Monad Transformers as Monoid Transformers

Mauro Jaskelioff
Functional Programming Laboratory, University of Nottingham, UK

Eugenio Moggi
DISI, Università di Genova, Italy

Abstract

The incremental approach to modular monadic semantics constructs complex monads by using monad transformers to add computational features to a pre-existing monad. A complication of this approach is that the operations associated to the pre-existing monad need to be lifted to the new monad.

In a companion paper by Jaskelioff, the lifting problem has been addressed in the setting of system $F_\omega$. Here, we recast and extend those results in a category-theoretic setting. We abstract and generalize from monads to monoids (in a monoidal category), and from monad transformers to monoid transformers. The generalization brings more simplicity and clarity, and lays the foundation for an abstract theory of operation lifting with applicability beyond monads.

Key words: Monad, Monoid, Monoidal Category, Codensity Monad

1. Introduction

Since monads have been proposed to model computational effects [Mog89, Mog91], they have proven to be extremely useful also to structure functional programs [Wad92b, Wad92a, JW93]. In these applications monads come with operations to manipulate the computational effects they model. For example, an exception monad may come with operations for throwing an exception and for handling it, and a state monad may come with operations for reading and updating the state. Consequently, the structures one is really working with are monads and a set of operations associated to them. The monadic approach to the denotational semantics of a programming language, which has been adapted also to other forms of programming language semantics based on interpreters [LHJ95] or compilers [LH96], consists of three steps [Mog97, BHM00]:

- identify a metalanguage with computational types, to hide the interpretation of computational types and operations manipulating computations;
• define a translation of the programming language into the metalanguage;

• give a denotational semantics of the metalanguage, by interpreting computational types and operations on computations using a monad and a set of operations associated to it.

However, there is a caveat: when the programming language involves a mixture of computational effects, the number of operations for manipulating computations grows, the monad needed to interpret computational types gets more complex, and the semantics of operations associated to it gets more complex, too. To tackle these issues one can adopt a modular approach, by providing a set of basic building blocks and operations to build more complex blocks, so that complex monads can be built from simpler ones. Roughly speaking, one can identify two modular approaches

• the incremental approach, taken in [LHJ95, Mog97, BHM00], uses unary operations, called monad transformers, which build complex monads by adding one computational feature to a pre-existing monad;

• the compositional approach, taken in [LG02, HPP06], uses binary operations, which build complex monads by combining two pre-existing monads.

Both approaches fall short in dealing with operations associated to monads. This problem was identified in [LHJ95], which proposed a non-modular workaround, namely to lift in an ad-hoc manner the operations associated to a monad through a monad transformer. Therefore, the number of liftings of operations grows like the product of the number of monad transformers and operations involved. Alternatively, one may achieve modularity by restricting the format of operations, for instance algebraic operations in the sense of [PP01] are straightforward to lift. However, the applicability of the monadic approach becomes limited if all the operations on computations have to be algebraic.

The compositional approach fits well with the algebraic view of computational effects advocated in [PP01, HPP06], where monads are replaced by algebraic theories, and combining computational effects is reduced to composition of algebraic theories. Unfortunately, some computational monads are not induced by algebraic theories, and some operations on computations are not algebraic.

The incremental approach is the most popular among functional programmers, because monad transformers are easy to implement. However, there has been limited progress in addressing the lifting problem since [LHJ95], until a new insight was brought by Jaskelioff [Jas08, Jas09].

Jaskelioff gives a uniform lifting for a class of operations, which includes (after some minor repackaging) all the operations described in [LHJ95], through any monad transformer with a functorial behaviour. On algebraic operations Jaskelioff’s lifting agrees with the straightforward lifting, and it is compatible with most of the ad-hoc liftings found in the literature or in Haskell’s libraries. This uniform lifting has been implemented in Haskell [Jas08] and studied in the setting of system $F_\omega$ [Jas09].
The aim of this paper is to study Jaskelioff’s uniform lifting, and more generally the lifting problem, in a category-theoretic setting. Our main contribution is to develop a theory of monoid transformers and lifting of operations in an abstract categorical setting, that generalises, clarifies, and extends the current theory of monad transformers [LHJ95, Mog97, BHM00, Jas09].

Category Theory is well-known for its ability to abstract and generalize. We make good use of this ability, by developing a general theory of lifting for monoid transformers, where monoids are taken in some unspecified monoidal category. By a suitable choice of monoidal category, the general theory can be specialized to monads, strong monads, and monads expressible in system $F\omega$ (or some other typed calculus). Also other structures proposed for modeling computational effects, namely arrows [Hug00] and Freyd’s categories [PR97], can be viewed as monoids in suitable monoidal categories [HJ06, Atk08]. Therefore, the general theory of lifting may have a wider applicability than originally envisaged.

Note for Readers. We assume a modest knowledge of Category Theory, and the notions relevant to the paper, but outside the scope of an introductory text book, are briefly recalled in Section 2. Further information can be found in more advanced texts such as [ML71, BW85, Bor94, BW95]. Each Section includes an extensive collection of examples. Some examples are not self-contained and assume knowledge of certain topics. However, these examples are not essential to understand the rest of the paper.

Summary. Section 2 introduces monoidal categories (together with functors and natural transformations) and notions, such as exponentials and monoids, definable in the setting of any monoidal category. Section 3 introduces a taxonomy of operations associated to a monoid, and gives the most general formulation of the lifting problem, namely what it means to lift an operation along a monoid morphism (Theorem 3.4 shows that lifting of algebraic operations is always possible). Section 4 introduces a taxonomy of monoid transformers, gives examples of strong monad transformers clarifying where they fit in the taxonomy, and provides two additional solutions to the lifting problem (see Section 4.2). Section 5 concludes with some considerations on related and future work.

2. Monoidal Categories

It is well-known [ML71] that monads on a category $\mathcal{C}$ correspond to monoids in the (strict) monoidal category $\text{Endo}(\mathcal{C})$ of endofunctors on $\mathcal{C}$. A similar correspondence holds when monads are replaced by strong monads on a cartesian closed category $\mathcal{C}$ or by monads expressible in system $F\omega$ (or some other typed calculus of adequate expressivity), provided $\text{Endo}(\mathcal{C})$ is replaced with a suitable (strict) monoidal category $\hat{\mathcal{E}}$. These observations suggest that a theory of monad transformers can be viewed as an instance of a theory of monoid transformers in the setting of a monoidal category $\hat{\mathcal{E}}$. There are two main advantages in moving to this more abstract setting:

- simplicity: monoids (in a monoidal category $\hat{\mathcal{E}}$) are simpler than monads;
• generality: the theory has several instantiations by choosing different $\hat{E}$.

**Definition 2.1 (Monoidal Category [ML71]).** A monoidal category $\hat{E}$ is a tuple $(\hat{E}, \otimes, I, \alpha, \lambda, \rho)$, where

- $\hat{E}$ is a category, $\otimes: \hat{E} \times \hat{E} \rightarrow \hat{E}$ is a bifunctor, $I \in \hat{E}$ is an object
- $\alpha_{a,b,c}: a \otimes (b \otimes c) \rightarrow (a \otimes b) \otimes c$, $\lambda_a: I \otimes a \rightarrow a$, $\rho_a: a \otimes I \rightarrow a$ are natural isomorphisms such that (the following diagrams commute)

$$
\begin{align*}
& a \otimes (b \otimes (c \otimes d)) & \xrightarrow{\alpha} & (a \otimes b) \otimes (c \otimes d) & \xrightarrow{\alpha} & ((a \otimes b) \otimes c) \otimes d \\
& id \otimes \alpha & & & \alpha \otimes id & \\
& \downarrow & & \downarrow & & \\
& a \otimes ((b \otimes c) \otimes d) & \xrightarrow{\alpha} & (a \otimes (b \otimes c)) \otimes d \\
& & \downarrow & & \\
& & a \otimes (I \otimes b) & \xrightarrow{\alpha} & (a \otimes I) \otimes b & \\
& & \downarrow & & \\
& & \id \otimes \lambda & = & \rho \otimes \id & \\
& & \downarrow & & \\
& & a \otimes b & \xrightarrow{=} & a \otimes b & \\
\end{align*}
$$

When the natural isomorphisms $\alpha$, $\lambda$, and $\rho$ are identities, the diagrams necessarily commute, and the monoidal category is called strict.

**Definition 2.2 (Monoid).** The category $\text{Mon}(\hat{E})$ of monoids in a monoidal category $\hat{E}$ is given by

- objects are monoids $\hat{M} = (M, e, m)$, i.e. diagrams

$$
\begin{align*}
& I \xrightarrow{e} M \xleftarrow{m} M \otimes M \\
& (M \otimes M) \otimes M \xrightarrow{m \otimes id} M \otimes M \\
& M \otimes (M \otimes M) \xrightarrow{id \otimes m} M \otimes M \xrightarrow{m} M \\
& I \otimes M \xrightarrow{e \otimes id} M \otimes M \xleftarrow{id \otimes e} M \otimes I \\
& \downarrow \quad \downarrow \quad \downarrow \\
& \quad \quad \quad M \quad M \\
& \quad \quad \quad \quad \quad m \\
\end{align*}
$$
arrows from $\hat{M}_1$ to $\hat{M}_2$ are arrows $M_1 \xrightarrow{f} M_2$ in $\mathcal{E}$ such that

$$
\begin{array}{c}
I \xrightarrow{e_1} M_1 \xleftarrow{m_1} M_1 \otimes M_1 \\
\downarrow f \quad \quad \downarrow f \otimes f \\
I \xrightarrow{e_2} M_2 \xleftarrow{m_2} M_2 \otimes M_2
\end{array}
$$

Identities and composition in $\text{Mon}(\hat{\mathcal{E}})$ are inherited from $\mathcal{E}$.

The forgetful functor $U: \text{Mon}(\hat{\mathcal{E}}) \rightarrow \mathcal{E}$ maps a monoid $\hat{M}$ to $M$ and an arrow $\hat{M}_1 \xrightarrow{f} \hat{M}_2$ to $M_1 \xrightarrow{f} M_2$.

**Definition 2.3 (Exponential).** An exponential of $b$ to $a$ in a monoidal category $\hat{\mathcal{E}}$ is a map $\text{ev}: b^a \otimes a \rightarrow b$ satisfying the universal property

$$
\forall x \in \mathcal{E}, \forall f: x \otimes a \rightarrow b. \exists! \Lambda f: x \rightarrow b^a \text{ such that } b^a \otimes a \xrightarrow{\text{ev}} b \xrightarrow{\Lambda f} x \otimes a
$$

**Definition 2.4 (Monoidal Functor).** Given two monoidal categories $\hat{\mathcal{E}}$ and $\hat{\mathcal{E}}'$, a monoidal functor $\hat{T}$ from $\hat{\mathcal{E}}$ to $\hat{\mathcal{E}}'$ is a tuple $(T, \phi_I, \phi_a)$, where

- $T: \mathcal{E} \rightarrow \mathcal{E}'$ is a functor
- $\phi_I: I' \rightarrow TI$ is a map, $\phi_{a,b}: Ta \otimes' T'b \rightarrow T(a \otimes b)$ is a natural transformation such that

$$
\begin{array}{ccc}
Ta \otimes' (Tb \otimes' Tc) & \xrightarrow{id \otimes' \phi} & Ta \otimes' T(b \otimes c) & \xrightarrow{\phi} & T(a \otimes (b \otimes c)) \\
\downarrow & & \downarrow & & \downarrow T(\alpha) \\
(Ta \otimes' Tb) \otimes' Tc & \xrightarrow{\phi \otimes' \text{id}} & T(a \otimes b) \otimes' Tc & \xrightarrow{\phi} & T((a \otimes b) \otimes c)
\end{array}
$$

$$
\begin{array}{ccc}
I' \otimes Ta & \xrightarrow{\lambda'} & Ta \\
\downarrow & & \downarrow T(\lambda) \\
TI \otimes' Ta & \xrightarrow{\phi} & T(I \otimes a)
\end{array}
$$

$$
\begin{array}{ccc}
Ta \otimes' I' & \xrightarrow{\rho'} & Ta \\
\downarrow & & \downarrow T(\rho) \\
Ta \otimes' TI & \xrightarrow{\phi} & T(a \otimes I)
\end{array}
$$
When the map $\phi_I$ and the natural transformation $\phi$ are identities, the monoidal functor is called strict, and the commuting diagrams amount to require that $I' = TI$, $Ta \otimes' Tb = T(a \otimes b)$, $\alpha' = T(\alpha)$, $\lambda' = T(\lambda)$ and $\rho' = T(\rho)$.

