these Eilenberg-Moore categories have non-determinism and resource-sensitivity built into the languages, see Section 5. Moreover, in the current framework we can diagrammatically express relevant identifications, like the (β)-rule for the lambda calculus and the "trace equation" of Rabinovich [27].

The use of a category of algebras $\mathcal{EM}(T)$ instead of **Sets** requires a bit more care. The first part of the paper is devoted to extending the theory developed in [8] to this algebraic setting. The most important innovation that we need is the replication operation !, known from linear logic. Since Eilenberg-Moore categories are typically linear — with tensors $\otimes$ instead of cartesian products $\times$ — we have to use replication ! explicitly if we need non-linear features. For instance, substitution $t[s/v]$, that is, replacing all occurrences of the variable v in t by a term s, may require that we use s multiple times, namely if the variable v occurs multiple times in t. Hence the type of s must be $!\mathit{Terms}$, involving explicit replication !.

Since substitution is defined by initiality (induction) with parameters, like in [8], we need to adapt this approach. Specifically, we need to use replication ! in the so-called "strength" operation of a functor. It leads to a strength map of the form:

$$F(A) \otimes {!B} \xrightarrow{\;\mathit{strength}\;} F(A \otimes {!B}).$$

It turns out that such a strength map always exists, see Proposition 1.

In the end, in Section 5, we show how an initial algebra — of an endofunctor on such a rich universe as given by algebra-valued presheaves — gives rise to a very rich language in which many features are built-in and provided implicitly by the categorical infrastructure. This forms an illustration of the desirable situation where the formalism does the work for you. It shows the strength of the concepts that Joachim Lambek already worked on, almost fifty years ago.

2 Monads and their Eilenberg-Moore categories

This section recalls the basics of the theory of monads, as needed here. For more information, see *e.g.* [24,2,23,4]. A monad is a functor $T\colon \mathbf{C} \to \mathbf{C}$ together with two natural transformations: a unit $\eta\colon \mathrm{id}_{\mathbf{C}} \Rightarrow T$ and multiplication $\mu\colon T^2 \Rightarrow T$. These are required to make the following diagrams commute, for $X \in \mathbf{C}$.

$$\begin{tikzcd}
T(X) \arrow[r, "\eta_{T(X)}"] \arrow[dr, equal] & T^2(X) \arrow[d, "\mu_X"] & T(X) \arrow[l, "T(\eta_X)"'] \arrow[dl, equal] \\
 & T(X) &
\end{tikzcd}
\qquad
\begin{tikzcd}
T^3(X) \arrow[r, "\mu_{T(X)}"] \arrow[d, "T(\mu_X)"'] & T^2(X) \arrow[d, "\mu_X"] \\
T^2(X) \arrow[r, "\mu_X"'] & T(X)
\end{tikzcd}$$

A comonad on $\mathbf{C}$ is a monad on the opposite category $\mathbf{C}^{\mathrm{op}}$.

We briefly describe the examples of monads on **Sets** that we use in this paper.

- The lift monad $1 + (-)$ maps a set X to the set $1 + X$ containing an extra point, and a function $f\colon X \to Y$ to $1 + f\colon 1 + X \to 1 + Y$ given by $(1 + f)(\kappa_1(*)) = \kappa_1(*)$ and $(1 + f)(\kappa_2(x)) = \kappa_2(f(x))$. Here we write the coprojections of the coproduct as maps $\kappa_i\colon X_i \to X_1 + X_2$. The unit of the lift monad is the second injection $\eta(x) = \kappa_2(x)$ and multiplication is given by $\mu = [\kappa_1, \mathrm{id}]$.
- The powerset monad $\mathcal{P}$ maps a set X to the set $\mathcal{P}(X)$ of subsets of X, and a function $f\colon X \to Y$ to $\mathcal{P}(f)\colon \mathcal{P}(X) \to \mathcal{P}(Y)$ given by direct image. Its unit is given by singleton $\eta(x) = \{x\}$ and multiplication by union $\mu(\{X_i \in \mathcal{P}X \mid i \in I\}) = \bigcup_{i\in I} X_i$. We also use the *finite* powerset monad $\mathcal{P}_{\mathrm{fin}}$ which sends a set X to the set of finite subsets of X.
- For a semiring S, the multiset monad $\mathcal{M} = \mathcal{M}_S$ is defined on a set X as:

  $$\mathcal{M}(X) \;=\; \{\varphi\colon X \to S \mid \mathrm{supp}(\varphi) \text{ is finite }\}.$$

  This monad captures multisets $\varphi \in \mathcal{M}(X)$, where the value $\varphi(x) \in S$ gives the multiplicity of the element $x \in X$. When $S = \mathbb{N}$, this is sometimes called the bag monad. On functions $f\colon X \to Y$, $\mathcal{M}$ is defined as

  $$\mathcal{M}(f)(\varphi)(y) \;=\; \sum_{x\in f^{-1}(y)} \varphi(x).$$

  The support set of a multiset $\varphi \in \mathcal{M}(X)$ is defined as $\mathrm{supp}(\varphi) = \{x \in X \mid \varphi(x) \neq 0\}$. The finite support requirement is necessary for $\mathcal{M}$ to be a monad. The unit of $\mathcal{M}$ is given by $\eta(x) = (x \mapsto 1)$ for $x \in X$ and the multiplication by

  $$\mu(\Phi)(x) \;=\; \sum_{\varphi\in\mathrm{supp}(\Phi)} \Phi(\varphi) \cdot \varphi(x), \qquad \text{for } \Phi \in \mathcal{M}\mathcal{M}(X).$$

Here, and later in the examples, we use the notation $(x \mapsto s)$ to denote a function $\varphi \colon X \to S$ assigning s to x and 0 to all other $x' \in X$.

For an arbitrary monad $T = (T, \eta, \mu)$ we write $\mathcal{EM}(T)$ for the category of Eilenberg-Moore algebras. Its objects are maps of the form $\alpha \colon T(X) \to X$ satisfying $\alpha \circ \eta = \mathrm{id}$ and $\alpha \circ \mu = \alpha \circ T(\alpha)$. A homomorphism $(\alpha \colon T(X) \to X) \longrightarrow (\beta \colon T(Y) \to Y)$ in $\mathcal{EM}(T)$ is a map $f \colon X \to Y$ between the underlying objects satisfying $f \circ \alpha = \beta \circ T(f)$. This yields a category, with obvious forgetful functor $\mathcal{U} \colon \mathcal{EM}(T) \to \mathbf{Sets}$. This functor $\mathcal{U}$ has a left adjoint $\mathcal{F}$, mapping an object X to the free algebra $\mathcal{F}(X) = (\mu \colon T^2(X) \to T(X))$ on X.

A category of algebras inherits all limits from the underlying category. In the special case of a monad T on **Sets**, all colimits also exist in $\mathcal{EM}(T)$, by a result of Linton, see [2, §9.3, Prop. 4]. This category $\mathcal{EM}(T)$ is also symmetric monoidal closed, provided the monad T is commutative (*i.e.* monoidal), see [21,20]. We write the tensors as $\otimes$, with tensor unit I, and exponential $\multimap$. The free functor $\mathcal{F} \colon \mathbf{Sets} \to \mathcal{EM}(T)$ preserves the monoidal structure: $\mathcal{F}(X \times Y) \cong \mathcal{F}(X) \otimes \mathcal{F}(Y)$ and $\mathcal{F}(1) \cong I$, where 1 is a singleton set. As a result the set $A \multimap B$, for $A, B \in \mathcal{EM}(T)$, contains the homomorphisms $A \to B$ in $\mathcal{EM}(T)$.

The free algebra adjunction $\mathcal{F} \dashv \mathcal{U}$ induces a comonad $\mathcal{FU} \colon \mathcal{EM}(T) \to \mathcal{EM}(T)$ that we write as $!_T$, or simply as $!$. This comonad $! = \mathcal{FU}$ is relevant in the context of linear logic, see [13], whence the notation $!$; its Eilenberg-Moore coalgebras can be understood as bases [16]. This comonad $!$ on $\mathcal{EM}(T)$ also preserves the monoidal structure: $!(A \times B) \cong {!A} \otimes {!B}$ and $!1 \cong I$. Via these isomorphisms one sees that objects of the form $!A$ carry a comonoid structure $(\varepsilon, \varDelta)$ given by:

$$\begin{tikzcd}
{!A} \arrow[r, "!(\langle\rangle)"] \arrow[dr, "\varepsilon"'] & {!1} \arrow[d, "\cong"] \\
& I
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
{!A} \arrow[r, "!(\langle \mathrm{id}, \mathrm{id} \rangle)"] \arrow[dr, "\varDelta"'] & {!(A \times A)} \arrow[d, "\cong"] \\
& {!A \otimes !A.}
\end{tikzcd}
\tag{1}$$

These maps yield projection and diagonal operations for weakening and contraction. Also, they turn the functor $(-) \otimes {!A} \colon \mathcal{EM}(T) \to \mathcal{EM}(T)$ into a comonad.

Typically this $!$ is used to introduce "classical" computation in the "linear" setting $\mathcal{EM}(T)$, since (linear) algebra maps $!A \to B$ in $\mathcal{EM}(T)$ are in bijective correspondence with (ordinary) functions $\mathcal{U}(A) \to \mathcal{U}(B)$. In a linear setting a map $A \to B$ consumes/uses the input/resource A exactly once. But maps of the form $!A \to B$ can use A an arbitrary number of times.

In the sequel, we shall be using initial algebras of functors in Eilenberg-Moore categories $\mathcal{EM}(T)$. Initiality corresponds to induction. For "stronger" forms of initiality, with parameters, one needs strong functors. We recall the basic fact that each functor $H \colon \mathbf{Sets} \to \mathbf{Sets}$ is strong, via a strength map $\mathrm{st} \colon H(X) \times Y \to H(X \times Y)$, given by: $\mathrm{st}(u, y) = H(\lambda x. \langle x, y \rangle)(u)$. For instance, for the powerset functor $\mathcal{P}$ this yields: $\mathrm{st}(u, y) = \{\langle x, y \rangle \mid x \in u\}$. In this set the element y is used multiple times. In the next result, giving a strength in a linear setting, such multiple use leads to an occurrence of the comonad $!$. This new result extends the uniform set-theoretic definition of strength to an algebraic context.