**Definition 2.5 (Monoidal Natural Transformation).** Given the monoidal functors $\hat{T}$ and $\hat{T}'$ from $\hat{E}$ to $\hat{E}'$, a monoidal natural transformation $\tau$ from $\hat{T}$ to $\hat{T}'$ is a natural transformation $\tau : T \Rightarrow T'$ such that

$\tau : T \Rightarrow T'$ such that

$$
\begin{align*}
I' & \xRightarrow{\phi_I} I' \\
T(a \otimes' Tb) & \xRightarrow{\tau \otimes' \tau} T'(a \otimes' Tb) \\
\phi' & \xRightarrow{\phi} \phi' \\
T(a) & \xRightarrow{\tau} T'(a)
\end{align*}
$$

**Theorem 2.6 (Extension).** A monoidal functor $\hat{T} : \hat{E} \rightarrow \hat{E}'$ induces a functor $T : \operatorname{Mon}(\hat{E}) \rightarrow \operatorname{Mon}(\hat{E}')$. Similarly a monoidal natural transformation $\tau : \hat{T} \Rightarrow \hat{T}'$ induces a natural transformation $\tau : T \Rightarrow T'$ such that

$$
\begin{array}{c}
\begin{array}{c}
\operatorname{Mon}(\hat{E}) \\
\downarrow \tau
\end{array} \\
\begin{array}{c}
\operatorname{Mon}(\hat{E}') \\
\downarrow \tau
\end{array}
\end{array}
\xRightarrow{T}
\begin{array}{c}
\begin{array}{c}
\hat{E} \\
\downarrow \tau
\end{array} \\
\begin{array}{c}
\hat{E}' \\
\downarrow \tau
\end{array}
\end{array}
$$

namely

$$
T \hat{M} \xrightarrow{\phi_I} TI \xrightarrow{T(e)} TM \xleftarrow{T(M \otimes M)} T'(M \otimes M) \xrightarrow{T'} TM \otimes' TM
$$

$$
\tau_M : T \hat{M} \xrightarrow{\tau_M} T' \hat{M}
$$

**2.1. Examples of Monoids**

We give constructions of objects in $\operatorname{Mon}(\hat{E})$, which may require additional assumptions on the monoidal category $\hat{E}$. More examples of monoids, in the form of strong monads, will be given in Section 3.1.

**Example 2.7** The initial monoid $\hat{I}$, is given by the diagram

$$
\begin{array}{c}
I \xrightarrow{id} I \\
\leftarrow \hat{I} \xleftarrow{\lambda} I \otimes I
\end{array}
$$

In fact, $\hat{I}$ is an initial object in $\operatorname{Mon}(\hat{E})$. 
Example 2.8 When $\mathcal{E}$ has $J$-limits, then $\text{Mon}(\hat{\mathcal{E}})$ has $J$-limits which are computed pointwise, therefore they are preserved by the forgetful functor $U$. In particular, if $\mathcal{E}$ has a terminal object $1$, then on $1$ there is a unique monoid structure $\hat{1}$, which yields a terminal object in $\text{Mon}(\hat{\mathcal{E}})$.

Example 2.9 When the exponential $a^a$ exists, we have a **monoid of endomorphisms** on $a$, given by the diagram

$$I \xrightarrow{i_a} a^a \xleftarrow{c_a} a^a \otimes a^a$$

where

$$i_a \triangleq \Lambda(I \otimes a \xrightarrow{\lambda} a \xrightarrow{\text{id}} a)$$

$$c_a \triangleq \Lambda((a^a \otimes a^a) \otimes a \xrightarrow{\alpha^{-1}} a^a \otimes (a^a \otimes a) \xrightarrow{\text{id} \otimes ev} a^a \otimes a \xrightarrow{\text{ev}} a)$$

Moreover, if $\hat{M} = (M, e, m)$ is a monoid, then $M \xrightarrow{\Lambda_m} M^M$ is a monoid morphism from $\hat{M}$ to the monoid of endomorphisms on $M$.

Example 2.10 When the left-adjoint $(-)^*$ to $U : \text{Mon}(\hat{\mathcal{E}}) \rightarrow \mathcal{E}$ exists, it gives **free monoids**. There are several assumptions on $\hat{\mathcal{E}}$, which imply the existence of free monoids, for instance:

- When $\mathcal{E}$ has denumerable coproducts, and for each $a \in \mathcal{E}$ the functors $a \otimes -$ and $- \otimes a$ preserve these coproducts, then $a^*$ exists (see [ML71]) and its carrier is given by the coproduct of the family $(a^n \mid n \in \mathbb{N})$ with $a^0 \triangleq I$ and $a^{n+1} \triangleq a \otimes a^n$.

- When $\mathcal{E}$ has binary coproducts and $\omega$-colimits, and for each $a \in \mathcal{E}$ the functors $a \otimes -$ and $- \otimes a$ preserves $\omega$-colimits and $- \otimes a$ preserves also binary coproducts, then $a^*$ exists and its carrier is given by the colimit of the $\omega$-chain $(a_n \xrightarrow{f_n} a_{n+1} \mid n \in \mathbb{N})$ with $f_0 \triangleq I \xrightarrow{\text{inl}} I + (a \otimes I)$ and $f_{n+1} \triangleq I + (a \otimes a_n) \xrightarrow{\text{id} + (\text{id} \otimes f_n)} I + (a \otimes a_{n+1})$.

Example 2.11 Given a monoid $\hat{M} = (M, e, m)$ in $\hat{\mathcal{E}}$, one can define a **sub-monoid** of $\hat{M}$ as follows. Take a monic $M' \xrightarrow{i} M$ in $\mathcal{E}$, if there exist maps $e'$ and $m'$ (necessarily unique, since $i$ is monic) such that

$$\begin{array}{ccc}
I & \xrightarrow{e} & M \\
& \searrow & \downarrow i \\
& & M' \xleftarrow{m'} M' \otimes M'
\end{array}$$

then $\hat{M}' = (M', e', m')$ is a monoid, called the sub-monoid of $\hat{M}$ induced by $M'$, and $\hat{M}' \xrightarrow{i} \hat{M}$ is a monoid monomorphism.
The definition of quotient of a monoid $\hat{M}$ is more involved: one should consider an equivalence relation on $M$, and unless $\otimes$ preserves coequalizers of equivalence relations, the quotient monoid $M/R$ is ill-defined. We give instead concrete descriptions of sub-monads and quotient monads in $\textbf{Set}$, i.e. sub-monoids and quotient monoids in $\textbf{Endo}(\textbf{Set})$ of Example 2.13. Given a monad $\hat{M} = (M, \eta, -^\ast)$ on $\textbf{Set}$ presented as a Kleisli triple (see [Man76, Mog91]):

- A sub-monad of $\hat{M}$ is uniquely identified by a family of subsets $(S_X \subseteq MX \mid X)$ such that $\forall X. \forall x \in X. \eta_X(x) \in S_X$ and $\forall X, Y. \forall f : X \longrightarrow S_Y. \forall x \in S_X. g^\ast x \in S_Y$ where $g = X \xrightarrow{f} S_Y \xhookrightarrow{} MY$.

- A quotient monad of $\hat{M}$ is uniquely identified by a family of equivalence relations $(R_X \subseteq MX \times MX \mid X)$ such that $\forall X, Y. \forall f : X \longrightarrow R_Y. \forall (x_1, x_2) \in R_X. (g_1^\ast x_1, g_2^\ast x_2) \in R_Y$ where $g_i = X \xrightarrow{f} R_Y \xrightarrow{\pi_i} MY$.

The class of sub-monads of $\hat{M}$ (and similarly for quotient monads) has an obvious partial order (given by pointwise inclusion) which is closed w.r.t. arbitrary meets (computed by pointwise intersection), namely $(\bigwedge_{S \in \mathcal{S}} S)_X = \bigcap_{S \in \mathcal{S}} S_X$.

Therefore, any family $S = (S_X \subseteq MX \mid X)$ of subsets generates the smallest sub-monad containing $S$, and any family $R = (R_X \subseteq MX \times MX \mid X)$ of relations generates the smallest quotient monad containing $R$.

2.2. Examples of Monoidal Categories

We give examples of monoidal categories, and when possible we say whether they have free monoids or exponentials.

**Example 2.12** A category $\mathcal{C}$ with finite products (e.g. the category $\textbf{Set}$ of sets) forms a symmetric monoidal category $(\mathcal{C}, \times, 1, \alpha, \lambda, \rho)$, where $\times$ is a binary product functor, $1$ is a terminal, and the natural isomorphisms are uniquely determined by the universal properties of products. In this monoidal category exponentials (in the sense of Definition 2.3) correspond to the usual notion of exponentials for a cartesian closed category.

**Example 2.13** If $\mathcal{C}$ is a (small) category, then the category $\textbf{Endo}(\mathcal{C})$ of endofunctors over $\mathcal{C}$ forms a strict monoidal category $(\textbf{Endo}(\mathcal{C}), \circ, \text{Id})$, where $\circ$ is functor composition and $\text{Id}$ is the identity functor, more precisely

- **objects** are endofunctors $F : \mathcal{C} \longrightarrow \mathcal{C}$
- **arrows** from $F$ to $G$ are natural transformations $\tau : F \longrightarrow G$
- **tensor** is functor composition $(G \circ F)(-) \triangleq G(F(-))$
- **unit** is the identity functor $\text{Id}(-) \triangleq -$. 
Also the category of profunctors $\mathcal{C}^{op} \times \mathcal{C} \to \text{Set}$ forms a monoidal category (see [Bor94]), and there is a monoidal functor from endofunctors to profunctors mapping $F$ to $\mathcal{C}(-, F-)$.

If $\mathcal{C}$ has $J$-limits, i.e. limits for diagrams of shape $J$, then so does $\text{Endo}(\mathcal{C})$, these $J$-limits in $\text{Endo}(\mathcal{C})$ are computed pointwise and are preserved by the functors $- \circ F : \text{Endo}(\mathcal{C}) \to \text{Endo}(\mathcal{C})$ (similar results hold for $J$-colimits). Moreover, in this monoidal category an exponential $G^F$ corresponds to a right Kan’s extension of $G$ along $F$, characterized by a bijection from $H \to G^F$ to $H \circ F \to G$ natural in the endofunctor $H$.

**Example 2.14** We define the category $\text{Endo}(\text{Set})_f$ of finitary endofunctors on $\text{Set}$. This category inherits the monoidal structure of $\text{Endo}(\text{Set})$, but unlike $\text{Endo}(\text{Set})$ it has also free monoids (see Example 2.10) and exponentials. These results generalize when $\text{Set}$ is replaced by a locally finitely presentable enriched category (see [KP93]). A finitary endofunctor $F$ on $\text{Set}$ is determined by its action on finite sets (e.g. see [BW95]), we give two equivalent characterizations

- $F$ preserves filtered colimits;
- for any $x \in FX$, exist $n$ finite, $i : n \to X$ and $x' \in Fn$ s.t. $(Fi)x' = x$.

We write $\text{Endo}(\text{Set})_f$ for the full subcategory of $\text{Endo}(\text{Set})$ whose objects are finitary endofunctors.

The first characterization implies that the identity functor is finitary, composition of finitary endofunctors is finitary, and the colimit in $\text{Endo}(\text{Set})$ of a diagram in $\text{Endo}(\text{Set})_f$ is in $\text{Endo}(\text{Set})_f$. Therefore, $\text{Endo}(\text{Set})_f$ inherits from $\text{Endo}(\text{Set})$ the monoidal structure and colimits, thus the inclusion of $\text{Endo}(\text{Set})_f$ into $\text{Endo}(\text{Set})$ is a strict monoidal functor, which creates and preserves colimits. As a consequence, the monoidal category $\text{Endo}(\text{Set})_f$ has free monoids, since it satisfies the second condition in Example 2.10 ($\omega$-colimits are preserved by $F \otimes -$ because they are a special case of filtered colimits).

The second characterization implies that $\text{Endo}(\text{Set})_f$ is equivalent to the category of functors $\text{Set}^{\text{Set}_f}$, where $\text{Set}_f$ is the full small subcategory of $\text{Set}$ whose objects are finite cardinals (aka natural numbers). In one direction the equivalence is given by restricting an endofunctor $F$ to $\text{Set}_f$ (we denote this restriction with $F_f$), in the other direction it is given by the left Kan extension along the inclusion $J : \text{Set}_f \to \text{Set}$

$$\text{Lan}_J F_f = \int^n_{-n \times (F_f n)}$$

i.e. the coend (see [ML71, Ch 9 and 10]) of $S : \text{Set}^{op}_{\text{Set}_f} \times \text{Set}_f \to \text{Endo}(\text{Set})$ where $S(m,n) \cong -m \times (F_f n)$. In fact, $S$ factors through $\text{Endo}(\text{Set})_f$, as $-m \times A$ is finitary when $m \in \text{Set}_f$ and $A \in \text{Set}$, thus the coend (which is a colimit) is in $\text{Endo}(\text{Set})_f$, too. The monoidal structure on $\text{Endo}(\text{Set})_f$ induces on $\text{Set}^{\text{Set}_f}$. 


the following tensor (with unit given by the inclusion functor $J$)

$$(H \otimes F)a \doteq \int^n (Fa)^n \times (Hn)$$

i.e. the coend with parameter for $S : \text{Set}_f \times \text{Set}_f^{op} \times \text{Set}_f \to \text{Set}$ where $S(a, m, n) \doteq (Fa)^m \times (Hn)$. The exponential $G^F$ in $\text{Set}^{\text{Set}_f}$ is given by

$$(G^F)a \doteq \int^n (Gn)((Fn)^a)$$

i.e. the end with parameter for $T : \text{Set}_f \times \text{Set}_f^{op} \times \text{Set}_f \to \text{Set}$ where $T(a, m, n) \doteq (Gn)^{(Fm)^a}$. To prove that $G^F$ is an exponential requires general properties of ends and coends, which can be found in [ML71, Ch 9].

**Example 2.15** Let $(A, \cdot)$ be a partial combinatory algebra, i.e. a set $A$ with two distinct elements $K \neq S$ and a partial binary operation $\cdot : A \times A \to A$, we write $a b$ for $\cdot(a, b)$, such that

- $K x y = x$ i.e. $K$ fixes elements
- $S x y \uparrow$ i.e. $S$ and $(S x) y$ are defined
- $S x y z \simeq x z (y z)$ i.e. both terms are either undefined or equal

The category $\mathcal{P}_A$ of partial equivalence relations over $A$ is given by

- **objects** are symmetric and transitive relations $R \subseteq A \times A$ (called PERs); $A/R$ denotes the set of $R$-equivalence classes, i.e. the set of subsets $X \subseteq A$ such that $\exists x \in X \wedge (\forall a \in A. a x \in X \iff a R x)$;

- **arrows** from $R_1$ to $R_2$ are maps $f : A/R_1 \to A/R_2$ with a realizer, i.e. an $r \in A$ such that $\forall X \in A/R_1. \forall x \in X. r x \in f(X)$ ($r \vdash_A f$ for short).