Proposition 1. *Let T be a commutative monad on* **Sets**, *and $H\colon \mathcal{EM}(T) \to \mathcal{EM}(T)$ be an arbitrary functor. For algebras A, B there is a "non-linear" strength map:*

$$H(A) \otimes {!B} \xrightarrow{\quad \mathrm{st} \quad} H(A \otimes {!B}). \tag{2}$$

This strength map is natural in A and B, and makes the following unit and associativity diagrams commute.

$$\begin{tikzcd}
H(A) \otimes {!1} \arrow[r, "\mathrm{st}"] \arrow[dr, "\cong"'] & H(A \otimes {!1}) \arrow[d, "\cong"] \\
& H(A)
\end{tikzcd}$$

$$\begin{tikzcd}
(H(A) \otimes {!B}) \otimes {!C} \arrow[r, "\mathrm{st}\otimes\mathrm{id}"] \arrow[d, "\cong"'] & H(A \otimes {!B}) \otimes {!C} \arrow[r, "\mathrm{st}"] & H((A \otimes {!B}) \otimes {!C}) \arrow[d, "\cong"] \\
H(A) \otimes ({!B} \otimes {!C}) \arrow[d, "\cong"'] & & H(A \otimes ({!B} \otimes {!C})) \arrow[d, "\cong"] \\
H(A) \otimes {!(B \times C)} \arrow[rr, "\mathrm{st}"] & & H(A \otimes {!(B \times C)})
\end{tikzcd}$$

Proof. We recall from [21,20] that the tensor $\otimes$ of algebras comes with a special map $\otimes\colon A \times B \to A \otimes B$ which is bilinear (a homomorphism in each variable separately), and also universal, in the following manner: for each bilinear map $f\colon A \times B \to C$ there is a unique algebra map $\overline{f}\colon A \otimes B \to C$ with $\overline{f} \circ \otimes = f$. Thus, there is a bijective correspondence:

$$\begin{array}{c}
\text{bilinear maps } \; A \times B \longrightarrow C \\
\hline\hline
\text{algebra maps } \; A \otimes B \longrightarrow C
\end{array}$$

For the definition of strength we use the following correspondences.

$$\begin{array}{c}
H(A) \otimes {!B} \xrightarrow{\mathrm{st}} H(A \otimes {!B}) \\
\hline\hline
{!B} \longrightarrow H(A) \multimap H(A \otimes {!B}) \\
\hline\hline
\mathcal{U}(B) \longrightarrow \mathcal{U}\big(H(A) \multimap H(A \otimes {!B})\big)
\end{array}$$

We shall construct this last map, and then obtain strength st via these correspondences. For an element $b \in \mathcal{U}(B)$, applying the unit $\eta = \eta_{\mathcal{U}(B)}$ of the monad yields an element $\eta(b) \in \mathcal{U}(\mathcal{F}(\mathcal{U}(B))) = \mathcal{U}(!B)$. Since $\otimes$ is bilinear, the map $(-) \otimes \eta(b)\colon A \to A \otimes {!B}$ is a homomorphism of algebras. Applying the functor H yields another homomorphism:

$$H(A) \xrightarrow{\quad H\left((-)\otimes\eta(b)\right) \quad} H(A \otimes {!B})$$

This algebra map is an element of the set $\mathcal{U}\big(H(A) \multimap H(A \otimes {!B})\big)$. Hence we are done.

We prove naturality of strength, and leave the unit and associativity diagrams to the interested reader. For algebra maps $f\colon A \to C$ and $g\colon B \to D$ we wish to show that the following diagram commutes.

$$\begin{tikzcd}
H(A) \otimes !B \arrow[r, "\mathrm{st}"] \arrow[d, "H(f)\otimes !g"'] & H(A \otimes !B) \arrow[d, "H(f\otimes !g)"] \\
H(C) \otimes !D \arrow[r, "\mathrm{st}"'] & H(C \otimes !D)
\end{tikzcd}$$

Thus, we need to show for each $b \in \mathcal{U}(B)$ that the two algebra maps $H(A) \to H(C \otimes !D)$, obtained as:

$$H(f \otimes !g) \circ H((-) \otimes \eta(b)) \qquad \text{and} \qquad H((-) \otimes \eta(g(b))) \circ H(f)$$

are the same. But this is obvious by naturality of η. □

It is not hard to see that an algebra of the form $A \otimes !B$ is isomorphic to a copower $\mathcal{U}(B) \cdot A = \coprod_{b \in B} A$, since for every algebra C,

$$\dfrac{\dfrac{\dfrac{A \otimes !B \longrightarrow C}{\mathcal{U}(B) \longrightarrow (A \multimap C)}}{(A \longrightarrow C)_{b \in B}}}{\coprod_{b\in B} A \longrightarrow C}$$

Such tensors/copowers $A \otimes !B$ are used in [25] to model state-based computation, where the state A can be used only linearly.

Example 2. As illustration, and for future reference, we shall describe the strength map (2) explicitly for several functors $H\colon \mathcal{EM}(T) \to \mathcal{EM}(T)$.

1. If H is the identity functor, then obviously the strength map $A \otimes !B \to A \otimes !B$ is the identity.
2. If H is the constant functor K_C, sending everything to the algebra C, then the strength map is defined via the projection ε obtained via (1):

$$K_C(A) \otimes !B = C \otimes !B \xrightarrow{\mathrm{id}\otimes\varepsilon} C \otimes I \cong C = K_C(A \otimes !B).$$

3. If $H = H_1 + H_2$, with strength maps st^i for H_i we get a new strength map $\mathrm{st}\colon (H_1(A) + H_2(A)) \otimes !B \to H_1(A \otimes !B) + H_2(A \otimes !B)$ determined by:

$$\mathrm{st}(\kappa_1 s \otimes t) \;=\; \kappa_1 \mathrm{st}^1(s \otimes t) \qquad \mathrm{st}(\kappa_2 s \otimes t) \;=\; \kappa_2 \mathrm{st}^2(s \otimes t).$$

This follows since the map st is defined via Currying in:

$$!B \longrightarrow \big(H_1(A) + H_2(A)\big) \multimap \big(H_1(A \otimes !B) + H_2(A \otimes !B)\big)$$

The linear map on the right is clearly obtained via a sum + of the existing strength maps st^i.

4. Similarly, if $H = H_1 \times H_2$, then:

$$\mathrm{st}(\langle s_1, s_2\rangle \otimes t) \;=\; \langle \mathrm{st}^1(s_1 \otimes t), \mathrm{st}^2(s_2 \otimes t)\rangle.$$

5. For a tensor $H = H_1 \otimes H_2$ we need the duplication map Δ from (1) in:

$$\mathrm{st}((s_1 \otimes s_2) \otimes t) \;=\; \mathrm{st}^1(s_1 \otimes t) \otimes \mathrm{st}^2(s_2 \otimes t).$$

Diagrammatically this is:

$$\begin{tikzcd}
(H_1(A) \otimes H_2(A)) \otimes !B \arrow[d, "\mathrm{id}\otimes\Delta"] & \\
(H_1(A) \otimes H_2(A)) \otimes (!B \otimes !B) \arrow[r, "\cong"] & (H_1(A) \otimes !B) \otimes (H_2(A) \otimes !B) \arrow[d, "\mathrm{st}^1\otimes\mathrm{st}^2"'] \\
& H_1(A \otimes !B) \otimes H_2(A \otimes !B).
\end{tikzcd}$$

6. Finally we look at functors of the form $H = !H_1$. Then:

$$\mathrm{st}(s \otimes t) \;=\; !(\mathrm{st}^1((-) \otimes t))(s),$$

where ! is applied to the map $\mathrm{st}^1((-) \otimes t) \colon H_1(A) \to H_1(A \otimes !B)$.

With this strength map (2) we can now use the following strengthened formulation of initiality, involving additional parameters. This version of initiality is standard, see *e.g.* [14], except that it is formulated here in a linear setting with !. In principle, such strengthened versions of induction can be formulated more generally in terms of comonads and distributive laws, like in [32], but such generality is not needed here.

Lemma 3. *Let $H\colon \mathcal{EM}(T) \to \mathcal{EM}(T)$ be an endofunctor on the category of Eilenberg-Moore algebras of a commutative monad T on* **Sets**. *If this functor H has an initial algebra $a\colon H(A) \xrightarrow{\cong} A$, then: for each $c\colon H(C) \otimes !B \to C$ there is a unique map $h\colon A \otimes !B \to C$ in $\mathcal{EM}(T)$ in a commuting diagram:*

$$\begin{tikzcd}
H(A) \otimes !B \arrow[r, "\mathrm{id}\otimes\Delta"] \arrow[d, "a\otimes\mathrm{id}"', "\cong"] & H(A) \otimes !B \otimes !B \arrow[r, "\mathrm{st}\otimes\mathrm{id}"] & H(A \otimes !B) \otimes !B \arrow[r, "H(h)\otimes\mathrm{id}"] & H(C) \otimes !B \arrow[d, "c"] \\
A \otimes !B \arrow[rrr, "h"'] & & & C
\end{tikzcd} \tag{3}$$

Proof. Use initiality of a wrt. the algebra $H(!B \multimap C) \to (!B \multimap C)$ obtained by abstraction $\Lambda(-)$ in:

$$\begin{tikzcd}
c' = \Lambda\Big(H(!B \multimap C) \otimes !B \arrow[r, "\mathrm{id}\otimes\Delta"] & H(!B \multimap C) \otimes !B \otimes !B \arrow[d, "\mathrm{st}\otimes\mathrm{id}"] & & \\
& H((!B \multimap C) \otimes !B) \otimes !B \arrow[r, "H(\mathrm{ev})\otimes\mathrm{id}"] & H(C) \otimes !B \arrow[r, "c"] & C\Big).
\end{tikzcd}$$

□

For future use we mention the following basic result. The proof is left as exercise to the reader.

Lemma 4. *Consider a situation:*

$$\begin{tikzcd} \mathbf{A} \arrow[loop left, "H"] \arrow[r, bend left, "F"] & \mathbf{A} \arrow[l, bend left, "G"] \end{tikzcd} \qquad \textit{where } F \dashv G.$$

In the presence of an isomorphism $HF \cong FH$, if $U_X \in \mathbf{A}$ is (carries) the free H-algebra on $X \in \mathbf{A}$, then $F(U_X)$ is the free H-algebra on $F(X)$. □

2.1 Intermezzo: strength for endofunctors on Hilbert spaces

In Proposition 1 we have seen that strength maps exist in an algebraic context if we restrict ourselves to objects of the form $!B$, which, as we have seen in (1), come equipped with a comonoid structure for weakening and contraction. In a quantum context such comonoid structure is described in terms of Frobenius algebras. In [5] it shown that on a finite-dimensional Hilbert space such a Frobenius algebra structure corresponds to an orthonormal basis. The next result shows that in this situation one also define strength maps, much like in the proof of Proposition 1. (This will not be used in the sequel.)