The fact that $(A, \cdot)$ is a partial combinatory algebra ensures that identity maps are realizable, composition of realizable maps is realizable, $\mathcal{P}_A$ is locally cartesian closed and has equalizers and finite colimits.

Many definitions and results of Category Theory can be recast in a topos (see [Joh77]) or even in a locally cartesian closed category $\mathcal{E}$. Moreover, through the global section functor $\Gamma : \mathcal{E} \to \text{Set}$, every internal category in $\mathcal{E}$ induces a small category (or equivalently an internal category in $\text{Set}$). The small category $\mathcal{P}_A$ is obtained in this way from an internal category $\mathcal{C}$ in a realizability topos $\mathcal{R}_A$ (see [Hyk88, AL90]). In $\mathcal{R}_A$ one can recast the construction of the strict monoidal category of endofunctors (see Example 2.13) and apply it to $\mathcal{C}$. The result is an internal category $\text{Endo}(\mathcal{C})$, and through the global section functor one gets the category $\text{Endo}(\mathcal{P}_A)_r$ of realizable endofunctors and realizable natural transformations. We give a direct description of $\text{Endo}(\mathcal{P}_A)_r$ as the subcategory of $\text{Endo}(\mathcal{P}_A)$ such that

- **objects** are endofunctors $F : \mathcal{P}_A \to \mathcal{P}_A$ with a realizer, i.e. an $r \in A$ such that $a \vdash_A f$ implies $r a \vdash_A F(f)$ for every $a \in A$ and arrow $f$ in $\mathcal{P}_A$.
arrows from $F$ to $G$ are natural transformations $\tau : F \to G$ with a realizer, i.e. an $r \in A$ such that $r \vdash_A \tau_R$ for every object $R$ of $\mathcal{P}_A$.

$\text{Endo}(\mathcal{P}_A)_r$, inherits the (strict) monoidal structure of $\text{Endo}(\mathcal{P}_A)$, because realizable endofunctors and realizable natural transformations are closed w.r.t. identities and composition. Therefore the inclusion of $\text{Endo}(\mathcal{P}_A)_r$ into $\text{Endo}(\mathcal{P}_A)$ is a strict monoidal functor. $\text{Endo}(\mathcal{P}_A)_r$, unlike $\text{Endo}(\mathcal{P}_A)$, has free monoids and exponentials (this is because the completeness and cocompleteness properties of the internal category, which induces $\mathcal{P}_A$). We give a concrete description of an exponential $ev: H \otimes F \to G$ for a pair realizable of functors $F$ and $G$:

- An arrow $R \vdash F \to S$ in $\mathcal{P}_A$ induces a realizable natural transformation $Y(f) : Y_\ast \to Y_R$ such that $Y(f)_R = T^f$. Therefore, when $Y_R \otimes F \vdash G$ is realizable, also $Y_\ast \otimes F \vdash Y(f)_R \otimes F \vdash G$ is. This induces a function $H(f) : A/H(R) \to A/H(S)$, and by elementary considerations one can give an $a \in A$ such that $a R \vdash_A H(f)$ whenever $r \vdash_A f$.

- $ev: H \otimes F \to G$ is given by $ev_R([a]) = \tau_R([l])$, where $\tau : Y_{FR_\ast} \otimes F \vdash G$ is the natural transformation realized by $a$ and $l$ is the interpretation of the identity combinator, i.e. $\lambda x = x$.

Example 2.16 Consider system $F\omega$ with $\beta\eta$-equivalence (see [Bar92, Gha96]). We define the strict monoidal category $\mathcal{E}_{F\omega}$ of endofunctors and natural transformations expressible in $F\omega$. Most results in [Jas09] can be recast as category-theoretic properties of $\mathcal{E}_{F\omega}$. For convenience, we recall the syntax of $F\omega$

- kinds $k ::= \ast | k \to k$
- type constructors $U ::= X | U \to U | \forall X: k.U | \lambda X: k.U | U U$
- terms $e ::= x | \lambda x: U. e | e e | \Lambda X: k. e | e U$

and introduce some notational conventions: we write $e_U$ for $e_U$ (polymorphic instantiation) and we write definitions $f_X(x : A) \equiv t$ for $f \equiv \Lambda X : \ast. \lambda x : A.t$.

Objects are expressible endofunctors, i.e. pairs $\vec{F} = (F, \text{map}^F)$ with $F : \ast \to \ast$ closed type constructor and $\text{map}^F : \forall X,Y: \ast. (X \to Y) \to FX \to FY$ closed term such that the following $\beta\eta$-equivalences hold

$$\text{map}^F_{X,X}(\text{id}_X) = \text{id}_{FX} : FX \to FX$$
$$\text{map}^F_{X,Z}(g \circ f) = (\text{map}^F_{X,Y} g) \circ (\text{map}^F_{X,Y} f) : FX \to FZ$$

where, $\text{id}_X \equiv \lambda x : X.x$ is the identity on $X$ and $g \circ f \equiv \lambda x : X.g(fx)$ is the composition of $g : Y \to Z$ and $f : X \to Y$.
Example 2.17

We define the strict monoidal category objects are pairs $(P, R)$ of closed terms $\tau : \forall X : \ast, FX \to GX$ such that such that the following $\beta\eta$-equivalence holds

$$(\text{map}_{X,Y}^G f) \circ \tau_A = \tau_B \circ (\text{map}_{X,Y}^F f) : FX \to GY$$

Identity on $\hat{F}$ is the $\beta\eta$-equivalence class of $\iota_F \triangleq \lambda X : \ast, \lambda x : FX. x$, and composition of $[\sigma]$ and $[\tau]$ is $[\sigma] \circ [\tau] \triangleq [\lambda X : \ast, \sigma_X \circ \tau_X]$.

tensor $\hat{G} \circ \hat{F}$ is $(G \circ F, \text{map})$ with $\text{map}_{X,Y} (f : X \to Y) \equiv \text{map}_{X,Y}^G (\text{map}_{X,Y}^F f)$.

unit is the pair $(\text{Id}, \text{map})$ with $\text{Id} \triangleq \lambda X : \ast, X$ and $\text{map}_{A,B} (f : A \to B) \triangleq f$.

Unfortunately, some results in [Jas99] are false. For instance, let $\hat{K}$ and $\hat{M}$ be the expressive functors such that $KX \equiv \forall Z : \ast, (X \to Z) \to Z$ and $MX \equiv X$, then from $\forall X : \ast, KX \to MX$ given by from $X \cdot (c : KX) \equiv c_X (\text{id}_X)$ is not a natural transformation from $\hat{K}$ to $\hat{M}$ (as claimed in [Jas99, Proposition 14]). In fact, naturality of from amount to say that $c : KX f : X \to Y \vdash f(c_X \text{id}_X) = c_Y f : Y$ is a $\beta\eta$-equivalence, but this is impossible, because the two terms are different $\beta\eta$-normal forms.

This failure is related to the lack of weak exponentials in $\hat{E}_{F,\omega}$ (Example 2.17 gives an alternative construction, which avoids these pitfalls). More specifically, when $\hat{G}$ is the the identity functor and $\hat{F}$ is the constant functor $FX = A$ (for some closed type $A$), there are no natural transformations from $\hat{G} \circ \hat{F}$ to $\hat{G}$.

In the sequel we confuse $\beta\eta$-equivalences class with their elements, when it is safe to do so, and use the following auxiliary notation:

- $T$ is the set of $\beta\eta$-equivalence classes of closed types $A$;
- $E(A)$ is the set of $\beta\eta$-equivalence classes of closed terms $e$ of type $A \in T$;
- $P(A)$ is the set of PERs on $E(A)$; given $R \in P(A)$ we denote with $E(R)$ the set of $R$-equivalence classes, i.e. the set of subsets $X \subseteq E(A)$ such that $\exists e \in X \wedge (\forall e' \in E(A). e' \in X \iff e' R e)$.

The category $\mathcal{P}_{F,\omega}$ is given by

**objects** are pairs $(A, R)$ with $A \in T$ and $R \in P(A)$;
arrows from \((A_1, R_1)\) to \((A_2, R_2)\) are \(f : E(R_1) \rightarrow E(R_2)\) with a realizer \(r \vdash f\), i.e. \(r \in E(A_1 \rightarrow A_2)\) such that \(\forall X \in E(R_1), \forall e \in X. r e \in f(X)\).

Identity maps and composition are defined in the obvious way. Moreover, \(\mathcal{P}_{F, \omega}\) is cartesian closed and has equalizers and finite colimits.

The category \(\text{Endo}(\mathcal{P}_{F, \omega})\) of endofunctors and natural transformations realizable in \(F\omega\) is the sub-category of \(\text{Endo}(\mathcal{P}_{F, \omega})\) such that

- **objects** are endofunctors \(F : \mathcal{P}_{F, \omega} \rightarrow \mathcal{P}_{F, \omega}\) with a realizer \(\hat{F} \vdash F\), i.e. \(F\) is a pair \((F, \text{map}^F)\) with \(\hat{F} : * \rightarrow *\) closed type constructor (uniquely determined by \(F\) modulo \(\beta\eta\)-equivalence) such that \(F(A, R) = (B, S)\) implies \(B = \hat{F}A\) and \(\text{map}^F \in E(\forall X, Y : *. (X \rightarrow Y) \rightarrow \hat{F}X \rightarrow \hat{F}Y)\) such that \(f : (A, R) \rightarrow (B, S)\) in \(\mathcal{P}_{F, \omega}\) and \(e \vdash f\) implies \(\text{map}^F_{A,B} e \vdash F(f)\);

- **arrows** from \(F\) to \(G\) are natural transformations \(\tau : F \rightarrow G\) with a realizer \(r \vdash \tau\), i.e. \(r \in E(\forall X : *. \hat{F}X \rightarrow \hat{G}X)\) such that \(r_A \vdash \tau_{(A, R)}\) for any \((A, R)\).

\(\text{Endo}(\mathcal{P}_{F, \omega})\), inherits the (strict) monoidal structure of \(\text{Endo}(\mathcal{P}_{F, \omega})\), and the inclusion functor is strict monoidal. We show (by analogy with Example 2.15) that \(\text{Endo}(\mathcal{P}_{F, \omega})\) has an exponential \(ev : H \otimes F \rightarrow G\) for any \(F\) and \(G\).

- \(H(A, R) \triangleq (\forall Z : *. (A \rightarrow \hat{F}Z) \rightarrow \hat{G}Z, S)\) with \(a S b \iff a\) and \(b\) are realizers for the same natural transformation \(\tau : Y_{(A, R)} \otimes F \rightarrow G\), where \(Y_{(A, R)}\) is the realizable endofunctor \(-_{(A, R)}\) given by exponentiation to \((A, R)\) in \(\mathcal{P}_{F, \omega}\).

- As realizer for \(H\) we take \((\hat{H}, \text{map}^H)\) with \(\hat{H}X \triangleq (\forall Z : * . (A \rightarrow \hat{F}Z) \rightarrow \hat{G}Z\) and \(\text{map}^H_{X,Y}(f : X \rightarrow Y, c : \hat{H}X) \triangleq \Lambda Z : *. \lambda k : Y \rightarrow \hat{F}Z. cZ(k \circ f)\). In particular, \(\text{map}^H\) determines the action of \(H\) of arrows in \(\mathcal{P}_{F, \omega}\).

- \(ev : H \otimes F \rightarrow G\) is realized by \(r \in E(\forall X. \hat{H}(\hat{F}X) \rightarrow \hat{G}X)\) given by \(rX(c : H(\hat{F}X)) \equiv cX(\hat{F}(\hat{F}X))\).

**Example 2.18** If \(\hat{E}\) is a (small) monoidal category, then the category \(\text{Endo}(\hat{E})\) of strong endofunctors over \(\hat{E}\) forms a strict monoidal category \((\text{Endo}(\hat{E}), \circ, \text{Id})\), more precisely

- **objects** are \(\hat{F} = (F, t^F)\) with \(F : \mathcal{E} \rightarrow \mathcal{E}\) functor \(t^F_{a,b} : a \otimes Fb \rightarrow F(a \otimes b)\) and natural transformation such that

  \[
  \begin{align*}
  I \otimes Fa & \rightarrow F(I \otimes a) & a \otimes (b \otimes Fc) & \rightarrow a \otimes F(b \otimes c) & F(\alpha) \\
  F(\lambda) & \downarrow & \downarrow & \downarrow & F(\alpha) \\
  Fa & \rightarrow (a \otimes b) \otimes Fc & \rightarrow F((a \otimes b) \otimes c) \\
  \end{align*}
  \]
arrows from $\hat{F}$ to $\hat{G}$ are natural transformations $\tau : F \rightarrow G$ such that

$$
\begin{array}{c}
\begin{array}{c}
a \otimes Fb \\
\downarrow t^F
\end{array}
\xrightarrow{\text{id} \otimes \tau}
\begin{array}{c}
a \otimes Gb \\
\downarrow t^G
\end{array}
\end{array}
$$

$$
F(a \otimes b) \xrightarrow{\tau} G(a \otimes b)
$$

**tensor** $\hat{G} \circ \hat{F}$ is the pair $(G \circ F, t)$ with

$$
t_{a,b} \hat{=} a \otimes G(Fb) \xrightarrow{t^G} G(a \otimes Fb) \xrightarrow{G(t^F)} G(F(a \otimes b))
$$

**unit** $\hat{\text{id}}$ is the pair $(\text{id}, t)$ with $t_{a,b} \hat{=} \text{id}_{a \otimes b}$.

Moreover, the forgetful functor $U : \text{Endo}(\hat{\mathcal{E}})_m \longrightarrow \text{Endo}(\mathcal{E})$, mapping $\hat{F}$ to $F$, is strict monoidal. Also the category $\text{Endo}(\hat{\mathcal{E}})_m$ of **monoidal endofunctors** forms a strict monoidal category.