Proposition 5. *Let $H\colon \mathbf{FdHilb} \to \mathbf{FdHilb}$ be an arbitrary endofunctor on the category of finite-dimensional Hilbert spaces and (continuous) linear maps. For a space $W \in$ $\mathbf{FdHilb}$ with orthonormal basis $B = \{b_1, \ldots, b_n\}$ there is a collection of strength maps:*

$$H(V) \otimes W \xrightarrow{\quad \mathrm{st}^B \quad} H(V \otimes W)$$

natural in V. The unit and associativity diagrams commute for these strength maps.

Proof. Each base vector $b_i \in W$ yields a (continuous) linear map $(-) \otimes b_i\colon V \to V \otimes W$, and thus by applying the functor H we get $H((-) \otimes b_i)\colon H(V) \to H(V \otimes W)$. For an arbitrary vector $w \in W$ we write $w = \sum_i w_i b_i$ and define $f(u, w) = \sum_i w_i H((-) \otimes b_i)(u) \in$ $H(V \otimes W)$. This yields a bilinear map $H(V) \times W \to H(V \otimes W)$, and thus a unique linear map $\mathrm{st}^B\colon H(V) \otimes W \to H(V \otimes W)$, with $\mathrm{st}^B(u \otimes w) = f(u, w)$. □

As already suggested above, this strength map in **FdHilb** really depends on the basis. This can be illustrated in a simple example (using Hilbert spaces over $\mathbb{C}$). Take as endofunctor $H(X) = X \otimes X$, with $V = \mathbb{C}$ and $W = \mathbb{C}^2 = \mathbb{C} \oplus \mathbb{C}$ in **FdHilb**. The strength map $H(\mathbb{C}) \otimes \mathbb{C}^2 \to H(\mathbb{C} \otimes \mathbb{C}^2)$ then amounts to a map $\mathbb{C}^2 \to \mathbb{C}^4$, using that $\mathbb{C}$ is the tensor unit. But we shall not drop this $\mathbb{C}$.

First we take the standard basis $B = \{(1, 0), (0, 1)\}$ on $\mathbb{C}^2$. The resulting strength map $\mathrm{st}^B\colon H(\mathbb{C}) \otimes \mathbb{C}^2 \to H(\mathbb{C} \otimes \mathbb{C}^2)$ is given by:

$$\begin{array}{l} \mathrm{st}^B((u \otimes v) \otimes (w_1, w_2)) \\ \quad = \; w_1 H((-) \otimes (1,0))(u \otimes v) + w_2 H((-) \otimes (0,1))(u \otimes v) \\ \quad = \; w_1((u \otimes (1,0)) \otimes (v \otimes (1,0))) + w_2((u \otimes (0,1)) \otimes (v \otimes (0,1))) \\ \quad = \; w_1(uv, 0, 0, 0) + w_2(0, 0, 0, uv) \\ \quad = \; uv(w_1, 0, 0, w_2). \end{array}$$

But if we take the basis $C = \{(1,1),(1,-1)\}$ we get a different strength map:

$$\begin{aligned}
&\mathrm{st}^C((u \otimes v) \otimes (w_1, w_2)) \\
&\quad = \mathrm{st}^C((u \otimes v) \otimes (\tfrac{w_1+w_2}{2}(1,1) + \tfrac{w_1-w_2}{2}(1,-1))) \\
&\quad = \tfrac{w_1+w_2}{2} H((-) \otimes (1,1))(u \otimes v) + \tfrac{w_1-w_2}{2} H((-) \otimes (1,-1))(u \otimes v) \\
&\quad = \tfrac{w_1+w_2}{2}((u \otimes (1,1)) \otimes (v \otimes (1,1))) + \tfrac{w_1-w_2}{2}((u \otimes (1,-1)) \otimes (v \otimes (1,-1))) \\
&\quad = \tfrac{w_1+w_2}{2}(uv, uv, uv, uv) + \tfrac{w_1-w_2}{2}(uv, -uv, -uv, uv) \\
&\quad = uv(w_1, w_2, w_2, w_1).
\end{aligned}$$

3 Functor categories and presheaves

Later on in this paper we will be using presheaf categories of the form $\mathcal{EM}(T)^{\mathbb{N}}$, where T is a monad on **Sets** and $\mathbb{N} \hookrightarrow$ **Sets** is the full subcategory with the natural numbers $n \in \mathbb{N}$ as objects, considered as n-element set $\{0, 1, \ldots, n-1\}$. In this section we collect some basic facts about such functor categories. Some of these results apply more generally.

For instance, if $\mathbf{A}$ is a (co)complete category, then so is the functor category $\mathbf{A}^{\mathbf{C}}$. Limits and colimits in this category are constructed elementwise. We write $\varDelta\colon \mathbf{A} \to \mathbf{A}^{\mathbf{C}}$ for the functor that sends an object $X \in \mathbf{A}$ to the constant functor $\varDelta(X)\colon \mathbf{C} \to \mathbf{A}$ that sends everything to X. This functor $\varDelta$ should not be confused with the natural transformation $\varDelta$ from (1). Each functor $F\colon \mathbf{A} \to \mathbf{B}$ yields a functor $F^{\mathbf{C}}\colon \mathbf{A}^{\mathbf{C}} \to \mathbf{B}^{\mathbf{C}}$, also by an elementwise construction: on objects $F^{\mathbf{C}}(P)(Y) = F(P(Y))$, and on morphisms (natural transformations): $F^{\mathbf{C}}(\sigma)_Y = F(\sigma_Y)$.

Lemma 6. *An adjunction $F \dashv G$ as on the left below yields an adjunction $F^{\mathbf{C}} \dashv G^{\mathbf{C}}$ as on the right.*

$$\begin{tikzcd}
\mathbf{A} \arrow[r, bend left, "F"] \arrow[r, phantom, "\perp"] & \mathbf{B} \arrow[l, bend left, "G"]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\mathbf{A}^{\mathbf{C}} \arrow[r, bend left, "F^{\mathbf{C}}"] \arrow[r, phantom, "\perp"] & \mathbf{B}^{\mathbf{C}} \arrow[l, bend left, "G^{\mathbf{C}}"]
\end{tikzcd}
\qquad \square$$

In the sequel we extend the notion $(-)^{\mathbf{C}}$ to bifunctors $F\colon \mathbf{A} \times \mathbf{A} \to \mathbf{A}$, yielding a new bifunctors $F^{\mathbf{C}}\colon \mathbf{A}^{\mathbf{C}} \times \mathbf{A}^{\mathbf{C}} \to \mathbf{A}^{\mathbf{C}}$, given by $F^{\mathbf{C}}(P, Q)(X) = F(P(X), Q(X))$. In particular, this yields a tensor product $\otimes^{\mathbf{C}}$ on $\mathbf{A}^{\mathbf{C}}$, assuming a tensor product $\otimes$ on $\mathbf{A}$. The tensor unit $I \in \mathbf{A}$ then yields a unit $\varDelta(I) \in \mathbf{A}^{\mathbf{C}}$.

For an endofunctor like F we write $\mathbf{Alg}(F)$ and $\mathbf{CoAlg}(F)$ for the categories of (functor) algebras and coalgebras of F.

Lemma 7. *For an endofunctor $F\colon \mathbf{A} \to \mathbf{A}$ there is an obvious functor $\varDelta\colon \mathbf{Alg}(F) \to \mathbf{Alg}(F^{\mathbf{C}})$, sending an F-algebra $a\colon F(X) \to X$ to a "constant" $F^{\mathbf{C}}$-algebra on $\varDelta(X)$. We write this algebra as $a^{\varDelta}$, which is a natural transformation $F^{\mathbf{C}}(\varDelta(X)) \Rightarrow \varDelta(X)$, with components:*

$$F^{\mathbf{C}}(\varDelta(X))(Y) = F(X) \xrightarrow{(a^{\varDelta})_Y = a} X = \varDelta(X)(Y).$$

This functor $\varDelta\colon \mathbf{Alg}(F) \to \mathbf{Alg}(F^{\mathbf{C}})$ has a left adjoint if $\mathbf{C}$ has a final object 1, and a right adjoint if there is an initial object $0 \in \mathbf{C}$.

Similarly, there is a functor $\varDelta\colon \mathbf{CoAlg}(F) \to \mathbf{CoAlg}(F^{\mathbf{C}})$ *which has a left (resp. right) adjoint in presence of a final (resp. initial) object in* $\mathbf{C}$.

Proof. Assume a final object $1 \in \mathbf{C}$. The "evaluate at 1" functor $(-)(1)\colon \mathbf{A}^{\mathbf{C}} \to \mathbf{A}$ yields a functor $\mathbf{Alg}(F^{\mathbf{C}}) \to \mathbf{Alg}(F)$. This gives a left adjoint to $\varDelta\colon \mathbf{Alg}(F) \to \mathbf{Alg}(F^{\mathbf{C}})$ since there is a bijective correspondence:

$$\begin{array}{c}
\begin{pmatrix} F^{\mathbf{C}}(P) \\ {\scriptstyle\beta}\Downarrow \\ P \end{pmatrix} \xRightarrow{\quad\varphi\quad} \begin{pmatrix} F^{\mathbf{C}}(\varDelta(X)) \\ \Downarrow{\scriptstyle a^{\varDelta}} \\ \varDelta(X) \end{pmatrix} \\ \hline\hline
\begin{pmatrix} F(P(1)) \\ {\scriptstyle\beta_1}\downarrow \\ P(1) \end{pmatrix} \xrightarrow[\quad f\quad]{} \begin{pmatrix} F(X) \\ \downarrow{\scriptstyle a} \\ X \end{pmatrix}
\end{array}$$

In this situation φ consists of a collection of maps $\varphi_Y\colon P(Y) \to X$, forming a map of algebras from β_Y to a, natural in Y—so that $\varphi_Z \circ P(g) = \varphi_Y$ for each $g\colon Y \to Z$ in $\mathbf{C}$. The correspondence can be described as follows.