**Example 2.19** Given a monoidal category $\hat{\mathcal{E}}$ with $J$-limits, i.e. limits for diagrams of shape $J$, we write $\text{Lim}_J(\hat{\mathcal{E}})$ for the full sub-category of $\mathcal{E}$ whose objects $a \in \mathcal{E}$ **preserve $J$-limits**, i.e. the functor $a \otimes - : \mathcal{E} \longrightarrow \mathcal{E}$ preserves $J$-limits. This sub-category inherits the monoidal structure from $\mathcal{E}$, in fact

- $I$ preserves $J$-limits, because $I \otimes -$ is isomorphic (through $\lambda$) to the identity functor on $\mathcal{E}$, and the identity functor preserves all limits;

- if $a$ and $b$ preserve $J$-limits, then so does $a \otimes b$, because $(a \otimes b) \otimes -$ is isomorphic (through $\alpha$) to $a \otimes (b \otimes -)$, which is the composition of the $J$-limits preserving functors $a \otimes -$ and $b \otimes -$.

When $\mathcal{C}$ is a (small) category with $J$-limits, then the (strict) monoidal category $\hat{\mathcal{E}}$ of endofunctors over $\mathcal{C}$ has $J$-limits (see Example 2.13), and $\text{Lim}_J(\hat{\mathcal{E}})$ is exactly the category of endofunctors on $\mathcal{C}$ preserving $J$-limits in $\mathcal{C}$.

### 3. Operations and Lifting

Given a monoidal category $\hat{\mathcal{E}}$, we introduce several classes of **operations** associated to a monoid in $\hat{\mathcal{E}}$, and define what it means to **lift** such operations along a monoid morphism. In this section, we prove that lifting exists and is unique, when restricting to **algebraic operations**. In the following section, we establish lifting results for wider classes of operations.

**Definition 3.1 (Operations).** *Given a functor $H : \text{Mon}(\hat{\mathcal{E}}) \longrightarrow \mathcal{E}$, an $H$-operation for the monoid $M = (M, e, m)$ is a map $HM \xrightarrow{\text{op}} M$ in $\mathcal{E}$.*
A first-order operation for $\hat{M}$ of signature $S \in \mathcal{E}$ is a map $S \otimes M \xrightarrow{\text{op}} M$, i.e. op is an $H$-operation for the functor $H(-) = S \otimes U(-)$, and such op is called algebraic when (the following diagram commutes)

$$
\begin{array}{ccc}
S \otimes (M \otimes M) & \xrightarrow{\alpha} & (S \otimes M) \otimes M \\
\downarrow & & \downarrow \\
S \otimes M & \xrightarrow{\text{op}} & M \\
\end{array}
$$

Definition 3.2 (Lifting). Given an $H$-operation $H\hat{M} \xrightarrow{\text{op}_M} M$ for $\hat{M}$ and a monoid morphism $h : M \longrightarrow \hat{N}$, an $H$-operation $H\hat{N} \xrightarrow{\text{op}_N} N$ for $\hat{N}$ is a lifting of op along $h$ when

$$
\begin{array}{ccc}
H\hat{N} & \xrightarrow{\text{op}_N} & N \\
\uparrow & & \uparrow \\
H(h) & & h \\
\downarrow & & \downarrow \\
H\hat{M} & \xrightarrow{\text{op}_M} & M \\
\end{array}
$$

Proposition 3.3. Algebraic operations $S \otimes M \xrightarrow{\text{op}} M$ for $\hat{M} = (M, e, m)$ are in bijective correspondence with maps $S \xrightarrow{\text{op}'_M} M$, i.e. $H$-operations for the functor $H(-) = S$, given by

$$
\text{op}' \triangleq S \xrightarrow{\rho^{-1}} S \otimes I \xrightarrow{\text{id} \otimes e} S \otimes M \xrightarrow{\text{op}} M
$$

$$
\text{op} \triangleq S \otimes M \xrightarrow{\text{op}'_M \otimes \text{id}} M \otimes M \xrightarrow{m} M
$$

Theorem 3.4 (Unique algebraic lifting). If $h : \hat{M} \longrightarrow \hat{N}$ is a monoid morphism, and $S \otimes M \xrightarrow{\text{op}_M} M$ is an algebraic operation for $\hat{M}$, then there is a unique $S \otimes N \xrightarrow{\text{op}_N} N$ which is algebraic for $\hat{N}$ and a lifting of op$^M$ along $h$.

Proof Use Proposition 3.3 and replace op$^M$ and op$^N$ with op$^{M'}$ and op$^{N'}$. □

Remark 3.5 An operation associated to $\hat{M}$ may fail to be an $H$-operation, but can be transformed into one. For instance, when $\mathcal{E}$ has the dual of exponentials $G \xrightarrow{\text{ev}} G_F \otimes F$ (see Definition 2.3), then there is a bijection between operations $H'M \longrightarrow M \otimes F$ and $H$-operations $H\hat{M} \longrightarrow M$ with $H(-) \triangleq (H'(-))_F$. 
3.1. Examples

In this section we give examples of (strong) monads on \( \textbf{Set} \) and associated operations, saying explicitly whether the operations are algebraic, first-order or more general instances of \( H \)-operations. In the cartesian closed category \( \textbf{Set} \) of sets monads coincide with strong monads, since every endofunctor on \( \textbf{Set} \) is strong, more precisely \( U : \text{Endo}(\textbf{Set}) \to \text{Endo}(\textbf{Set}) \) of Example 2.18 is an isomorphism. In other cartesian closed categories this is not the case, and there are good reasons to prefer strong monads (see [Mog89, Mog91]).

There are equivalent ways of defining strong monads on a cartesian closed category \( \mathcal{C} \), we borrow the definition adopted in Haskell, and freely use simply typed lambda-calculus as \textit{internal language} to denote objects and maps in \( \mathcal{C} \).

**Definition 3.6 (Strong Monad).** A strong monad on a cartesian closed category \( \mathcal{C} \) is a triple \( \mathcal{M} = (M, \text{ret}^M, \text{bind}^M) \) consisting of

- a map \( M : |\mathcal{C}| \to |\mathcal{C}| \) on the objects of \( \mathcal{C} \)
- a family \( \text{ret}^M_X : X \to MX \) of maps with \( X \in \mathcal{C} \)
- a family \( \text{bind}^M_{X,Y} : MX \times (MY)^X \to MY \) of maps with \( X,Y \in \mathcal{C} \)

such that for every \( a : A, f : (MB)^A, u : MA \) and \( g : (MC)^B \)

\[
\begin{align*}
\text{bind}^M_{A,B}(\text{ret}^M_A(a), f) &= fa \\
\text{bind}^M_{A,A}(u, \text{ret}^M_A) &= u \\
\text{bind}^M_{A,C}(u, \lambda a : A. \text{bind}^M_{B,C}(f a, g)) &= \text{bind}^M_{B,C}(\text{bind}^M_{A,B}(u, f), g)
\end{align*}
\]

A strong monad morphism \( \tau : \tilde{M} \to \tilde{N} \) is a family \( \tau_X : MX \to NX \) of maps with \( X \in \mathcal{C} \) such that for every \( a : A, u : MA \) and \( f : (MB)^A \)

\[
\begin{align*}
\tau_A(\text{ret}^M_A(a)) &= \text{ret}^N_A(a) \\
\tau_B(\text{bind}^M_{A,B}(u, f)) &= \text{bind}^N_{B,C}(\tau_A u, \lambda a : A. \tau_B(f a))
\end{align*}
\]

**Example 3.7** The monad \( \tilde{M} = (M, \text{ret}^M, \text{bind}^M) \) of continuations in \( R \) is

\[
\begin{align*}
MX &= R(X) \\
\text{ret}^M_X(x : X) &= \lambda k : R.X. k x \\
\text{bind}^M_{X,Y}(m : MX, f : MY^X) &= \lambda k : R.Y. m(\lambda x : X. f x k)
\end{align*}
\]

It has two algebraic operations, one for the functor \( S_{\text{abort}} X = R \) and the other for the functor \( S_{\text{callcc}} X = X(R^X) \), namely

\[
\begin{align*}
\text{abort}_X(r : R) &= \lambda k : R.X. r \\
\text{callcc}_X(f : (MX)(R^{MX})) &= \lambda k : R.X. f(\lambda t : MX. t k) k
\end{align*}
\]

Usually the associated operation is \( \text{callcc}_{X,Y} : (MX)^{(MY)^X} \to MX \), which is \textit{definable} from \( \text{callcc}, \text{abort} \), unit and bind of the monad (see [Jas09]).
Example 3.8 The monad $\hat{M} = (M, \text{ret}^M, \text{bind}^M)$ of environments in $S$ is

$$
\begin{align*}
MX & \equiv X^S \\
\text{ret}^M_X(x : X) & \equiv \lambda s : S.x \\
\text{bind}^M_{X,Y}(m : MX, f : MY^X) & \equiv \lambda s : S. f (m s) s
\end{align*}
$$

It has an algebraic operation for the functor $S_{\text{read}}X = X^S$ and a first-order operation (but not algebraic) for the functor $S_{\text{local}}X = S^S \times X$, namely

$$
\begin{align*}
\text{read}_X(f : (MX)^S) & \equiv \lambda s : S. s f s s \\
\text{local}_X(f : S^S, t : MX) & \equiv \lambda s : S. t (f s)
\end{align*}
$$

Example 3.9 The monad $\hat{M} = (M, \text{ret}^M, \text{bind}^M)$ of side-effects on $S$ is

$$
\begin{align*}
MX & \equiv (X \times S)^S \\
\text{ret}^M_X(x : X) & \equiv (x, 0) \\
\text{bind}^M_{X,Y}(m : MX, f : MY^X) & \equiv \lambda s : S. \text{let} (a, s') = m s \text{ in } f a s'
\end{align*}
$$

It has two algebraic operations, one for the functor $S_{\text{read}}X = X^S$ and the other for the functor $S_{\text{write}}X = S \times X$, namely

$$
\begin{align*}
\text{read}_X(k : (MX)^S) & \equiv \lambda s : S. k s s \\
\text{write}_X(s : S, m : MX) & \equiv \lambda s' : S. m s
\end{align*}
$$

Example 3.10 The monad $\hat{M} = (M, \text{ret}^M, \text{bind}^M)$ of complexity on a monoid $(W, 0, +)$ in $\text{Set}$ is

$$
\begin{align*}
MX & \equiv X \times W \\
\text{ret}^M_X(x : X) & \equiv (x, 0) \\
\text{bind}^M_{X,Y}((x, w) : MX, f : MY^X) & \equiv \lambda s : S. \text{let } (y, w') = f x \text{ in } (y, w + w')
\end{align*}
$$

It has an algebraic operation for the functor $S_{\text{add}}X = X \times W$ and $H$-operations for the functors $H_{\text{collect}}_{\text{A}} MX = MA \times X^{(A \times W)}$, namely

$$
\begin{align*}
\text{add}_X(t : MX, w : W) & \equiv \lambda s : (X \times W). s t \\
\text{collect}_{\text{A}}_{X,Y}(t : MA, f : X^{(A \times W)}) & \equiv \lambda s : (X, W). t f s
\end{align*}
$$

Usually the associated operation is $\text{collect}_X : MX \longrightarrow M(X \times W)$, which is definable from the operations $\text{collect}_{\text{A}}$, unit and bind of the monad.

Example 3.11 The monad $\hat{M} = (M, \text{ret}^M, \text{bind}^M)$ of exceptions in $E$ is

$$
\begin{align*}
MX & \equiv X + E \\
\text{ret}^M_X(x : X) & \equiv \text{inl} x \\
\text{bind}^M_{X,Y}(m : MX, f : MY^X) & \equiv \lambda s : S. f m s
\end{align*}
$$
It has an algebraic operation for the functor \( S_{\text{throw}}X = E \) and a first-order operation (but not algebraic) for the functor \( S_{\text{handle}}X = X \times X^E \), namely

\[
\text{throw}_X (e : E) \triangleq \text{inr} \ e \\
\text{handle}_X : (m : MX, \ h : (MX)^E) \triangleq [\text{inl}, \ h](m)
\]

**Example 3.12** Algebraic theories [Man76] are presented by operations and equations. They are a common source of monads with associated operations, since every algebraic theory induces a monad on \( \text{Set} \). A more general way to get monads, not pursued here, is through the **equational systems** of [FH09].

An **algebraic signature** \( \Sigma \) consists of a set \( O \) of operations and a function \( \# \) assigning to each \( o \in O \) its arity \( \#o \in \text{Set} \) (\( \Sigma \) is called **finitary** when each \( \#o \) is finite). A signature induces an endofunctor \( \Sigma X = \bigsqcup_{o \in O} X^{\#o} \), which allows to give a concise definition of \( \Sigma \)-**algebra** and \( \Sigma \)-**homomorphism**. Given an endofunctor \( F \) (on \( \text{Set} \)) the category \( F\text{-Alg} \) of \( F \)-algebras is

- **objects** are \( F \)-algebras \( A = (A, \alpha) \), i.e. a set \( A \) (the carrier) and a map \( FA \xrightarrow{\alpha} A \) (the interpretation of the operations \( o \in O \) when \( F = \Sigma \))
- **arrows** from \( A_1 \) to \( A_2 \) are maps \( h : A_1 \rightarrow A_2 \) such that

\[
\begin{array}{ccc}
FA_1 & \xrightarrow{Fh} & FA_2 \\
\downarrow \alpha_1 & & \downarrow \alpha_2 \\
A_1 & \xrightarrow{h} & A_2
\end{array}
\]

Identities and composition are inherited from \( \text{Set} \).