– Given φ as above we take $\overline{\varphi} = \varphi_1\colon P(1) \to X$. This is clearly an algebra map $\beta_1 \to a$.
– Given $f\colon P(1) \to X$ we define a natural transformation $\overline{f}$ with components $\overline{f}_Y = f \circ P(!_Y)$, where $!_Y\colon Y \to 1$ is the unique map. It is not hard to see that $\overline{f}$ is natural and a map of algebras, as required.

One has $\overline{\overline{\varphi}} = \varphi$ and $\overline{\overline{f}} = f$. The rest of the lemma is obtained in the same manner. □

This result is useful to obtain preservation of initial algebra or final coalgebras by the functor $\varDelta$.

In a similar manner an endofunctor $F\colon \mathbf{C} \to \mathbf{C}$ gives rise to a functor $\mathbf{A}^F\colon \mathbf{A}^{\mathbf{C}} \to \mathbf{A}^{\mathbf{C}}$, via $\mathbf{A}^F(P)(Y) = P(F(Y))$. In this situation a natural transformation $\sigma\colon F \Rightarrow G$ yields another natural transformation $\mathbf{A}^{\sigma}\colon \mathbf{A}^F \Rightarrow \mathbf{A}^G$ with components $(\mathbf{A}^{\sigma})_{P,Y} = P(\sigma_Y)\colon P(F(Y)) \to P(G(Y))$. This is of interest when F is a monad or comonad.

Lemma 8. *Let* $T\colon \mathbf{C} \to \mathbf{C}$ *be a (co)monad. Then so is* $\mathbf{A}^T\colon \mathbf{A}^{\mathbf{C}} \to \mathbf{A}^{\mathbf{C}}$.

Proof. In the obvious way: for instance if $T = (T, \eta, \mu)$ is a monad, then $\mathbf{A}^T$ is also a monad, where $\eta^T_P = \mathbf{A}^{\eta}_P\colon P \Rightarrow \mathbf{A}^T(P)$ and $\mu^T_P = \mathbf{A}^{\mu}_P\colon (\mathbf{A}^T)^2(P) = \mathbf{A}^{T^2}(P) \Rightarrow \mathbf{A}^T(P)$ have components:

$$\begin{tikzcd} P(Y) \arrow[r, "P(\eta_Y)"] & P(T(Y)) & P(T^2(Y)). \arrow[l, "P(\mu_Y)"'] \end{tikzcd}$$ □

We now restrict to functor categories of the form $\mathbf{A}^{\mathbb{N}}$, where $\mathbb{N} \hookrightarrow \mathbf{Sets}$ is the full subcategory with natural numbers $n = \{0, 1, \ldots, n-1\}$ as objects. We shall introduce a "weakening" monad $\mathcal{W}$ on this category, via the previous lemma. This $\mathcal{W}$ comes from [8] where it is written as δ and where it is called context extension. But it is called differentiation in [10] and dynamic allocation in [9]. Here we prefer to call it weakening to emphasise this aspect of context extension. We write it as $\mathcal{W}$ and not as δ to avoid a conflict with the comultiplication operation of the comonad !. Before we proceed we recall that in the category $\mathbb{N}$ the number 0 is initial, and the number $n + m$ is the coproduct of $n, m \in \mathbb{N}$. In any category with coproducts, the functor $(-) + X$ is a monad. We apply this to the category $\mathbb{N}$ and get the lift monad $(-) + 1$. The previous lemma now gives the following result.

Lemma 9. *For an arbitrary category* $\mathbf{A}$ *define a monad* $\mathcal{W} = \mathbf{A}^{(-)+1} \colon \mathbf{A}^{\mathbb{N}} \to \mathbf{A}^{\mathbb{N}}$, *so that:*

$$\mathcal{W}(P)(n) \;=\; P(n+1) \qquad \mathcal{W}(P)(f) \;=\; P(f + \mathrm{id}_1) \qquad \mathcal{W}(\sigma)_n \;=\; \sigma_{n+1}.$$

The unit $\mathrm{up} \colon \mathrm{id} \Rightarrow \mathcal{W}$ *and multiplication* $\mathrm{ctt} \colon \mathcal{W}^2 \Rightarrow \mathcal{W}$ *of this monad have components* $\mathrm{up}_P \colon P \Rightarrow \mathcal{W}(P)$ *and* $\mathrm{ctt}_P \colon \mathcal{W}(\mathcal{W}(P)) \to \mathcal{W}(P)$ *with:*

$$\begin{tikzcd} P(n) \arrow[r, "\mathrm{up}_{P,n} = P(\kappa_1)"] & P(n+1) & P(n+2). \arrow[l, "\mathrm{ctt}_{P,n} = P([\mathrm{id}, \kappa_2])"'] \end{tikzcd} \qquad \square$$

The map up sends an item in $P(n)$ to $P(n + 1)$, where the context $n + 1$ has an additional variable v_{n+1}. The abbreviation ctt stands for "contract"; it can be understood as substitution $[v_{n+1}/v_{n+2}]$, removing the last variable. There is also a "swap" map $\mathrm{swp} \colon \mathcal{W}^2 \Rightarrow \mathcal{W}^2$ defined, following [8], as:

$$\begin{tikzcd} \mathcal{W}^2(P)(n) = P(n+2) \arrow[r, "\mathrm{swp}_{P,n} = P(\mathrm{id} + [\kappa_2, \kappa_1])"] & P(n+2) = \mathcal{W}^2(P)(n). \end{tikzcd} \tag{4}$$

This swap map can be understood as simultaneous substitution $[v_{n+1}/v_{n+2}, v_{n+2}/v_{n+1}]$.

The following trivial observation will be useful later on (combined with Lemma 4). It includes commutation with (co)limits, which are given in a pointwise manner in functor categories.

Lemma 10. *The weakening functor* $\mathcal{W} \colon \mathbf{A}^{\mathbb{N}} \to \mathbf{A}^{\mathbb{N}}$ *commutes with functors* $F^{\mathbb{N}}$ *defined pointwise:* $\mathcal{W}F^{\mathbb{N}} = F^{\mathbb{N}}\mathcal{W}$. *This also holds when* F *is a bifunctor, covering products* $\times$, *tensors* $\otimes$ *and coproducts* $+$. □

In [8] it is shown, for the special case where $\mathbf{A} = \mathbf{Sets}$, that this functor $\mathcal{W}$ has both a left and a right adjoint. In our more general situation basically the same constructions can be used, but they require more care. We will describe left and right adjoints separately.

Lemma 11. *Assuming the category* $\mathbf{A}$ *has finite copowers* $n \cdot X = X + \cdots + X$ *(n times), the weakening functor* $\mathcal{W} \colon \mathbf{A}^{\mathbb{N}} \to \mathbf{A}^{\mathbb{N}}$ *has a left adjoint, given by* $Q \mapsto (-) \cdot Q(-)$; *this right-hand-side is the functor* $\mathbb{N} \to \mathbf{A}$ *given by* $n \mapsto n \cdot Q(n)$.

Proof. There is a bijective correspondence between components of natural transformations:

$$\frac{Q(n) \overset{\sigma_n}{\Longrightarrow} \mathcal{W}(P)(n) = P(n+1)}{\overline{\;n \cdot Q(n) \underset{\tau_n}{\Longrightarrow} P(n)\;}}$$

Given σ_n we take the cotuple:

$$\overline{\sigma}_n = \Big(n \cdot Q(n) \xrightarrow{\big[P([\mathrm{id}, i]) \circ \sigma_n\big]_{i \in n}} P(n)\Big).$$

In the reverse direction, given τ, we take:

$$\overline{\tau}_n = \Big(Q(n) \xrightarrow{Q(\kappa_1)} Q(n+1) \xrightarrow{\kappa_2} (n+1) \cdot Q(n+1) \xrightarrow{\tau_{n+1}} P(n+1)\Big).$$

It is not hard see that these $\overline{(-)}$ constructions yield natural transformations and are each other's inverses. □

Remark 12. In the sequel we typically use $\mathbf{A} = \mathcal{EM}(T)$, the Eilenberg-Moore category of a monad T on **Sets**. If we assume that the monad T is commutative, then, the category $\mathcal{EM}(T)$ is symmetric monodial closed (see [21,20]), with tensor unit $I = \mathcal{F}(1)$, where $\mathcal{F}$ is the free monad functor and 1 is the terminal object in the underlying category. In that case we can reorganise the left adjoint from the previous lemma, since:

$$\begin{array}{rcll} n \cdot Q(n) & \cong & n \cdot (I \otimes Q(n)) & \\ & \cong & (n \cdot I) \otimes Q(n) & \text{using that } (-) \otimes X \text{ is a left adjoint} \\ & \cong & (n \cdot \mathcal{F}(1)) \otimes Q(n) & \\ & \cong & \mathcal{F}(n \cdot 1) \otimes Q(n) & \text{since free functors preserve coproducts} \\ & \cong & \mathcal{F}(n) \otimes Q(n). & \end{array}$$

In this Eilenberg-Moore situation we can thus describe the adjunction from the previous lemma as $\mathcal{F} \otimes^{\mathbb{N}} (-) \dashv \mathcal{W}$, where $\otimes^{\mathbb{N}}$ is the pointwise tensor on $\mathcal{EM}(T)^{\mathbb{N}}$, and where $\mathcal{F} \in \mathcal{EM}(T)^{\mathbb{N}}$ is the restriction of the free functor $\mathcal{F} \colon \mathbf{Sets} \to \mathcal{EM}(T)$ to $\mathbb{N} \hookrightarrow \mathbf{Sets}$. It corresponds to the free variable functor V from [8] — and will be used as such later on.

The right adjoint to weakening is more complicated. In the set-theoretic setting of [8] it is formulated in terms of sets of natural transformations. In the current more general context this construction has to be done internally, via a suitable equaliser.

Lemma 13. *If a category* $\mathbf{A}$ *is complete, then the weakening functor* $\mathcal{W} \colon \mathbf{A}^{\mathbb{N}} \to \mathbf{A}^{\mathbb{N}}$ *has a right adjoint.*

Proof. For a presheaf $Q \in \mathbf{A}^{\mathbb{N}}$ we define a new presheaf $\mathcal{R}(Q) \in \mathbf{A}^{\mathbb{N}}$, where $\mathcal{R}(Q)(n)$ is obtained as equaliser in $\mathbf{A}$:

$$\begin{tikzcd} \mathcal{R}(Q)(n) \arrow[r, tail, "e_n"] & \prod\limits_{m \in \mathbb{N}} Q(m)^{((m+1)^n)} \arrow[r, shift left, "\varphi"] \arrow[r, shift right, "\psi"'] & \prod\limits_{g \colon m \to m'} Q(m')^{((m+1)^n)} \end{tikzcd}$$

where the parallel maps φ, ψ are defined by:

$$\begin{array}{rcl} \varphi & = & \langle\, \langle\, Q(g + \mathrm{id}) \circ \pi_f \circ \pi_m \,\rangle_{f \colon n \to m+1} \,\rangle_{g \colon m \to m'} \\ \psi & = & \langle\, \langle\, \pi_{(g+\mathrm{id}) \circ f} \circ \pi_{m'} \,\rangle_{f \colon n \to m+1} \,\rangle_{g \colon m \to m'} . \end{array}$$

We sketch the adjunction $\mathcal{W} \dashv \mathcal{R}$. For a natural transformation $\sigma\colon \mathcal{W}(P) \Rightarrow Q$ we have maps $\sigma_n\colon P(n+1) \to Q(n)$ in $\mathbf{A}$, and define:

$$\sigma'_n \;=\; \langle\, \langle\, \sigma_m \circ P(f)\,\rangle_{f\colon n\to m+1}\,\rangle_{m\in\mathbb{N}} \;:\; P(n) \longrightarrow \prod_{m\in\mathbb{N}} Q(m)^{((m+1)^n)}$$

We leave it to the reader to verify that $\varphi \circ \sigma'_n = \psi \circ \sigma'_n$. This equation yields a unique map $\overline{\sigma}_n\colon P(n) \to \mathcal{R}(Q)(n)$.

Conversely, given $\tau\colon P \Rightarrow \mathcal{R}(Q)$ we get $e_n \circ \tau_n\colon P(n) \to \prod_{m\in\mathbb{N}} Q(m)^{((m+1)^n)}$. Hence we take:

$$\overline{\tau}_n \;=\; \pi_{\mathrm{id}_{n+1}} \circ \pi_n \circ e_{n+1} \circ \tau_{n+1} \;:\; P(n+1) \longrightarrow Q(n).$$

Remaining details are left to the reader. □

Along the same lines of the previous result one obtains the following standard result.

Lemma 14. *Let $(I, \otimes, \multimap)$ be the monoidal closed structure of a complete category $\mathbf{A}$. The pointwise monoidal structure $(\varDelta(I), \otimes^{\mathbb{N}})$ on the functor category $\mathbf{A}^{\mathbb{N}}$ is then also closed.* □

When $\mathbf{A} = \mathcal{EM}(T)$ like in Remark 12, we can thus write weakening as exponent $\mathcal{W} \cong [\mathcal{F}, -]$, using the adjunction $\mathcal{F} \otimes^{\mathbb{N}} (-) \dashv \mathcal{W}$.

What we still need is the following.

Lemma 15. *For a commutative monad T on* **Sets**, *the weakening monad $\mathcal{W}\colon \mathcal{EM}(T)^{\mathbb{N}} \to \mathcal{EM}(T)^{\mathbb{N}}$ is a strong monad wrt. the pointwise monoidal structure $(\varDelta(I), \otimes^{N})$. The strength map is given by:*

$$\mathcal{W}(P) \otimes^{\mathbb{N}} Q \xrightarrow{\;\mathrm{st}=\mathrm{id}\otimes\mathrm{up}\;} \mathcal{W}(P \otimes^{\mathbb{N}} Q).$$ □

4 Substitution

By now we have collected enough material to transfer the definition of substitution used in [8] for set-valued presheaves to the algebra-valued presheaves used in the present setting. Here we rely on Lemma 4, where in [8] a uniformity principle of fixed points is used. Alternatively, free constructions can be used. But we prefer the rather concrete approach followed below in order to have a good handle on substitution.

Proposition 16. *Let T be a commutative monad on* **Sets**, *and $H\colon \mathcal{EM}(T)^{\mathbb{N}} \to \mathcal{EM}(T)^{\mathbb{N}}$ be a strong functor with:*

- *an isomorphism $\phi\colon H\mathcal{W} \overset{\cong}{\Rightarrow} \mathcal{W}H$;*
- *for each $P \in \mathcal{EM}(T)^{\mathbb{N}}$ a free H-algebra $H^*(P) \in \mathcal{EM}(T)^{\mathbb{N}}$ on P. This $H^*(P)$ can equivalently be described as initial algebra of the functor $P + H(-)\colon \mathcal{EM}(T)^{\mathbb{N}} \to \mathcal{EM}(T)^{\mathbb{N}}$ in:*

$$P + H(H^*(P)) \xrightarrow[\cong]{[\eta_P, \theta_P]} H^*(P) \tag{5}$$

In this situation one can define a substitution map:

$$\mathcal{W}H^*(\mathcal{F}) \otimes !H^*(\mathcal{F}) \xrightarrow{\quad\text{sbs}\quad} H^*(\mathcal{F}),$$

where $\mathcal{F} \in \mathcal{EM}(T)^{\mathbb{N}}$ *is the "free variable" presheaf given by restricting the free functor* $\mathcal{F}\colon \mathbf{Sets} \to \mathcal{EM}(T)$.

One may read $\text{sbs}_n(s \otimes U) = s[U/v_{n+1}]$ where U is of !-type. The type $\mathcal{W}H^*(\mathcal{F}) \otimes !H^*(\mathcal{F}) \to H^*(\mathcal{F})$ of the substitution map sbs is very rich and informative:

- The first argument s of type $\mathcal{W}H^*(\mathcal{F})$ describes the term s in an augmented context; hence there is a variable v_{n+1} in which substitution is going to happen;
- The second argument U of replication type $!H^*(\mathcal{F})$ is going to be substituted for the variable v_{n+1}. Since this variable may occur multiple times (zero or more) in s, we need to be able to use U multiple times. Hence the replication comonad ! is needed in its type. It does not occur in the set-theoretic (cartesian) setting used in [8].

Proof. By Lemma 4 we know that $\mathcal{W}H^*(\mathcal{F})$ is the free H-algebra on $\mathcal{W}(\mathcal{F})$, and thus an initial algebra of the functor $\mathcal{W}(\mathcal{F}) + H(-)\colon \mathcal{EM}(T)^{\mathbb{N}} \to \mathcal{EM}(T)^{\mathbb{N}}$, via:

$$\mathcal{W}(\mathcal{F}) + H(\mathcal{W}H^*(\mathcal{F})) \underset{\cong}{\xrightarrow{\text{id}+\phi}} \mathcal{W}(\mathcal{F}) + \mathcal{W}H(H^*(\mathcal{F})) = \mathcal{W}\big(\mathcal{F} + H(H^*(\mathcal{F}))\big) \underset{\cong}{\xrightarrow{\mathcal{W}([\eta_{\mathcal{F}},\theta_{\mathcal{F}}])}} \mathcal{W}H^*(\mathcal{F}).$$

The strong version of induction described in Lemma 3 implies that it suffices to produce a map of the form:

$$\big(\mathcal{W}(\mathcal{F}) + H(H^*(\mathcal{F}))\big) \otimes !H^*(\mathcal{F}) \xrightarrow{\qquad\qquad} H^*(\mathcal{F}).$$

so that the substitution map $\text{sbs}\colon \mathcal{W}H^*(\mathcal{F}) \otimes !H^*(\mathcal{F}) \to H^*(\mathcal{F})$ arises like in Diagram (3). Tensors $\otimes$ distribute over coproducts $+$, since $\mathcal{EM}(T)^{\mathbb{N}}$ is monoidal closed by Lemma 14, and so it suffices to produce two maps:

$$\mathcal{W}(\mathcal{F}) \otimes !H^*(\mathcal{F}) \longrightarrow H^*(\mathcal{F}) \qquad\qquad H(H^*(\mathcal{F})) \otimes !H^*(\mathcal{F}) \longrightarrow H^*(\mathcal{F}). \tag{6}$$

We get the second of these maps simply via projection and the initial algebra in (5):

$$H(H^*(\mathcal{F})) \otimes !H^*(\mathcal{F}) \longrightarrow H(H^*(\mathcal{F})) \xrightarrow{\theta_{\mathcal{F}}} H^*(\mathcal{F}).$$

For the first map in (6) we start with the isomorphisms:

$$\mathcal{W}(\mathcal{F})(n) \;=\; \mathcal{F}(n+1) \;\cong\; \mathcal{F}(n) + \mathcal{F}(1) \;\cong\; \mathcal{F}(n) + I \;=\; (\mathcal{F} + \varDelta(I))(n).$$

Again using that $\otimes$ distributes over $+$, and that $\varDelta(I)$ is the tensor unit in $\mathcal{EM}(T)^{\mathbb{N}}$ we see that the first map in (6) amounts to two maps:

$$\mathcal{F} \otimes !H^*(\mathcal{F}) \longrightarrow H^*(\mathcal{F}) \qquad\qquad !H^*(\mathcal{F}) \longrightarrow H^*(\mathcal{F}).$$

In the second case we recall that ! is a comonad, to that there is a counit $\varepsilon\colon !H^*(\mathcal{F}) \to H^*(\mathcal{F})$, and in the first case we project and use the unit η from (5):

$$\mathcal{F} \otimes !H^*(\mathcal{F}) \longrightarrow \mathcal{F} \xrightarrow{\eta_{\mathcal{F}}} H^*(\mathcal{F}). \qquad \square$$

In [8] it is shown that the substitution map satisfies certain equations. They are not of immediate relevance here.

5 Examples

We apply the framework developed in the previous sections to describe some term calculi as initial algebras in a category of algebra-valued presheaves $\mathcal{EM}(T)^{\mathbb{N}}$. For convenience, we no longer use explicit notation for the pointwise functors like $\otimes^{\mathbb{N}}$ or $!^{\mathbb{N}}$ on $\mathcal{EM}(T)^{\mathbb{N}}$ and simply write them as $\otimes$ and $!$. Hopefully this does not lead to (too much) confusion.

5.1 Non-deterministic lambda calculus

We start with the finite powerset monad $T = \mathcal{P}_{\mathrm{fin}}$, so that $\mathcal{EM}(\mathcal{P}_{\mathrm{fin}})$ is the category **JSL** of join semilattices and maps preserving $(\bot, \vee)$. Let $\Lambda \in \mathbf{JSL}^{\mathbb{N}}$ be the initial algebra of the endofunctor on $\mathcal{EM}(T)^{\mathbb{N}} = \mathbf{JSL}^{\mathbb{N}}$, given by:

$$P \longmapsto \mathcal{F} + \mathcal{W}(P) + (P \otimes !P),$$

where $\mathcal{F}(n) = \mathcal{P}_{\mathrm{fin}}(n)$ and $!P(n) = \mathcal{P}_{\mathrm{fin}}(P(n))$. Thus, Λ is the free algebra on $\mathcal{F}$ — written as $H^*(\mathcal{F})$ in Proposition 16 — for the functor $H(P) = \mathcal{W}(P) + (P \otimes !P)$.