There is an obvious forgetful functor \( U_F : F\text{-Alg} \rightarrow \text{Set} \), mapping \( A \) to \( A \). When \( \Sigma \) is the endofunctor induced by an algebraic signature, \( U_\Sigma \) has a left adjoint \( T_\Sigma \) (\( T_\Sigma X \) is called the free \( \Sigma \)-algebra over \( X \), and an element \( t \) in its carrier is called a \( \Sigma \)-term with free variables in \( X \)). The monad induced by the adjunction \( U_\Sigma \vdash T_\Sigma \) is the free monad \( \Sigma^* \) over \( \Sigma \) (see Example 2.10), and \( \Sigma\text{-Alg} \) is isomorphic to the category \( \text{Set}^{\Sigma^*} \) of Eilenberg-Moore algebras for \( \Sigma^* \) (see [ML71]).

An **algebraic theory** \( T = (\Sigma, Eq) \) consists of an algebraic signature \( \Sigma \) and a set \( Eq \) of equations between \( \Sigma \)-terms (with free variables in some set \( X \)). The theory \( T \) induces a full sub-category \( T\text{-Alg} \) of \( \Sigma\text{-Alg} \), whose objects are the \( \Sigma \)-algebras **satisfying** all the equations in \( Eq \). Also in this case there is a forgetful functor \( U_T : T\text{-Alg} \rightarrow \text{Set} \) (the restriction of \( U_\Sigma \) to \( T\text{-Alg} \)), which has a left adjoint. The monad \( M_T \) on \( \text{Set} \) induced by the adjunction is a quotient monad of \( \Sigma^* \) (see Example 2.11), and \( T\text{-Alg} \) is isomorphic to \( \text{Set}^{M_T} \) (see [ML71]).

All monads given in this section, except that in Example 3.7, are induced by algebraic theories. Moreover, all monads for **collection types** (such as lists, bags, sets) arise from balanced finitary algebraic theories (see [Man98]).
• The monad of Example 3.8 $MX = X^S$ corresponds to $T_{\text{env}}$ given by $O = \{\text{read}\}, \#\text{read} = S$ and equations

$$t = \text{read}(t \mid i \in S)$$

$$\text{read}(\text{read}(t_{i,j} \mid j \in S) \mid i \in S) = \text{read}(t_{i,i} \mid i \in S)$$

• The monad of Example 3.9 $MX = (X \times S)^S$ corresponds to $T_{\text{state}}$ given by $T_{\text{env}}$ extended with $O = \{\text{write}_a \mid s \in S\}, \#\text{write}_a = 1$ and equations

$$\text{read}(t_i \mid i \in S) = \text{read}(\text{write}_i(t_i) \mid i \in S)$$

$$\text{write}_i(\text{read}(t_j \mid j \in S)) = \text{write}_i(t_i) \text{ with } i \in S$$

$$\text{write}_i(\text{write}_j(t)) = \text{write}_{i+j}(t) \text{ with } i, j \in S$$

• The monad of Example 3.10 $MX = X \times W$ corresponds to $T_{\text{list}}$ given by $O = \{\text{add}_w \mid w \in W\}, \#\text{add}_w = 1$ and equations

$$\text{add}_0(t) = t$$

$$\text{write}_i(\text{write}_j(t)) = \text{write}_{i+j}(t) \text{ with } i, j \in W$$

• The monad of Example 3.11 $MX = X + E$ corresponds to $T_{\text{exc}}$ given by $O = \{\text{throw}_e \mid e \in E\}, \#\text{throw}_e = 0$ and no equations

• The list monad $MX = X^*$ corresponds to $T_{\text{list}}$ given by $O = \{\text{nil}, \text{app}\}, \#\text{nil} = 0, \#\text{app} = 2$ and equations

$$\text{app}(\text{nil}, t) = t = \text{app}(t, \text{nil})$$

$$\text{app}(\text{app}(t_1, t_2), t_3) = \text{app}(t_1, \text{app}(t_2, t_3))$$

• The (finite) set monad corresponds to $T_{\text{list}}$ extended with the equations

$$\text{app}(t_1, t_2) = \text{app}(t_2, t_1)$$

$$\text{app}(t, t) = t$$

The monad $\hat{M}$ induced by an algebraic theory $T = (\Sigma, \text{Eq})$ has an associated algebraic operation $\text{op}_X : \Sigma(MX) \to MX$ of signature $\Sigma$, where $\text{op}_X$ is the $\Sigma$-algebra structure on $MX$. When $\hat{M}$ is the free monad $\Sigma^*$, i.e. $T = (\Sigma, \emptyset)$, one can associate to $\hat{M}$ two other operations

• $\text{elim}_X : X^{\Sigma X} \times X^A \to X^{\Sigma(MA)}$ captures initiality of $MA$ among the $\Sigma$-algebras over $A$, namely $\text{elim}_X(\alpha, f)$ is the unique $\Sigma$-homomorphism $f^*$ from $\Sigma(MA) \stackrel{\text{op}_A}{\to} MA$ (the free algebra over $A$) to $\Sigma X \xrightarrow{\alpha} X$ such that $f^* \circ \text{ret}^M_\alpha = f$. $\text{elim}$ generalizes $\text{bind}^M_{A,X}$ (see the try construct in [PP09]), and usually cannot be presented as an $H$-operation.

• $\text{case}_X : MA \times X^A \times X^{\Sigma(MA)} \to X$ does case analysis on $MA$, which is isomorphic to $A + (\Sigma(MA))$. The instance of $\text{case}$ obtained by replacing $X$ with $MX$, i.e. $\text{case}_X : (MX)^A \times (MX)^{\Sigma(MA)} \to MX$, can be presented as an $H$-operation for $HNX = NA \times (NX)^A \times (NX)^{\Sigma(MA)}$, provided the $M$ in contravariant position is fixed.
4. Monoid Transformers

This section introduces a taxonomy of monoid transformers in the setting of a monoidal category \( \hat{E} \), gives examples of monoid transformers motivated by the incremental approach to monadic semantics, and provides solutions to the lifting problem depending on where a monoid transformer fits in the taxonomy. Given a category \( \mathcal{A} \), we could define a transformer on \( \mathcal{A} \) as a gadget

\[
\begin{array}{c}
\mathcal{A}' \\
\downarrow \text{lift}^T \Rightarrow \mathcal{A}
\end{array}
\]

where

- \( \mathcal{A}' \) is a sub-category of \( \mathcal{A} \), e.g. \( \mathcal{A} \) itself or the discrete category \( |\mathcal{A}| \) with the same objects of \( \mathcal{A} \),
- \( \text{In} : \mathcal{A}' \rightarrow \mathcal{A} \) is the inclusion functor,
- \( T : \mathcal{A}' \rightarrow \mathcal{A} \) is a functor, and
- \( \text{lift}^T : \text{In} \Rightarrow T \) is a natural transformation, therefore \( \text{lift}^T_a : a \rightarrow Ta \) is a map in \( \mathcal{A} \) for any \( a \in \mathcal{A}' \).

Monoid transformers are simply transformers on \( \text{Mon}(\hat{E}) \). The minimum requirement on a monoid transformer \( T \) is to map a monoid \( \hat{M} \in \text{Mon}(\hat{E}) \) to a monoid \( T\hat{M} \) (and a monoid morphism \( \hat{M} \rightarrow T\hat{M} \)). The maximum requirement in the taxonomy is a monoid transformer \( T \) induced by a monoidal endofunctor \( \hat{T} \) on \( \hat{E} \).

**Definition 4.1 (Monoid Transformers).**

1. A monoid transformer is a pair \( (T, \text{lift}^T) \) such that

\[
\begin{array}{c}
\text{Mon}(\hat{E}) \\
\downarrow \text{lift}^T \Rightarrow \text{Mon}(\hat{E})
\end{array}
\]

2. A covariant monoid transformer is a pair \( (T, \text{lift}^T) \) such that

\[
\begin{array}{c}
\text{Id} \\
\downarrow \text{lift}^T \Rightarrow \text{Mon}(\hat{E})
\end{array}
\]

3. A functorial monoid transformer is a covariant monoid transformer
(\(T, \text{lift}^T\)) with an underlying transformer on \(\mathcal{E}\), also denoted \((\hat{T}, \text{lift}^\hat{T})\), i.e.

\[
\begin{array}{c}
\text{Id} \\
\downarrow \text{lift}^T \\
\text{Mon}(\hat{\mathcal{E}}) \\
\downarrow U \\
\vspace{1cm}
\text{Id} \\
\downarrow \text{lift}^T \\
\mathcal{E} \\
\downarrow T
\end{array}
\xrightarrow{\Rightarrow}
\begin{array}{c}
\text{Id} \\
\downarrow \text{lift}^\hat{T} \\
\text{Mon}(\hat{\mathcal{E}}) \\
\downarrow U \\
\vspace{1cm}
\text{Id} \\
\downarrow \text{lift}^T \\
\mathcal{E} \\
\downarrow \hat{T}
\end{array}
\]

in particular \(U(\text{lift}^\hat{T}_M) = \text{lift}^T_M\).

4. A monoidal monoid transformer is a functorial monoid transformer

\((\hat{T}, \text{lift}^\hat{T})\) induced by a transformer \(\hat{\mathcal{E}} \xrightarrow{\text{lift}^\hat{T}} \hat{\mathcal{E}}\) with \(\hat{T}\) a monoidal functor and \(\text{lift}^\hat{T}\) monoidal natural transformation (see Theorem 2.6).

**Proposition 4.2.** The following implications on monoid transformers hold:

\[
\text{monoidal} \Rightarrow \text{functorial} \Rightarrow \text{covariant} \Rightarrow \text{transformer}.
\]

**Proof** Immediate from the definitions and Theorem 2.6 \(\square\)

4.1. Examples

We give examples of strong monad transformers on a cartesian closed category \(\mathcal{C}\), i.e. monoid transformers on the strict monoidal category of strong endofunctors on \(\mathcal{C}\). Some examples require additional assumptions on \(\mathcal{C}\) besides cartesian closure. We give also examples of (strong) monad transformers on \(\text{Set}\), showing that the implications in Proposition 4.2 cannot be reversed.

There are equivalent ways of defining strong endofunctors on a cartesian closed category \(\mathcal{C}\). As already done for strong monads (see Definition 3.6), we borrow the definition adopted in Haskell, and freely use simply typed lambda-calculus as *internal language* to denote objects and maps in \(\mathcal{C}\).

**Definition 4.3 (Strong Endofunctor).** A strong endofunctor on a cartesian closed category \(\mathcal{C}\) is a pair \(\hat{F} = (F, \text{map}^F)\) consisting of

- a map \(F : |\mathcal{C}| \rightarrow |\mathcal{C}|\) on the objects of \(\mathcal{C}\)
- a family \(\text{map}^F_{X,Y} : Y^X \times FX \rightarrow FY\) of maps with \(X, Y \in \mathcal{C}\) such that for every \(u : FA, f : B^A\) and \(g : C^B\):

\[
\begin{align*}
\text{map}^F_{A,A}(\text{id}_A, u) & = u \\
\text{map}^F_{A,C}(g \circ f, u) & = \text{map}^F_{B,C}(g, \text{map}^F_{A,B}(f, u))
\end{align*}
\]
A strong natural transformation $\tau : \hat{F} \rightarrow \hat{G}$ is a family $\tau_X : FX \rightarrow GX$ of maps with $X \in \mathcal{C}$ such that for every $u : FA$ and $f : B^A$

$$\tau_B(\text{map}^F_{A,B}(f, u)) = \text{map}^G_{A,B}(f, \tau_A(u))$$

**Example 4.4** The transformer $(T, \text{lift}^T)$ for adding *environments* in $S \in \mathcal{C}$ is

- $T$ maps a strong monad $\hat{M}$ to the strong monad $\hat{N}$ given by

  $$\begin{align*}
  N X & \doteq MX^S \\
  \text{ret}^N_X(x) & \doteq \lambda s : S. \text{ret}^M_X(x) \\
  \text{bind}^N_{X,Y}(c, f) & \doteq \lambda s : S. \text{bind}^M_{X,Y}(c s, \lambda x : X. f x s) 
  \end{align*}$$

- $\text{lift}^T$ maps a strong monad $\hat{M}$ to $\tau : \hat{M} \rightarrow T\hat{M}$ given by

  $$\tau_X(c : MX) \doteq \lambda s : S. c$$

This transformer is monoidal. More precisely, it is induced by the following monoidal functor $\hat{T} = (T, \phi_I, \phi)$ and monoidal natural transformation $\text{lift}^T$

- $T$ maps a strong functor $\hat{F}$ to the strong functor $\hat{G}$ given by

  $$\begin{align*}
  GX & \doteq (FX)^S \\
  \text{map}^G_{X,Y}(f, u) & \doteq \lambda s : S. \text{map}^F_{X,Y}(f, us)
  \end{align*}$$

  and maps $\tau : \hat{F}_1 \rightarrow \hat{F}_2$ to $T\tau : T\hat{F}_1 \rightarrow T\hat{F}_2$ given by

  $$(T\tau)_X(u) \doteq \lambda s : S. \tau_X(us)$$

- $\phi_I : \text{Id} \rightarrow T(\text{Id})$ and $\phi_{\hat{F}_2,\hat{F}_1} : T\hat{F}_2 \circ T\hat{F}_1 \rightarrow T(\hat{F}_2 \circ \hat{F}_1)$ are

  $$\begin{align*}
  \phi_{I,X}(x : X) & \doteq \lambda s : S. x \\
  \phi_{\hat{F}_2,\hat{F}_1,X}(u : F_2((F_1X)^S)^S) & \doteq \lambda s : S. \text{map}^F_{F_2X}((F_1X)^S, (\lambda f : (F_1X)^S. f x s, us))
  \end{align*}$$