We describe this initial algebra as a co-triple of (join-preserving) maps:

$$\mathcal{F} + \mathcal{W}(\Lambda) + (\Lambda \otimes !\Lambda) \xrightarrow[\cong]{[\mathrm{var},\mathrm{lam},\mathrm{app}]} \Lambda$$

Elements of the set of terms $\Lambda(n) \in \mathbf{JSL}$ with variables from $\{v_1, \ldots, v_n\}$ are inductively given by:

- $\mathrm{var}_n(V)$, where $V \in \mathcal{F}(n) = \mathcal{P}_{\mathrm{fin}}(n) = \mathcal{P}_{\mathrm{fin}}(\{v_1, v_2, \ldots, v_n\})$;
- $\mathrm{lam}_n(N) = \lambda v_{n+1}.\, N$, where $N \in \mathcal{W}(\Lambda)(n) = \Lambda(n+1)$;
- $\mathrm{app}(M, \{N_1, \ldots, N_k\}) = M \cdot \{N_1, \ldots, N_k\}$, where $M, N_1, \ldots, N_k \in \Lambda(n)$;
- $\bot \in \Lambda(n)$, and $M \vee N \in \Lambda(n)$, for $M, N \in \Lambda(n)$.

The join-preservation property that holds by construction for these maps yields:

$$\begin{array}{rclrcl}
\mathrm{var}_n(\emptyset) & = & \bot & \mathrm{var}_n(V \cup V') & = & \mathrm{var}_n(V) \vee \mathrm{var}_n(V') \\
\lambda v_{n+1}.\, \bot & = & \bot & (M \vee M') \cdot U & = & (M \cdot U) \vee (M' \cdot U) \\
\lambda v_{n+1}.\, (M \vee N) & = & (\lambda v_{n+1}.\, M) \vee (\lambda v_{n+1}.\, N) & M \cdot (U \cup U') & = & (M \cdot U) \vee (M \cdot U') \\
\bot \cdot U & = & \bot & M \cdot \emptyset & = & \bot.
\end{array}$$

The non-standard features of this term calculus are the occurrences of *sets* of variables V in $\mathrm{var}_n(V)$ and of *sets* of terms as second argument in application $M \cdot \{N_1, \ldots, N_k\}$. But using these join preservation properties they can be described also in terms of single variables/terms:

$$\begin{array}{rcl}
\mathrm{var}_n(\{v_1, \ldots, v_k\}) & = & \mathrm{var}_n(\{v_1\}) \vee \cdots \vee \mathrm{var}_n(\{v_k\}) \\
M \cdot \{N_1, \ldots, N_k\} & = & M \cdot \{N_1\} \vee \cdots \vee M \cdot \{N_k\}.
\end{array}$$

Following Proposition 16 — recall that $\Lambda = H^*(\mathcal{F})$ — there is a substitution map sbs in $\mathbf{JSL}^{\mathbb{N}}$:

$$\mathcal{W}(\Lambda) \otimes !\Lambda \xrightarrow{\quad \text{sbs} \quad} \Lambda.$$

The diagram that defines sbs, according to Lemma 3, can be split up in three separate diagrams, for variables, abstraction and application.

$$\begin{tikzcd}
\mathcal{W}(\mathcal{F}) \otimes !\Lambda \arrow[r, "\cong"] \arrow[d, "\mathcal{W}(\text{var})\otimes\text{id}"', "\cong"] & \mathcal{F} \otimes !\Lambda + !\Lambda \arrow[d, "{[\text{var}\circ\pi_1, \bigvee]}"] \\
\mathcal{W}(\Lambda) \otimes !\Lambda \arrow[r, "\text{sbs}"] & \Lambda
\end{tikzcd} \tag{7}$$

$$\begin{tikzcd}
\mathcal{W}^2(\Lambda) \otimes !\Lambda \arrow[r, "\text{id}\otimes\text{up}"] \arrow[d, "\text{swp}\otimes\text{id}"', "\cong"] & \mathcal{W}^2(\Lambda) \otimes \mathcal{W}(!\Lambda) \arrow[r, equal] & \mathcal{W}(\mathcal{W}(\Lambda) \otimes !\Lambda) \arrow[r, "\mathcal{W}(\text{sbs})"] & \mathcal{W}(\Lambda) \arrow[dd, "\text{lam}"] \\
\mathcal{W}^2(\Lambda) \otimes !\Lambda \arrow[d, "\mathcal{W}(\text{lam})\otimes\text{id}"'] & & & \\
\mathcal{W}(\Lambda) \otimes !\Lambda \arrow[rrr, "\text{sbs}"] & & & \Lambda
\end{tikzcd} \tag{8}$$

$$\begin{tikzcd}
(\mathcal{W}(\Lambda) \otimes !\mathcal{W}(\Lambda)) \otimes !\Lambda \arrow[r, "\text{st}"] \arrow[d, equal] & (\mathcal{W}(\Lambda) \otimes !\Lambda) \otimes !(\mathcal{W}(\Lambda) \otimes !\Lambda) \arrow[r, "\text{sbs}\otimes!(\text{sbs})"] & \Lambda \otimes !\Lambda \arrow[dd, "\text{app}"] \\
\mathcal{W}(\Lambda \otimes !\Lambda) \otimes !\Lambda \arrow[d, "\mathcal{W}(\text{app})\otimes\text{id}"'] & & \\
\mathcal{W}(\Lambda) \otimes !\Lambda \arrow[rr, "\text{sbs}"] & & \Lambda
\end{tikzcd} \tag{9}$$

In more conventional notation, reading $\text{sbs}_n(M \otimes U) = M[U/v_{n+1}]$ where $M \in \Lambda$ and $U \subseteq \Lambda$ is finite, we can write these three diagrams (7) – (9) as:

$$\begin{array}{rcl}
\text{var}_{n+1}(V)[U/v_{n+1}] & = & \begin{cases} \text{var}_n(V) & \text{if } v_{n+1} \notin V \\ \text{var}_n(V - v_{n+1}) \vee \bigvee U & \text{if } v_{n+1} \in V \end{cases} \\
(\lambda v_{n+2}.\, M)[U/v_{n+1}] & = & \lambda v_{n+1}.\, M[v_{n+1}/v_{n+2}, v_{n+2}/v_{n+1}][U/v_{n+2}] \\
& = & \lambda v_{n+1}.\, M[v_{n+1}/v_{n+2}, U/v_{n+1}] \\
(M \cdot \{N_1, \ldots, N_k\})[U/v_{n+1}] & = & M[U/v_{n+1}] \cdot \{N_1[U/v_{n+1}], \ldots, N_k[U/v_{n+1}]\}.
\end{array}$$

By construction the substitution map sbs is bilinear, so that:

$$\begin{array}{rclcrcl}
\bot[U/v_{n+1}] & = & \bot & \qquad & (M \vee M')[U/v_{n+1}] & = & (M[U/v_{n+1}]) \vee (M'[U/v_{n+1}]) \\
M[\emptyset/v_{n+1}] & = & \bot & \qquad & M[(U \cup U')/v_{n+1}] & = & (M[U/v_{n+1}]) \vee (M[U'/v_{n+1}]).
\end{array}$$

We see that the mere initiality of Λ in the presheaf category $\mathcal{EM}(\mathcal{P}_{\text{fin}})^{\mathbb{N}}$ gives a lot of information about the term calculus involved. As usual, (initial) algebras of functors do not involve any equations. If needed, they will have to be imposed explicitly. For instance, in the current setting it makes sense to require the analogue of the familiar

(β)-rule $(\lambda x.\, M)N = M[N/x]$ for (ordinary) lambda terms. Here it can be expressed diagrammatically as:

$$\begin{tikzcd}
\mathcal{W}(\Lambda) \otimes {!\Lambda} \arrow[r, "\mathrm{lam} \otimes \mathrm{id}"] \arrow[rd, "\mathrm{sbs}"'] & \Lambda \otimes {!\Lambda} \arrow[d, "\mathrm{app}"] \\
& \Lambda
\end{tikzcd}$$

Another example is the (η)-rule: $\lambda x.\, Mx = M$, if x is not a free variable in M. In this setting it can be expressed as

$$\begin{tikzcd}
\Lambda \otimes I \arrow[r, "\mathrm{up} \otimes \mathrm{new}"] \arrow[rrdd, "\cong"'] & \mathcal{W}(\Lambda) \otimes \mathcal{W}(!\Lambda) \arrow[r, equal] & \mathcal{W}(\Lambda \otimes {!\Lambda}) \arrow[d, "\mathcal{W}(\mathrm{app})"] \\
& & \mathcal{W}(\Lambda) \arrow[d, "\mathrm{lam}"] \\
& & \Lambda
\end{tikzcd}$$

Here, $\mathrm{new}\colon I \to \mathcal{W}(!\Lambda)$ denotes the generation of a free variable. It is defined as:

$$\mathrm{new} = \Big(I = {!1} \xrightarrow{\delta} {!!1} = {!I} \xrightarrow{!\kappa_2} {!(\mathcal{F} + I)} \cong {!\mathcal{W}(\mathcal{F})} \xrightarrow{!\mathcal{W}(\mathrm{var})} {!\mathcal{W}(\Lambda)} = \mathcal{W}(!\Lambda)\Big).$$

5.2 Weighted lambda calculus

We now consider a weighted version of the lambda calculus, by changing the monad in the above example. We fix a semiring S and consider the associated multiset monad $\mathcal{M} = \mathcal{M}_S$. Its Eilenberg-Moore algebras are semimodules over S. We write $\mathbf{SMod} = \mathcal{EM}(\mathcal{M})$, with morphism given by linear maps. Let $\Lambda \in \mathbf{SMod}^{\mathbb{N}}$ be the initial algebra of the endofunctor on $\mathcal{EM}(\mathcal{M})^{\mathbb{N}} = \mathbf{SMod}^{\mathbb{N}}$ given by:

$$P \longmapsto \mathcal{F} + \mathcal{W}(P) + (P \otimes {!P}),$$

where $\mathcal{F}(n) = \mathcal{M}(n)$ and $!P(n) = \mathcal{M}(P(n))$. Notice that this functor is formally the same as for the non-deterministic lambda calculus (in the previous subsection).

As before, we describe this initial algebra as a co-triple of (linear) maps:

$$\mathcal{F} + \mathcal{W}(\Lambda) + (\Lambda \otimes {!\Lambda}) \xrightarrow[\cong]{[\mathrm{var},\mathrm{lam},\mathrm{app}]} \Lambda$$

Elements of the set of terms $\Lambda(n) \in \mathbf{SMod}$ with variables from $\{v_1, \ldots, v_n\}$ are inductively given by:

- $\mathrm{var}_n(\varphi)$, where $\varphi \in \mathcal{F}(n) = \mathcal{M}(n) = \mathcal{M}(\{v_1, v_2, \ldots, v_n\})$; we will typically write φ (and, in general, elements of $\mathcal{M}(X)$) as a (finite) formal sum $\sum_i s_i v_i$.
- $\mathrm{lam}_n(N) = \lambda v_{n+1}.\, N$, where $N \in \Lambda(n+1)$;
- $\mathrm{app}(M, \sum_i s_i N_i) = M \cdot (\sum_i s_i N_i)$, where $M, N_i \in \Lambda(n)$;
- $0 \in \Lambda(n)$, and $s \bullet N \in \Lambda(n)$, for $N \in \Lambda(n)$ and $s \in S$, and $N + M \in \Lambda(n)$ for $N, M \in \Lambda(n)$. We write a fat dot $\bullet$ for scalar multiplication in order to distinguish it from application.

Slightly generalising the previous example, the non-standard features of this term calculus are the occurrences of *linear combinations* of variables φ in $\mathrm{var}_n(\varphi)$ and of *linear combinations* of terms as second argument in application $M \cdot (\sum_i s_i N_i)$. But using the linearity properties of the maps, they can be described also in terms of single variables/terms:

$$\begin{array}{rcl} \mathrm{var}_n(\sum_i s_i v_i) & = & \sum_i s_i \bullet \mathrm{var}_n(1v_i) \\ M \cdot (\sum_i s_i N_i) & = & \sum_i s_i \bullet (M \cdot 1N_i). \end{array}$$

(Recall that $1x = \eta(x) \in \mathcal{M}(X)$ is the singleton multiset.)

The substitution operation can now be defined by similar diagrams as in (7) – (9) above. For this example, we give it immediately in a more conventional notation:

$$\begin{array}{rcl} \mathrm{var}_{n+1}(\sum_{i \le n+1} s_i v_i)[\psi/v_{n+1}] & = & \begin{cases} \mathrm{var}_n(\sum_{i \le n} s_i v_i) & \text{if } s_{n+1} = 0 \\ \mathrm{var}_n(\sum_{i \le n} s_i v_i) + s_{n+1} \bullet \psi & \text{if } s_{n+1} \ne 0 \end{cases} \\ (\lambda v_{n+2}.\, M)[\psi/v_{n+1}] & = & \lambda v_{n+1}.\, M[v_{n+1}/v_{n+2}, v_{n+2}/v_{n+1}][\psi/v_{n+2}] \\ & = & \lambda v_{n+1}.\, M[v_{n+1}/v_{n+2}, \psi/v_{n+1}] \\ (M \cdot (\sum_i s_i N_i))[\psi/v_{n+1}] & = & M[\psi/v_{n+1}] \cdot (\sum_i s_k(N_i[\psi/v_{n+1}])). \end{array}$$

5.3 Non-deterministic automata

In this section, we present expressions with a fixed point operator denoting behaviours of non-deterministic automata. This formalises, using initial algebras, the work in [29]. We work again, first in the category $\mathcal{EM}(\mathcal{P}_{\mathrm{fin}})^{\mathbb{N}} = \mathbf{JSL}^{\mathbb{N}}$.

The presheaf of expressions $\mathsf{E} \in \mathbf{JSL}^{\mathbb{N}}$ is the initial algebra of the functor on $\mathbf{JSL}^{\mathbb{N}}$ given by:

$$P \longmapsto \mathcal{F} + \mathcal{W}(P) + 2 + A \cdot {!P},$$

where $2 = \{\bot, \top\}$ is the constant presheaf, $\mathcal{F}(n) = \mathcal{P}_{\mathrm{fin}}(\{v_1, \ldots, v_n\})$, and $!P = \mathcal{P}_{\mathrm{fin}}(P)$.

Coalgebraically, non-deterministic automata are modelled using the functor $F(X) = 2 \times \mathcal{P}_{\mathrm{fin}}(X)^A$ (in **Sets**). Both functors are tightly connected but further studying this connection is out of the scope of this paper.

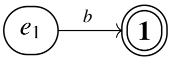
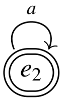
We can describe this initial algebra as the following map in $\mathbf{JSL}^{\mathbb{N}}$.

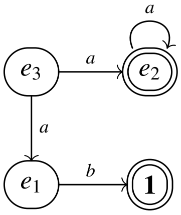
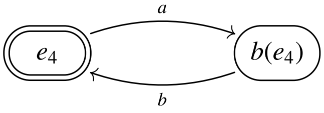
$$\mathcal{F} + \mathcal{W}(\mathsf{E}) + 2 + A \cdot {!\mathsf{E}} \xrightarrow[\cong]{[\mathrm{var},\mathrm{fix},\mathrm{ops},\mathrm{pre}]} \mathsf{E}$$

Elements of the set of expressions $\mathsf{E}(n) \in \mathbf{JSL}$ with variables from $\{v_1, \ldots, v_n\}$ are inductively given by:

- $\mathrm{var}_n(V)$ where $V \subseteq \{v_1, \ldots, v_n\}$;
- $\mathrm{fix}_n(e) = \mu v_{n+1}.e$, where $e \in \mathsf{E}(n+1)$;
- $\mathbf{0} = \mathrm{ops}(\bot)$;
- $\mathbf{1} = \mathrm{ops}(\top)$;
- $a(\{e_1, \ldots e_k\}) = \mathrm{pre}(\kappa_a(\{e_1, \ldots e_k\}))$, for $a \in A$ and $e_i \in \mathsf{E}(n)$;
- $\bot$ and $e \vee e'$ for any $e, e' \in \mathsf{E}(n)$.

The intuition behind these expressions is best explained operationally. Even though we have not given at this stage any operational semantics to the above expressions, and hence they are just syntax, we depict below several non-deterministic automata and corresponding expressions.

$e_1 = b(\mathbf{1})$	$e_2 = \mu x.a(\{x\}) \vee \mathbf{1}$
	

$e_3 = a(\{e_1, e_2\})$	$e_4 = \mu x.\mathbf{1} \vee a(b(x))$
 a e_3 a e_2 a e_1 b $\mathbf{1}$	

The binder μ has a very similar role to the Kleene star $(-)^\star$. For instance, it is the intended semantics that $e_2 = \mu x.a(\{x\}) \vee \mathbf{1} = a^\star$. Hence, semantically, we want μ to be a (least) fixed point. The fixed point equation $\mu x.e = e[\mu x.e/x]$ can be represented in this setting by the diagram:

$$\begin{tikzcd}
\mathcal{W}(!\mathsf{E}) \arrow[r, "\mathcal{W}(\Delta)"] \arrow[dd, "\mathcal{W}(\varepsilon)"'] & \mathcal{W}(!\mathsf{E} \otimes !\mathsf{E}) = \mathcal{W}(!\mathsf{E}) \otimes \mathcal{W}(!\mathsf{E}) \arrow[d, "\mathcal{W}(\varepsilon)\otimes id"] & \\
 & \mathcal{W}(\mathsf{E}) \otimes \mathcal{W}(!\mathsf{E}) = \mathcal{W}(\mathsf{E}) \otimes !\mathcal{W}(\mathsf{E}) \arrow[r, "id\otimes !\mathrm{fix}"] & \mathcal{W}(\mathsf{E}) \otimes !\mathsf{E} \arrow[d, "\mathrm{sbs}"] \\
\mathcal{W}(\mathsf{E}) \arrow[rr, "\mathrm{fix}"] & & \mathsf{E}
\end{tikzcd}$$

Here the map sbs is defined in the same way as for the non-deterministic lambda calculus. In order to obtain a (sound and) complete calculus of these expressions with respect to bisimilarity, one would have to impose more equations, see [29].

In formal language theory, the prime equivalence used is language (or trace) equivalence, which is coarser than bisimilarity. Rabinovich [27] showed that in the case of a simple fixed point calculus like the one we derive here, it is enough to add one axiom to the bisimilarity calculus, namely the axiom expressing distributivity of prefixing over $\vee$, given as $a(\{e \vee e'\}) = a(\{e\}) \vee a(\{e'\})$.

Here, we can express this diagrammatically as:

$$\begin{tikzcd}
 & & !!\mathsf{E} \arrow[dll, "\varepsilon"'] \arrow[drr, "!\varepsilon"] & & \\
!\mathsf{E} \arrow[r, "\kappa_a"'] & A \cdot \mathsf{E} \arrow[r, "\mathrm{pre}"'] & \mathsf{E} & A \cdot !\mathsf{E} \arrow[l, "\mathrm{pre}"] & !\mathsf{E} \arrow[l, "\kappa_a"]
\end{tikzcd} \tag{10}$$

This diagram is actually a more general formulation of the Rabinovich distributivity rule, since the starting point $!!\mathsf{E} = \mathcal{P}^2_{\text{fin}}(\mathsf{E})$ at the top involves sets of sets of expressions. To recover the binary version above one should start from $\{\{e_1, e_2\}\} \in\, !!\mathsf{E}$. The commutativity of the above diagram says that $a(\{e_1 \vee e_2\}) = a(\{e_1, e_2\})$, and the right side of this equation rewrites to $a(\{e_1, e_2\}) = a(\{e_1\}) \vee a(\{e_2\})$, since pre is a **JSL** map. The relevance of capturing this law diagrammatically is that it provides a guideline to which law gives trace semantics in other settings. For instance, we show in the next subsection that for weighted automata, the same diagram can be used to axiomatise trace semantics.