- $\text{lift}^T_{\hat{F}} : \hat{F} \rightarrow T\hat{F}$ is $\text{lift}^T_{\hat{F},X}(u : FX) \doteq \lambda s : S. u$

**Example 4.5** The transformer $(T, \text{lift}^T)$ for adding *side-effects* on $S \in \mathcal{C}$ is

- $T$ maps a strong monad $\hat{M}$ to the strong monad $\hat{N}$ given by

  $$\begin{align*}
  N X & \doteq M(X \times S)^S \\
  \text{ret}^N_X(x) & \doteq \lambda s : S. \text{ret}^M_X(x) \\
  \text{bind}^N_{X,Y}(c, f) & \doteq \lambda s : S. \text{bind}^M_{X \times S,Y \times S}(c s, \lambda x : X, s : S. f x s')
  \end{align*}$$
• lift\(\hat{T}\) maps a strong monad \(\hat{M}\) to \(\tau: \hat{M} \xrightarrow{T} \hat{T}\hat{M}\) given by
\[
\tau_X(c : M X) \triangleq \lambda s : S. \text{bind}_X^M(c, \lambda x : X. \text{ret}_X^M(x, s))
\]
This transformer is monoidal. More precisely, it is induced by the following monoidal functor \(\hat{T}\) and monoidal natural transformation lift\(\hat{T}\)

• \(T\) maps a strong functor \(\hat{F}\) to the strong functor \(\hat{G}\) given by
\[
GX \triangleq F(X \times S)^S
\]
\[
\text{map}^G_{X,Y}(f, u) \triangleq \lambda s : S. \text{map}^F_{X \times S, Y \times S}(\lambda(x : X, s' : S). (f \times s'), u \times s)
\]
and maps \(\tau: \hat{F}_1 \xrightarrow{T} \hat{F}_2 \circ T\hat{F}_1 \xrightarrow{T} \hat{T}\hat{F}_2 \circ T\hat{F}_1\) given by
\[
(T\tau)_X(u) \triangleq \lambda s : S. \tau_X(u \times s)
\]

• \(\phi_I: \text{Id} \xrightarrow{T} \text{Id}\) and \(\phi_{\hat{F}_2, \hat{F}_1}: T\hat{F}_2 \circ T\hat{F}_1 \xrightarrow{T} T(\hat{F}_2 \circ \hat{F}_1)\) are
\[
\phi_{\hat{F}_1,X}(x : X) \triangleq \lambda s : S. (x, s)
\]
\[
\phi_{\hat{F}_2, \hat{F}_1,X}(u : (F_2((F_1X \times S)^S \times S)^S) \times S) \triangleq \lambda s : S. \text{map}^F_{F_2(X \times S)^S \times S, F_1(X \times S)}(\lambda(f : F_1(X \times S)^S, s' : S). f \times s', u \times s)
\]

• lift\(\hat{T}: \hat{F} \xrightarrow{T} \hat{T}\hat{F}\) is lift\(\hat{T}: (u : FX) \xrightarrow{\lambda s : S. \text{map}_X^F(c, \lambda x : X. \text{ret}_X^M(x, s)))\)

Example 4.6 The transformer \((T, \text{lift}\hat{T})\) for adding complexity on a monoid \((W, 0, +)\) in \(\mathcal{C}\) is

• \(T\) maps a strong monad \(\hat{M}\) to the strong monad \(\hat{N}\) given by
\[
\begin{align*}
NX & \triangleq M(X \times W) \\
\text{ret}_X^N(x) & \triangleq \text{ret}_X^M(x, 0) \\
\text{bind}_X^N(c, f) & \triangleq \text{bind}_X^M(c, \lambda x : X, w : W).
\end{align*}
\]
\[
\text{bind}_X^M(f x, \lambda y : Y, w' : W). \text{ret}_X^M(y, w + w')
\]

Also this transformer is monoidal, but we skip the details.

Example 4.7 In this example we need additional assumptions on \(\mathcal{C}\), namely

• existence of binary sums \(A_1 \xrightarrow{\text{inl}} A_1 + A_2 \xleftarrow{\text{inr}} A_2\)

\[
\begin{array}{c}
A_1 \\
\downarrow \\
[f_1, f_2] \\
\downarrow \\
A
\end{array}
\]
\[
\begin{array}{c}
A_2 \\
\downarrow \\
\downarrow \\
\downarrow \\
\downarrow \\
A
\end{array}
\]

(we write \(f_1 + f_2\) for the action of \(+\) on maps), and
existence of initial algebras $\alpha_F : F(\mu X. FX) \longrightarrow \mu X. FX$ for every strong endofunctor $\hat{F}$ (for simplicity, we assume that $\alpha_F$ is the identity map).

The last assumption is difficult to satisfy. However, one could take as $C$ the cartesian closed category $\mathcal{P}_A$ of partial equivalence relations, and replace $\text{Endo}(\mathcal{P}_A)_s$ with the more restricted category $\text{Endo}(\mathcal{P}_A)_r$ of realizable endofunctors and realizable natural transformations (see Example 2.15). Given a realizable endofunctor $\hat{S}$, the transformer $(T, \text{lift}^T)$ for adding $\hat{S}$-steps is

- $T$ maps a realizable monad $\hat{M}$ to the realizable monad $\hat{N}$ given by

$$NX \triangleq \mu X'. M(X + SX')$$
$$\text{ret}^N_X(x) \triangleq \text{ret}^M_X + S(NX)(\text{inl} x)$$
$$\text{step}_X : S(NX) \longrightarrow NX$$
$$\text{step}_X(u) \triangleq \text{ret}^M_X + S(NX)(\text{inr} u)$$
$$\text{bind}^N_X,Y(c, f) \triangleq hc$$

where $NX \xrightarrow{h} NY$ is the unique $M(X + S-)$-algebra morphism from the initial algebra to $\beta : M(X + S(NY)) \longrightarrow NY$ given by
$$\beta(c) \triangleq \text{bind}^M_{X + S(NY), Y + S(NY)}(c, [f, \text{step}_Y])$$

- $\text{lift}^T$ maps a realizable monad $\hat{M}$ to $\tau : \hat{M} \longrightarrow \hat{N} = T\hat{M}$ given by
$$\tau(c : MX) \triangleq \text{bind}^M_{X, X + S(NX)}(c, \text{ret}^N_X)$$

This transformer is functorial. More precisely, the underlying realizable endofunctor transformer $(T, \text{lift}^T)$ is

- $T$ maps a realizable functor $\hat{F}$ to the realizable functor $\hat{G}$ given by

$$GX \triangleq \mu X'. F(X + SX')$$
$$\text{map}^G_{X, Y}(f, u) \triangleq hu$$

where $GX \xrightarrow{h} GY$ is the unique $F(X + S-)$-algebra morphism from the initial algebra to $\beta : F(X + S(G(Y))) \longrightarrow GY$ given by
$$\beta(u) \triangleq \text{map}^F_{X + S(GY), Y + S(GY)}(f + \text{id}_{S(GY)}, u)$$

and maps $\tau : \hat{F}_1 \xrightarrow{\text{lift}^T} \hat{F}_2$ to $T\tau : T\hat{F}_1 = \hat{G}_1 \xrightarrow{\text{lift}^T} \hat{G}_2 = T\hat{F}_2$ given by
$$T\tau(x) \triangleq hu$$

where $G_1 X \xrightarrow{h} G_2 X$ is the unique $F_1(X + S-)$-algebra morphism from the initial algebra to $\beta : F_1(X + S(G_2 X)) \longrightarrow G_2 X$ given by
$$\beta(u) \triangleq \tau_{X + S(G_2 X)}(u)$$
• $\text{lift}^T$ maps a realizable endofunctor $\hat{F}$ to $\tau : \hat{F} \longrightarrow \hat{G} = T\hat{F}$ given by

$$\tau_X(u : FX) \triangleq \text{map}^F_{X,X + S(GX)}(\text{inl}, u)$$

As shown in Example 4.9, this transformer may fail to be monoidal.

**Example 4.8** We define the list transformer, which needs additional assumptions, like those identified in Example 4.7. Therefore, we take as $\mathcal{C}$ the cartesian closed category $\mathcal{P}_A$ of partial equivalence relations, and replace $\text{Endo}(\mathcal{P}_A)_s$ with the more restricted category $\text{Endo}(\mathcal{P}_A)_r$ of realizable endofunctors and realizable natural transformations. The list transformer $(T, \text{lift}^T)$ is

• $T$ maps a realizable monad $\hat{M}$ to the realizable monad $\hat{N}$ given by

$$\tau_X : \mu X. M(1 + X \times X') \triangleq \mu X. N(1 + X \times X')$$

$$\text{nil}_X : \mu X. M(1 + X \times X') \triangleq \mu X. N(1 + X \times X')$$

$$\text{cons}_X : X \times N(1 + X \times X') \longrightarrow N(1 + X \times X')$$

$$\text{bind}^N_{X,Y}(c, f) \triangleq hc$$

where $N(1 + X \times X') \longrightarrow N(1 + X \times X')$ is the unique $M(1 + X \times X')$-algebra morphism from the initial algebra to $\beta : M(1 + X \times NY) \longrightarrow NY$ given by

$$\beta(c) \triangleq \text{bind}^M_{1+X \times NY, 1+Y \times NY}(c, [\text{nil}_Y, \lambda (x, l). \text{app}_Y((f x), l)])$$

with $\text{app}_X(\text{nil}_X, l) = l = \text{app}_X(l, \text{nil}_X)$

$$\text{cons}_X : (x,l) \triangleq \text{cons}_X(x, \text{app}_X(l,1))$$

$$\text{app}_X(\text{app}_X(\text{app}_X(l_1, l_2), l_3)) = \text{app}_X(l_1, \text{app}_X(l_2, l_3))$$

To prove that $\text{ret}^N$ and $\text{bind}^N$ satisfy the equations in Definition 3.6, one can use the following properties of $\text{nil}_X$, $\text{cons}_X$ and $\text{app}_X$

$$\text{app}_X(\text{cons}_X(x,l_1,l)) = \text{app}_X(l,\text{nil}_X)$$

$$\text{app}_X(\text{cons}_X(x_1, l_1, l_2)) = \text{cons}_X(x, \text{app}_X(l_1, l_2))$$

$$\text{app}_X(\text{app}_X(l_1, l_2), l_3) = \text{app}_X(l_1, \text{app}_X(l_2, l_3))$$

• $\text{lift}^T$ maps a realizable monad $\hat{M}$ to $\tau : \hat{M} \longrightarrow \hat{N} = T\hat{M}$ given by

$$\tau_X(c : MX) \triangleq \text{bind}^M_{X,1+X \times NX}(c, \text{ret}^N_X)$$

This transformer is functorial. More precisely, the underlying realizable endofunctor transformer $(T, \text{lift}^T)$ is
• $T$ maps a realizable functor $\hat{F}$ to the realizable functor $\hat{G}$ given by

$$GX \xrightarrow{h} GY$$

is the unique $F(1 + X \times \cdot)$-algebra morphism from the initial algebra to $\beta : F(1 + X \times GY) \rightarrow GY$ given by

$$\beta(u) = \text{map}_{1+X \times GY,1+Y \times GY}^F(\text{id}_1 + (f \times \text{id}_{GY}), u)$$

and maps $\tau : \hat{F}_1 \rightarrow \hat{F}_2$ to $T\tau : T\hat{F}_1 = \hat{G}_1 \rightarrow \hat{G}_2 = T\hat{F}_2$ given by

$$(T\tau)_{X}(u) = h u$$

where $G_1X \xrightarrow{h} G_2X$ is the unique $F_1(1 + X \times \cdot)$-algebra morphism from the initial algebra to $\beta : F_1(1 + X \times G_2X) \rightarrow G_2X$ given by

$$\beta(u) = \gamma_{1+X \times G_2X}(u)$$

• $\text{lift}^T$ maps a realizable endofunctor $\hat{F}$ to $\tau : \hat{F} \rightarrow \hat{G} = T\hat{F}$ given by

$$\tau_X(u : FX) = \text{map}_{X,1+X \times GX}^F(\text{inr}', u)$$

where $\text{inr}' : X \rightarrow 1 + X \times GX$ is given by

$$\text{inr}'(x) = \text{inr}(x, \text{map}_{X,1+X \times GX}^F(\lambda - . \text{inl}, u))$$

We conjecture that the list transformer is a quotient of the binary tree transformer, which adds $B$-steps for the functor $B(X) = 1+X \times X$ (see Example 4.7). A more precise statement requires the equational systems of [FH09].

**Example 4.9** We give four (strong) monad transformers on $\textbf{Set}$, which show that the implications in Proposition 4.2 cannot be reversed. When convenient, we use the fact that every endofunctor/monad on $\textbf{Set}$ is strong (see Section 3.1).