5.4 Weighted automata

We can obtain a syntax for weighted automata in a similar way by switching to the multiset monad $\mathcal{M}$, for a semiring S, and considering the initial algebra E of the functor on $\mathbf{SMod}^{\mathbb{N}}$:

$$P \;\longmapsto\; \mathcal{F} + \mathcal{W}(P) + S + A \cdot {!P},$$

where S denotes the constant presheaf $\Delta(S)$, and $\mathcal{F}(n) = \mathcal{M}(\{v_1, \dots, v_n\})$, and $!P = \mathcal{M}(P)$.

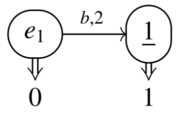
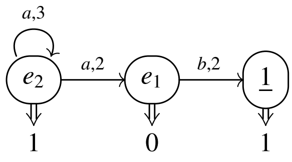
Again, coalgebraically, weighted automata are modelled using the functor $F(X) = S \times \mathcal{M}(X)^A$ (in **Sets**). We can describe the above initial algebra as the following isomorphism.

$$\mathcal{F} + \mathcal{W}(\mathsf{E}) + S + A \cdot {!\mathsf{E}} \xrightarrow[\cong]{[\text{var},\text{fix},\text{val},\text{pre}]} \mathsf{E}$$

Elements of the set of expressions $\mathsf{E}(n) \in \mathbf{SMod}$ with variables from $\{v_1, \dots, v_n\}$ are inductively given by:

- $\text{var}_n(V)$ where $V \subseteq \{v_1, \dots, v_n\}$;
- $\text{fix}_n(e) = \mu v_{n+1}.e$, where $e \in \mathsf{E}(n+1)$;
- $\underline{s} = \text{val}(s)$, for any $s \in S$;
- $a(\sum_i s_i e_i) = \text{pre}(\kappa_a(\sum_i s_i e_i))$;
- 0, $s \bullet e$, and $e + e'$ for any $e, e' \in \mathsf{E}(n)$ and $s \in S$.

Intuitively, the expression $\underline{s}$ denotes that the output of a state is s and $a(\sum_{k \leq m} s_k e_k)$ denotes a state with m transitions labelled by a, s_k to the states denoted by the expressions $e_1, \dots, e_m$. We depict below two examples of expressions (over the semiring of integers) and the respective (intended) weighted automata they represent.

$e_1 = b(2 \bullet \underline{1})$	$e_2 = \mu x.a(3x + 2e_1) + \underline{1}$
	 a,3 e_2 a,2 e_1 b,2 $\underline{1}$ 1 0 1

Note that e_1 above has output 0 because the expression only specifies the b transition and no concrete output. Intuitively, the expression e_1 specifies a state which has a b

transition with weight 2 to a state specified by the expression $\underline{1}$. The latter specifies a state with no transitions and with output 1. The use of + in e_2 allows the combined specification of transitions and outputs and hence e_2 has output 1.

Examples of equations one might want to impose in this calculus include $a(r0) = 0$, which can be expressed using the diagram:

$$\begin{tikzcd}
I = {!}1 = S \arrow[d, "(-)\bullet\eta(0)"'] \arrow[rrd, bend left, "0"] & & \\
{!}\mathsf{E} \arrow[r, "\kappa_a"'] & A\cdot{!}\mathsf{E} \arrow[r, "\mathrm{pre}"'] & \mathsf{E}
\end{tikzcd}$$

Recent papers [3,7] on generalising the work of Rabinovich, have proposed, in a rather *ad hoc* fashion, axiomatisations of trace semantics for weighted automata. The very same diagram (10) capturing Rabinovich's axiom for non-deterministic automata can now be interpreted in this weighted setting and it will result in the following axiom — which we present immediately a binary version, for readability.

$$a(s(e_1 + e_2)) \;=\; a(se_1) + a(se_2).$$

(We write *se* for the singleton multiset $s \bullet 1e$.)

Final remarks

We have exploited initiality in categories of algebra-valued presheaves $\mathcal{EM}(T)^{\mathbb{N}}$, for a monad T on **Sets**, to modularly obtain a syntactic calculus of expressions as initial algebra of a given functor. The theory involved provides a systematic description of several non-trivial examples: non-deterministic and weighted lambda calculus and expressions for (weighted) automata.

The use of presheaves to describe calculi with binding operators has first been studied by Fiore, Plotkin and Turi. They used set-valued presheaves and did not explore the formalisation of relevant equations for concrete calculi. They have also not explored their theory to derive fixed point calculi. Non-deterministic versions of the lambda calculus have appeared in, for instance, [6,26]. The version we derive in this paper is however slightly different because we have linearity also in the second argument of application.

As future work, we wish to explore the formalisation of equations that we presented for the concrete examples. This opens the door to developing a systematic account of sound and complete axiomatisations for different equivalences (bisimilarity, trace, failure, *etc.*). Preliminary work for (generalised) regular expressions and bisimilarity has appeared in [15,31,30] and for trace semantics in [3].

Acknowledgments

Thanks to Sam Staton for suggesting several improvements.

References

1. R. Atkey, P. Johann, and N. Ghani. When is a type refinement an inductive type? In M. Hofmann, editor, *Foundations of Software Science and Computation Structures*, number 6604 in Lect. Notes Comp. Sci., pages 72–87. Springer, Berlin, 2011.
2. M. Barr and Ch. Wells. *Toposes, Triples and Theories*. Springer, Berlin, 1985. Revized and corrected version available from URL: `www.cwru.edu/artsci/math/wells/pub/ttt.html`.
3. M. Bonsangue, S. Milius, and A. Silva. Sound and complete axiomatizations of coalgebraic language equivalence. *ACM Trans. on Computational Logic*, 14(1), 2013.
4. F. Borceux. *Handbook of Categorical Algebra*, volume 50, 51 and 52 of *Encyclopedia of Mathematics*. Cambridge Univ. Press, 1994.
5. B. Coecke, D. Pavlović, and J. Vicary. A new description of orthogonal bases. *Math. Struct. in Comp. Sci.*, pages 1–13, 2012.
6. U. de'Liguoro and A. Piperno. Non-deterministic extensions of untyped λ-calculus. *Inf. & Comp.*, 122:149–177, 1995.
7. Z. Ésik and W. Kuich. Free iterative and iteration k-semialgebras. *CoRR*, abs/1008.1507, 2010.
8. M. Fiore, G. Plotkin, and D. Turi. Abstract syntax and variable binding. In *LICS*, pages 193–202. IEEE Computer Society, 1999.
9. M. Fiore and S. Staton. Comparing operational models of name-passing process calculi. *Inf. & Comp.*, 2004(4):524–560, 2006.
10. M. Fiore and D. Turi. Semantics of name and value passing. In *Logic in Computer Science*, pages 93–104. IEEE, Computer Science Press, 2001.
11. J. Goguen, J. Thatcher, E. Wagner, and J. Wright. Initial algebra semantics and continuous algebras. *Journ. ACM*, 24(1):68–95, 1977.
12. I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace theory via coinduction. *Logical Methods in Computer Science*, 3(4:11), 2007.
13. B. Jacobs. Semantics of weakening and contraction. *Ann. Pure & Appl. Logic*, 69(1):73–106, 1994.
14. B. Jacobs. *Categorical Logic and Type Theory*. North Holland, Amsterdam, 1999.
15. B. Jacobs. A bialgebraic review of deterministic automata, regular expressions and languages. In K. Futatsugi, J.-P. Jouannaud, and J. Meseguer, editors, *Essays Dedicated to Joseph A. Goguen*, volume 4060 of *Lecture Notes in Computer Science*, pages 375–404. Springer, 2006.
16. B. Jacobs. Bases as coalgebras. In A. Corradini, B. Klin, and C. Cîrstea, editors, *Conference on Algebra and Coalgebra in Computer Science (CALCO 2011)*, number 6859 in Lect. Notes Comp. Sci., pages 237–252. Springer, Berlin, 2011. Extended version appears in *Logical Methods in Computer Science*.
17. B. Jacobs. *Introduction to Coalgebra. Towards Mathematics of States and Observations*. 2012. Book, in preparation; version 2 available from `www.cs.ru.nl/B.Jacobs/CLG/JacobsCoalgebraIntro.pdf`.
18. A. Joyal and I. Moerdijk. *Algebraic Set Theory*. Number 220 in LMS. Cambridge Univ. Press, 1995.
19. B. Klin. Bialgebras for structural operational semantics: An introduction. *Theor. Comp. Sci.*, 412(38):5043–5069, 2011.
20. A. Kock. Bilinearity and cartesian closed monads. *Math. Scand.*, 29:161–174, 1971.
21. A. Kock. Closed categories generated by commutative monads. *Journ. Austr. Math. Soc.*, XII:405–424, 1971.

22. J. Lambek. A fixed point theorem for complete categories. *Math. Zeitschr.*, 103:151–161, 1968.
23. E.G. Manes. *Algebraic Theories*. Springer, Berlin, 1974.
24. S. Mac Lane. *Categories for the Working Mathematician*. Springer, Berlin, 1971.
25. R. Møgelberg and S. Staton. Linearly-used state in models of call-by-value. In A. Corradini, B. Klin, and C. Cïrstea, editors, *Conference on Algebra and Coalgebra in Computer Science (CALCO 2011)*, number 6859 in Lect. Notes Comp. Sci., pages 293–313. Springer, Berlin, 2011.
26. M. Pagani and S. Ronchi della Rocca. Solvability in resource lambda-calculus. In *Proceedings of the 13th international conference on Foundations of Software Science and Computational Structures*, FOSSACS'10, pages 358–373, Berlin, Heidelberg, 2010. Springer-Verlag.
27. Alexander Moshe Rabinovich. A complete axiomatisation for trace congruence of finite state behaviors. In Stephen D. Brookes, Michael G. Main, Austin Melton, Michael W. Mislove, and David A. Schmidt, editors, *MFPS*, volume 802 of *Lecture Notes in Computer Science*, pages 530–543. Springer, 1993.
28. J. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000.
29. A. Silva. *Kleene coalgebra*. PhD thesis, Radboud University Nijmegen, 2010.
30. A. Silva, F. Bonchi, M. Bonsangue, and J. Rutten. Generalizing the powerset construction, coalgebraically. In *Proc. FSTTCS 2010*, volume 8 of *LIPIcs*, pages 272–283, 2010.
31. A. Silva, M. Bonsangue, and J. Rutten. Non-deterministic kleene coalgebras. *Logical Methods in Computer Science*, 6(3), 2010.
32. T. Uustalu, V. Vene, and A. Pardo. Recursion schemes from comonads. *Nordic Journ. Comput.*, 8(3):366–390, 2001.