1. The transformer $(T, \text{lift}^T)$ for adding continuations is defined as follows, $T$ maps a strong monad $\hat{M}$ to the strong monad $\hat{N}$ of continuations in $MR$ (see Example 3.7)

$$\begin{align*}
NX &\xrightarrow{\lambda k : (MR)^X. k x} (MR)^{(MR)^X} \\
\text{ret}_X^N(x) &\xrightarrow{\lambda k : (MR)^X. k x} (MR)^X \\
\text{bind}_X^N(c, f) &\xrightarrow{\lambda k : (MR)^Y. c(\lambda x : X. f \ x \ k)} (MR)^Y
\end{align*}$$

and $\text{lift}^T$ maps $\hat{M}$ to the morphism $\tau : \hat{M} \rightarrow T\hat{M}$ given by

$$\tau_X(c : MX) = \lambda k : (MR)^X. \text{bind}_{X,R}^M(c, k)$$

This transformer is **not covariant**, because $M$ is used in contravariant position in $NX$. 

2. Given a strong monad \( \hat{M} \), we say that a computation \( c : MX \) is **idempotent** when \( c = c; c \) where \( c_1; c_2 = \text{bind}^\hat{M}_{X,X}(c_1, \lambda x : X. c_2) \).

The transformer \((T, \text{lift}^T)\) making computations idempotent is defined as follows, \( T \) maps a strong monad \( \hat{M} \) to the smallest quotient monad (see Example 2.11) generated by the family of relations

\[
R_X = \{(c, c; c) \mid c \in MX\}
\]

and \( \text{lift}^T \) is the epimorphism from \( \hat{M} \) to the quotient monad.

This transformer is covariant, because \( \tau_X(c; c) = \tau_X(c) \) for any strong monad morphism \( \tau : \hat{M} \to \hat{N} \) and \( c : MX \), but it is **not functorial**. In fact, there are two monads \( \hat{M} \) and \( \hat{N} \) of complexity (see Example 3.10) with the same underlying endofunctor \( F(-) = - \times \text{bool} \), with \text{bool} the set of booleans, such that \( TM = \hat{M} \) and \( T\hat{N} = \text{Id} \):

- \( \hat{M} \) is the strong monad induced by the monoid \((\text{bool}, \text{false}, \text{or})\) in \text{Set}.
  Since this monoid is idempotent, all computations in \( MX \) are already idempotent, therefore \( TM = \hat{M} \).

- \( \hat{N} \) is the strong monad induced by the monoid \((\text{bool}, \text{false}, \text{xor})\) in \text{Set}.
  Since \( \text{xor}(\text{true}, \text{true}) = \text{false} \), the quotient monad \( T\hat{N} \) must identify \((x, \text{false})\) and \((x, \text{true})\) for any \( x : X \) (and this suffices to make all computations idempotent).

3. The transformer \((T, \text{lift}^T)\) for adding **exceptions** in \( E \) is defined as follows, \( T \) maps a strong monad \( \hat{M} \) to the strong monad \( \hat{N} \) given by

\[
\begin{align*}
NX & \doteq M(X + E) \\
\text{ret}^N_X(x) & \doteq \text{ret}^{\hat{M}_{X+E}}_X(\text{inl} x) \\
\text{throw}_X(e : E) & \doteq \text{ret}^{\hat{M}_{X+E}}_X(\text{inr} e) \\
\text{bind}^N_{X,Y}(c, f) & \doteq \text{bind}^{\hat{M}_{X+E,Y+E}}_{X,Y}(c, [f, \text{throw}_X])
\end{align*}
\]

and \( \text{lift}^T \) maps \( \hat{M} \) to the morphism \( \tau : \hat{M} \to T\hat{M} \) given by

\[
\tau_X(c : MX) \doteq \text{bind}^{\hat{M}}_{X,X+E}(c, \text{ret}^N_X)
\]

This transformer is functorial (since it is the instance of Example 4.7 with \( SX = E \)), more precisely \( T \) maps an endofunctor \( F \) to the endofunctor \( F(- + E) \), but it is **not monoidal**. In fact, if it were monoidal, then there should be a natural transformation

\[
\phi_{G,F} : G(F(- + E) + E) \to G(F(- + E)).
\]

However, this is impossible, when \( E = 1, GX = X \) and \( FX = 0 \).

4. The **identity transformer**, which maps \( \hat{M} \) to itself, is monoidal.
4.2. Transformers and Liftings

Theorem 3.4 showed how algebraic operations lift along monoid morphisms. Therefore, given a monoid transformer \((T, \text{lift}_T)\) and a monoid \(\hat{M}\), every algebraic operation \(S \otimes M \overset{\text{op}}{\longrightarrow} M\) for \(\hat{M}\) can be lifted along \(\text{lift}_T^\hat{M}\). We take advantage of the additional structure in functorial and monoidal monoid transformers to provide liftings for more general classes of operations. The results are summarized in Figure 4.2: as one goes from left to right the operations become more general, but the lifting theorems need additional assumptions on the transformers (or the monoidal category, see Theorem 4.15).

To simplify proofs, in the rest of this section we assume that \(\hat{E}\) is a strict monoidal category. However, statements and definitions do not rely on this simplifying assumption. We start with a result for monoidal monoid transformers.

**Theorem 4.10 (Monoidal Lifting).** If \((T, \text{lift}_T)\) is a monoidal monoid transformer with underlying monoidal functor \((T, \phi_I, \phi)\), and \(S \otimes M \overset{\text{op}}{\longrightarrow} M\) is a first-order operation for \(\hat{M}\), then there is a lifting of \(\text{op}\) along \(\text{lift}_T^\hat{M}\) given by

\[
\text{op}^T = S \otimes TM \xrightarrow{\text{lift}_T^S \otimes \text{id}} TS \otimes TM \xrightarrow{\phi} T(S \otimes M) \xrightarrow{T(\text{op})} TM
\]

More generally, if \(H\) is the functor \(H(-) = (S \otimes U(-)) \otimes F\), and \(H\hat{M} \overset{\text{op}}{\longrightarrow} M\) is an \(H\)-operation for \(\hat{M}\), then there is a lifting \(\text{op}^T\) of \(\text{op}\) along \(\text{lift}_T^\hat{M}\) given by

\[
\begin{array}{c}
(TS \otimes TM) \otimes F \xrightarrow{\phi \otimes \text{lift}_T^F} T(S \otimes M) \otimes TF \xrightarrow{\phi} T((S \otimes M) \otimes F)
\end{array}
\]

**Proof** The first-order case reduces to the general case when \(F = I\). To show that \(\text{op}^T \circ (\text{id} \otimes \text{lift}_T^\hat{M} \otimes \text{id}) = \text{lift}_T^\hat{M} \circ \text{op}\) we expand the definition of \(\text{op}^T\) and prove
that the following diagram commutes

\[
\begin{align*}
(TS \otimes TM) \otimes F & \xrightarrow{\phi \otimes \text{lift}_F^T} T(S \otimes M) \otimes TF \xrightarrow{\phi} T((S \otimes M) \otimes F) \\
(S \otimes TM) \otimes F & \xrightarrow{(1)} T(S \otimes M) \otimes F \xrightarrow{T \text{(op)}} TM \\
(S \otimes M) \otimes F & \xrightarrow{(2)} \text{op} \xrightarrow{\text{M}} M
\end{align*}
\]

1. because $\text{lift}_T$ is a monoidal natural transformation.
2. because $\text{lift}_T$ is a natural transformation.

\[\square\]

We now focus on functorial monoid transformers. Before proving the main result (Theorem 4.15), we establish the following lemma.

**Lemma 4.11 (Derived Lifting).** If $(T, \text{lift}_T)$ is a functorial monoid transformer, $\text{op}^N : HN \to N$ is an $H$-operation for $N$, $\text{op}^{N,T} : HT(\hat{N}) \to TN$ is a lifting of $\text{op}^N$ along $\text{lift}_T$, $t : M \to \hat{N}$ is a monoid morphism and $f : N \to M$ is a map, then

- $\text{op}^M \hat{=} H\hat{M} \xrightarrow{\text{H}(t)} H\hat{N} \xrightarrow{\text{op}^N} N \xrightarrow{f} M$ is an $H$-operation for $M$, and
- $\text{op}^{M,T} \hat{=} H(T\hat{M}) \xrightarrow{\text{H}(T(t))} HT(\hat{N}) \xrightarrow{\text{op}^{N,T}} TN \xrightarrow{T(f)} TM$ is a lifting of $\text{op}^M$ along $\text{lift}_M$. 

Proof The following diagram commutes

1. by definition of $\mathsf{op}^M$ and $\mathsf{op}^{M,T}$.
2. because $\text{lift}_T$ is a natural transformation.
3. because $\mathsf{op}^{N,T}$ is a lifting of $\mathsf{op}^N$ along $\text{lift}_N$.

4.2.1. Codensity Lifting

Consider the instance of Lemma 4.11 for $H(-) = S \otimes U(-)$: if $\mathsf{op}^N$ is an algebraic operation for $\hat{N}$, then $\mathsf{op}^M$ is a first-order operation and one gets a lifting $\mathsf{op}^{M,T}$ of $\mathsf{op}^M$ along $\text{lift}_M^T$ by taking as $\mathsf{op}^{N,T}$ the algebraic lifting of $\mathsf{op}^N$ along $\text{lift}_N^T$. We show that every first-order operation $\mathsf{op}^M$ can be defined (as describe in Lemma 4.11) using an algebraic operation $\mathsf{op}^N$, provided the monoidal category $\mathcal{E}$ has exponentials.

Remark 4.12 In the rest of this section we assume $\mathcal{E}$ to have exponentials. In this setting two maps $X \xrightarrow{f_i} G$ are equal iff $X \otimes F \xrightarrow{f_i \otimes \text{id}} G \otimes F \xrightarrow{ev} G$ are equal. The main example of strict monoidal category with exponentials is the category $\text{Endo}(\mathcal{P}_A)$, of realizable endofunctors (see Example 2.15).

Definition 4.13 (Codensity). The codensity monoid transformer $(K, \text{lift}_K)$ is given by

- $K\hat{M} = (M^M, i_M, c_M)$ see Example 2.9
- $\text{lift}_K^M = (M \xrightarrow{\Lambda_m} M^M)$ is a monoid morphism $\hat{M} \longrightarrow K\hat{M}$
Moreover, \( \text{lift}_{M}^{K} \) has a left inverse: \( M^{M} \xrightarrow{\text{down}_{M}} M \), i.e. \( \text{down}_{M} \circ \text{lift}_{M}^{K} = \text{id}_{M} \), given by \( \text{down}_{M} \doteq (M^{M} = M^{M} \otimes I \xrightarrow{id \otimes e} M^{M} \otimes M \xrightarrow{ev} M) \).

**Proof** This definition has some proof obligations, i.e.: \( \Lambda m \) is a monoid morphism and \( \text{down}_{M} \) is a left inverse of \( \text{lift}_{M}^{K} \). Diagrammatically:

\[ I \xrightarrow{e} M \xleftarrow{m} M \otimes M \xrightarrow{\Lambda m \otimes \Lambda m} M^{M} \otimes M^{M} \xrightarrow{\text{down}_{M}} M \]

To prove commutativity of the first diagram we use Remark 4.12.

- \( \Lambda m \) respects the unit of the monoid.

\[ I \otimes M \xrightarrow{e \otimes \text{id}} M \otimes M \xrightarrow{\text{id} \otimes M \otimes \text{id}} M \otimes M \otimes M \]

- \( \Lambda m \) respects the multiplication of the monoid.

\[ M \otimes M \otimes M \xrightarrow{m \otimes \text{id} \otimes \text{id}} M \otimes M \otimes M \xrightarrow{\text{id} \otimes ev} M \otimes M \]

\[ M^{M} \otimes M^{M} \otimes M \xrightarrow{\text{id} \otimes ev} M^{M} \otimes M \xrightarrow{ev} M \]
To prove commutativity of the second diagram we use the definition of $\text{down}_M$, the property of exponentials and the equation $m \circ (\text{id} \otimes e) = \text{id}_M$ for $M$

\[
\begin{array}{c}
M^M \otimes I \xrightarrow{\text{id} \otimes e} M^M \otimes M \xrightarrow{\text{ev}} M \\
\Lambda m \otimes \text{id} \longrightarrow \Lambda m \otimes \text{id} \longrightarrow e \\
M \otimes I \xrightarrow{\text{id} \otimes e} M \otimes M \\
\end{array}
\]

\[\square\]

**Theorem 4.14 (Codensity Properties).** If $\text{op} : S \otimes M \longrightarrow M$ is a first-order operation for $\hat{M}$, then

(a) $\text{op}^K = S \otimes M^M \xrightarrow{\Lambda(\text{op}) \otimes \Lambda m} M^M \otimes M^M \xrightarrow{\text{ev}} M$ is algebraic for $\hat{K} \hat{M}$

(b) $\text{op} = S \otimes M \xrightarrow{\text{id} \otimes \text{lift}_M^K} S \otimes M^M \xrightarrow{\text{op}^K} M^M \otimes M \xrightarrow{\text{down}_M} M$

Moreover, if $\text{op}$ is algebraic, then $\text{op}^K$ is the algebraic lifting of $\text{op}$ along $\text{lift}_M^K$.

**Proof** The operation $\text{op}^K$ is the algebraic operation induced by $S \xrightarrow{\Lambda(\text{op})} M^M$ (see Proposition 3.3), hence item (a) is proved. In order to prove item (b), we expand the definitions and the equation becomes

\[
\begin{array}{c}
S \otimes M = S \otimes M \otimes I \xrightarrow{\text{id} \otimes \text{id} \otimes e} M^M \otimes M^M \otimes I \xrightarrow{\text{id} \otimes \Lambda m \otimes \text{id}} M^M \otimes M^M \otimes M \xrightarrow{\text{id} \otimes \text{ev}} M^M \otimes M \xrightarrow{\text{ev}} M \\
\Lambda(\text{op}) = S \otimes M \xrightarrow{\Lambda(\text{op}) \otimes \text{id}} M^M \otimes M \xrightarrow{\text{id} \otimes \Lambda m \otimes \text{id}} M^M \otimes M^M \otimes M \\
\text{id} \otimes \text{ev} \\
S \otimes M \xrightarrow{\Lambda(\text{op}) \otimes \text{id}} M^M \otimes M \xrightarrow{\text{id} \otimes \text{ev}} M \\
\end{array}
\]

and the proof is given by the following commuting diagram (see Remark 4.12).

Finally, if $\text{op}$ is algebraic, then $\text{op} = S \xrightarrow{\Lambda(\text{op})} M^M$ (see Proposition 3.3). Therefore

\[
\Lambda(\text{op}) = S \xrightarrow{\text{op}^\prime} M \xrightarrow{\Lambda m} M^M
\]

and $\text{op}^K$ is the algebraic operation induced by the lifting of $\text{op}^\prime$ along $\text{lift}_M^K$. $\square$
We now state our main lifting result for functorial monoid transformers.

**Theorem 4.15 (Codensity Lifting).** If \((T, \text{lift}^T)\) is a functorial monoid transformer, and \(\text{op} : S \otimes M \to M\) is a first-order operation for \(\hat{M}\), then there is a lifting of \(\text{op}\) along \(\text{lift}^T_{\hat{M}}\) given by

\[
\text{op}^T = S \otimes TM \xrightarrow{\text{id} \otimes T(\text{lift}^K_{\hat{M}})} S \otimes T(M^M) \xrightarrow{\text{op}^{K,T}} T(M^M) \xrightarrow{T(\text{down}^\hat{M})} TM
\]

where \(\text{op}^{K,T}\) is the unique algebraic lifting of \(\text{op}^K\) along \(\text{lift}^T_{\hat{M}}\).

**Proof** Apply Lemma 4.11 by taking \(\text{op}^M = \text{op}, \hat{N} = K\hat{M}, \text{op}^N = S \otimes N \xrightarrow{\text{op}^K} N,\) thus \(\text{op}^N\) is algebraic for \(\hat{N}\) (by Theorem 4.14), \(t = \text{lift}^K_{\hat{M}}, f = \text{down}^\hat{M},\) and \(\text{op}^{N,T} : S \otimes (TN) \to TN\) the unique algebraic lifting of \(\text{op}^N\) along \(\text{lift}^T_{\hat{N}}\). □

4.3. Coincidence of Liftings

For some combinations of monoid transformers and operations it is possible that two (or more) of the lifting theorems summarized in Figure 4.2 are applicable. For instance, if \(\text{op}\) is an algebraic operation for \(\hat{M}\) and \((T, \text{lift}^T)\) is a monoidal monoid transformer, then one can apply both the algebraic lifting (Theorem 3.4) and the monoidal lifting (Theorem 4.10). We prove that when two lifting theorems are applicable, they yield the same result.

**Theorem 4.16 (Algebraic/Monoidal Coincidence).** When \((T, \text{lift}^T)\) is a monoidal monoid transformer, and \(\text{op} : S \otimes M \to M\) is an algebraic operation for \(\hat{M}\), the monoidal lifting (Theorem 4.10) and the algebraic lifting (Theorem 3.4) of \(\text{op}\) along \(\text{lift}^T_{\hat{M}}\) coincide.

**Proof** Since \(\text{op}\) is an algebraic operation for \(\hat{M} = (M, e, m)\), by Proposition 3.3 exists unique \(\text{op}' : S \to M\) such that \(\text{op} = m \circ (\text{op}' \otimes \text{id})\). Consider the following diagram, where the top path from \(S \otimes TM\) to \(TM\) is the monoidal lifting of \(\text{op}\), and the bottom path is the algebraic lifting of \(\text{op}\). The diagram commutes because of naturality of \(\text{lift}^T\) and \(\phi\).

\[
\begin{array}{c}
S \otimes TM \\
\xrightarrow{\text{op}' \otimes \text{id}} \\
M \otimes TM
\end{array}
\xrightarrow{\text{lift}^T_S \otimes \text{id}}
\begin{array}{c}
TS \otimes TM \\
\xrightarrow{T(\text{op}) \otimes \text{id}} \\
T(M \otimes M)
\end{array}
\xrightarrow{T(m)}
\begin{array}{c}
TM
\end{array}
\]

\[
\phi
\]

Theorem 4.17 (Algebraic/Codensity Coincidence). When \((T, \text{lift}^T)\) is a functorial monoid transformer (on a monoidal category with exponentials), and \(\text{op} : S \otimes M \to M\) is an algebraic operation for \(\hat{M}\), the codensity lifting (Theorem 4.15) and the algebraic lifting (Theorem 3.4) of \(\text{op}\) along \(\text{lift}^T_{\hat{M}}\) coincide.
Proof Since \( \mathsf{op} \) is an algebraic operation for \( \hat{M} = (M, e, m) \), by Proposition 3.3 exists unique \( \mathsf{op}' : S \to M \) such that \( \mathsf{op} = m \circ (\mathsf{op}' \otimes \text{id}) \). Similar characterizations hold for the following algebraic operations:

- \( \mathsf{op}^T : S \otimes TM \longrightarrow TM \) is the algebraic lifting of \( \mathsf{op} \) along \( \text{lift}^T_M \), therefore is algebraic for the monoid \( TM \) and corresponds to \( \text{lift}^T_M \circ \mathsf{op}' \).

- \( \mathsf{op}^K : S \otimes M^M \longrightarrow M^M \) (see Proposition 4.14) is the algebraic lifting of \( \mathsf{op} \) along \( \text{lift}^K_M \), therefore is algebraic for the monoid \( KM \) and corresponds to \( \text{lift}^K_M \circ \mathsf{op}' = \Lambda(\mathsf{op}) \).

- \( \mathsf{op}^{K,T} : S \otimes T(M^M) \longrightarrow T(M^M) \) given by the algebraic lifting of \( \mathsf{op}^K \) along \( \text{lift}^T_{KM} \) is algebraic for the monoid \( T(KM) \) and corresponds to \( \text{lift}^T_{KM} \circ \Lambda(\mathsf{op}) \).

\[ T(\text{lift}^K_M) \circ \text{lift}^T_M = \text{lift}^T_{KM} \circ \text{lift}^K_M \quad (\text{by naturality of } \text{lift}^T) \]

\[ \text{op}^{K,T} \] is the algebraic lifting of \( \text{op}^K \) along \( \text{lift}^T_{KM} \) and the following diagram commutes (the bottom path from \( S \otimes TM \) to \( TM \) is the codensity lifting of \( \mathsf{op} \))

\[
\begin{array}{ccc}
S \otimes TM & \xrightarrow{\text{id} \otimes \text{lift}^K_M} & S \otimes T(M^M) & \xrightarrow{\text{op}^{K,T}} & T(M^M) & \xrightarrow{T(\text{down}^M)} & TM \\
& & \downarrow & & \downarrow & & \\
& & S \otimes T(M^M) & \xrightarrow{\text{op}^{K,T}} & T(M^M) & \xrightarrow{T(\text{down}^M)} & TM \\
& & \downarrow & & \downarrow & & \\
& & S \otimes TM & \xrightarrow{\text{id} \otimes \text{lift}^K_M} & S \otimes T(M^M) & \xrightarrow{\text{op}^{K,T}} & T(M^M) & \xrightarrow{T(\text{down}^M)} & TM \\
\end{array}
\]

\[ \square \]

Theorem 4.18 (Codensity/Monoidal Coincidence). When \( (T, \text{lift}^T) \) is a monoidal monoid transformer (on a monoidal category with exponentials), and \( \mathsf{op} : S \otimes M \longrightarrow M \) is a first-order operation for \( \hat{M} \), the codensity lifting (Theorem 4.15) and the monoidal lifting (Theorem 4.10) of \( \mathsf{op} \) along \( \text{lift}^T_M \) coincide.

Proof The codensity lifting of \( \mathsf{op} \) is given by

\[ S \otimes TM \xrightarrow{\text{id} \otimes \text{lift}^K_M} S \otimes T(M^M) \xrightarrow{\text{op}^{K,T}} T(M^M) \xrightarrow{T(\text{down}^M)} TM \]

where \( \text{op}^{K,T} \) is the algebraic lifting of the algebraic operation \( \text{op}^K \) along \( \text{lift}^T_{KM} \) (see proof of Theorem 4.15), or equivalently (by Theorem 4.16) \( \text{op}^{K,T} \) is the monoidal lifting of \( \text{op}^K \) along \( \text{lift}^T_{KM} \).

Consider the following diagram, where the top path from \( S \otimes TM \) to \( TM \) is
the monoidal lifting of $\text{op}$, and the bottom path is the codensity lifting of $\text{op}$

\[
\begin{array}{c}
S \otimes TM \xrightarrow{\text{lift}^T_S \otimes \text{id}} TS \otimes TM \xrightarrow{\phi} T(S \otimes M) \xrightarrow{T(\text{op})} TM \\
\downarrow \text{id} \otimes T((\text{lift}^K_M) \quad \downarrow \text{id} \otimes T((\text{lift}^K_M) (1) \quad \downarrow \text{id} \otimes T((\text{lift}^K_M) (2) \quad \downarrow \text{T(down}_M) \\
S \otimes T(M^M) \xrightarrow{\text{lift}^T_S \otimes \text{id}} TS \otimes T(M^M) \xrightarrow{\phi} T(S \otimes M^M) \xrightarrow{T(\text{op}^K)} T(M^M)
\end{array}
\]

The diagram commutes for the following reasons:

1. because $\phi$ is a natural transformation.
2. by item (b) of Theorem 4.14 (and functoriality of $T$).

\[
\square
\]

5. Conclusion

Several category-theoretic notions, such as monads and adjunctions, can be recast in the setting of a 2-category (see [KS74]). In the case of monads, one could restrict to 2-categories with one object. A 2-category $\mathcal{C}$ with one object correspond to a strict monoidal category $\mathcal{E}$, and the correspondence induces a bijection between monads in $\mathcal{C}$ and monoids in $\mathcal{E}$. Finally, it is quite natural to drop the strictness assumption on $\mathcal{E}$, which amounts to replace 2-categories with bicategories [Bén67]. Therefore, the move from monads to monoids is a natural generalization. What was not straightforward (at least to us), and is the main novelty w.r.t. [Jas09], is the possibility of addressing the lifting problem (for monad transformers) at this level of generality.

The main results in the companion paper [Jas09] are related to the algebraic and codensity lifting (Theorems 3.4 and 4.15). Theorem 4.15 is not applicable to the monoidal category $\mathcal{E}_F$ of endofunctors expressible in $F_\omega$ (see Example 2.16), because it does not have exponentials. However, one can replace $\mathcal{E}_F$ with the monoidal category $\text{Endo}(\mathcal{P}_F)$ of Example 2.17, which provides also a fix to some false claims. [Jas09] uses expressible monad transformers (a proper subset of the monoid transformers on $\mathcal{E}_F$), which are amenable to implementation in a programming language.

The definition of functorial term of arity $S$ for an equational system on $\mathcal{C}$ (see [FH09]) is closely related to the definition of algebraic operation of signature $S$ for a monoid in the monoidal category of endofunctors on $\mathcal{C}$ (this is further evidence that the terminology “algebraic operation” is appropriate). In fact, if the category of algebras for an (iterated) equational system is equivalent to the category $\mathcal{C}_M^M$ of Eilenberg-Moore algebras for the monad $M$, then there is a bijection between algebraic operations $\text{op}$ of signature $S$ for $M$, i.e. natural trans-
formations $\text{op} : S \to M$, and functorial terms $T$ of arity $S$, i.e. functors such $C^M \to S\text{-Alg}$ that

$$C \xrightarrow{U} \downarrow \xrightarrow{\text{op}} \downarrow \xrightarrow{\text{op}_A} \downarrow,$$

given by

$$T(MA \xrightarrow{\alpha} A) = SA \xrightarrow{\text{op}_A} MA \xrightarrow{\alpha} A$$

$$\text{op}_A = SA \xrightarrow{\text{op}} S(MA) \xrightarrow{T(\mu_A)} MA.$$

This correspondence between functorial terms and algebraic operations suggests a reinterpretation (and generalization) of the notions introduced in [FH09] as described in the following table.

<table>
<thead>
<tr>
<th>Equational Systems [FH09]</th>
<th>Monoidal Category $E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterated equational system (IES)</td>
<td>monoid $M \in \text{Mon}(E)$</td>
</tr>
<tr>
<td>functorial signature $F$ (IES with $n = 0$)</td>
<td>object $F \in E$</td>
</tr>
<tr>
<td>category $F\text{-Alg}$ of $F$-algebras</td>
<td>free monoid $F^*$ over $F$</td>
</tr>
<tr>
<td>functorial term $T$ of arity $D$</td>
<td>map $\text{op} : D \to U(M)$</td>
</tr>
<tr>
<td>adding an equation to an IES</td>
<td>taking a quotient of $M$</td>
</tr>
<tr>
<td>$\text{IES} \vdash T_1 = T_2 : D$</td>
<td>$D \xrightarrow{\text{op}_1} \hat{M} \xrightarrow{\text{op}_2} \hat{N}$</td>
</tr>
</tbody>
</table>

A topic of future work is to investigate the use of free constructions for equational systems in defining strong monad transformers that add to a pre-existing monad new operations satisfying certain equations (see Example 4.8).

Another interesting line of research, already mentioned in the Introduction, is the use of monoid transformers for an incremental approach for arrows [Hug00] or other generalizations of monads proposed in the literature.

References


[Jas09] Mauro Jaskelioff. Modular monad transformers. In Castagna [Cas09], pages 64–79.


