Theoretical Computer Science 411 (2010) 4441–4466

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Monad transformers as monoid transformers

Mauro Jaskelioff [a,*], Eugenio Moggi [b]

[a] CIFASIS/Universidad Nacional de Rosario, Argentina
[b] DISI, Università di Genova, Italy

ARTICLE INFO

Keywords:
Monad
Monoid
Monoidal category

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 opens the way for lifting of operations with applicability beyond monads.

1. Introduction

Since monads have been proposed to model computational effects [31,32], they have proven to be extremely useful also to structure functional programs [42,41,18]. 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 [25] or compilers [24], consists of three steps [33,7]:

- 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*, which provides basic building blocks and *constructs* to build more complex blocks. Roughly speaking, one can identify two modular approaches

- the *incremental approach*, taken in [25,33,7], uses unary constructs, called monad transformers, which build complex monads by adding one computational feature to a pre-existing monad;

* Corresponding author. Tel.: +54 341 4802649.
E-mail addresses: jaskelioff@cifasis-conicet.gov.ar (M. Jaskelioff), moggi@disi.unige.it (E. Moggi).

0304-3975/$ – see front matter
doi:10.1016/j.tcs.2010.09.011

Lifting Theorems and their applicability
Assumptions on operation op and transformer T for lifting op through T

op	T	**Lifting theorem**
algebraic	basic	Theorem 3.4 (applies more generally to monoid maps)
first-order	functorial	Theorem 5.5 for monoidal category with exponentials
first-order	monoidal	Theorem 5.2 (applies to a more general form of op)

Fig. 1. Applicability of lifting theorems.

- the *compositional approach*, taken in [27,15], uses binary constructs, called monad combinations[1], for combining two pre-existing monads.

Both approaches fall short in dealing with operations associated to monads. This problem was identified in [25], which proposed a non-modular workaround, namely to *lift* in an ad hoc manner an operation through a monad transformer. Therefore, the number of liftings 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 [35] are easy to lift, but the monadic approach becomes of limited applicability if all operations have to be algebraic.

The compositional approach fits with the algebraic view of computational effects advocated in [35], and the combinations proposed in [15] give natural ways to *combine* monads induced by algebraic theories and to *lift* algebraic operations. However, some computational monads are not induced by algebraic theories, and some operations on computations are not algebraic.

The incremental approach is popular among functional programmers, because monad transformers are easy to implement. However, there has been limited progress in addressing the lifting problem, until a new insight was brought by [16,17]. Jaskelioff gives a *uniform way* of lifting operations in a certain class (which includes all the operations described in [25]) through any *functorial* monad transformer. This lifting has been implemented in [16] and studied in the setting of system $F\omega$ [17]. On algebraic operations it 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.

Contributions. Our main contribution is to develop a **theory of monoid transformers and lifting of operations in a categorical setting**, that generalizes, clarifies, and extends the current theory of monad transformers [25,33,7,17]. Category theory is known for its ability to abstract and generalize. We make good use of it, by developing a theory of lifting for monoid transformers, where monoids are taken in an unspecified monoidal category.

By a suitable choice of monoidal category, the theory specializes to monads, strong monads, finitary monads aka algebraic theories, and monads realizable in a typed or untyped calculus (such as system $F\omega$ or partial combinatory logic). Also other structures generalizing strong monads (such as arrows [14] and Freyd's categories [39]) *are* monoids in suitable monoidal categories [13,2]. Therefore, the theory may have a wider applicability.

Summary. Section 2 introduces *monoidal categories* (an internal language for monoidal categories) 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* and gives examples of strong monad transformers clarifying where they fit in the taxonomy. Section 5 provides more lifting results for monoid transformers (Theorems 5.5 and 5.2). Section 6 concludes with some considerations on related and future work. Fig. 1 says when the lifting theorems are applicable, while Fig. 2 summarizes the examples given in the paper of operations op associated to monads and monad transformers T. To assess the usefulness of the lifting theorems, use Fig. 1 to identify for which pairs (op, T) from Fig. 2 "op lifts through T". For instance, "callcc lifts through any T", because callcc is algebraic (Fig. 2).

Note for readers. We assume a modest knowledge of category theory. The notions relevant to the paper, but outside the scope of an introductory text book, are recalled in Section 2. Further information can be found in more advanced text books such as [28,4,8,5]. Each section includes several examples, some are not self-contained, but they are not needed to understand the main results. A reader may skip the examples at first, to get more directly to the lifting theorems, and then use Fig. 2 to select the examples of interest.

2. Monoidal categories

It is well known [28] that monads on a category $\mathcal{C}$ *correspond* to monoids in the (strict) monoidal category $\mathsf{Endo}(\mathcal{C})$ 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 $\mathsf{Endo}(\mathcal{C})$

[1] In the context of [15] it is more appropriate to call them theory combinations.

Taxonomy of operations op **associated to a monad** M

op algebraic $\Longrightarrow$ op first-order (see Definition 3.1)

Operation $\mathsf{op}_X : A(MX) \longrightarrow MX$ for M of arity A	**type**
$MX = R^{R^X}$ continuations (Example 3.8)	
$\mathsf{abort}_X : R \longrightarrow MX$	algebraic
$\mathsf{callcc}_X : (MX)^{(R^{MX})} \longrightarrow MX$	algebraic
$MX = X^S$ environments (Example 3.9)	
$\mathsf{read}_X : (MX)^S \longrightarrow MX$	algebraic
$\mathsf{local}_X : S^S \times MX \longrightarrow MX$	first-order
$MX = (X \times S)^S$ side-effects (Example 3.10)	
$\mathsf{read}_X : (MX)^S \longrightarrow MX$	algebraic
$\mathsf{write}_X : S \times MX \longrightarrow MX$	algebraic
$MX = X \times W$ complexity (Example 3.11)	
$\mathsf{add}_X : MX \times W \longrightarrow MX$	algebraic
$\underline{\mathsf{collect}}_X : MX \longrightarrow M(X \times W)$	none
$MX = X + E$ exceptions (Example 3.12)	
$\mathsf{throw}_X : E \longrightarrow MX$	algebraic
$\mathsf{handle}_X : MX \times (MX)^E \longrightarrow MX$	first-order

Taxonomy of monad transformers T

T monoidal $\Longrightarrow$ T functorial $\Longrightarrow$ T covariant $\Longrightarrow$ T basic (see Definition 4.1)

Transformer TMX	**type**
MX^S environments (Example 4.5)	monoidal
$M(X \times S)^S$ side-effects (Example 4.6)	monoidal
$M(X \times W)$ complexity (Example 4.7)	monoidal
$\mu X'.M(X + SX')$ S-steps[a] (Example 4.8)	functorial
$\mu X'.M(1 + X \times X')$ list (Example 4.9)	covariant
$MR^{(MR^X)}$ continuations (Example 4.10)	basic

Monoidal categories $\hat{\mathcal{E}}$ **with additional properties**

Monoidal category	**properties**
$\mathcal{C}$ with finite products (Example 2.14)	symmetric
profunctors (Example 2.16)	none
endofunctors (Example 2.16)	strict
strong endofunctors (Example 2.17)	strict
finitary endofunctors (Example 2.18)	strict, exponentials
expressible endofunctors in $F\omega$ (Example 2.19)	strict
realizable endofunctors in pCA (Example 2.20)	strict, exponentials
realizable endofunctors in $F\omega$ (Example 2.21)	strict, exponentials

Fig. 2. Overview of examples.

[a] By a suitable choice of the endofunctor S the transformer T becomes $TMX = M(X + E)$ exceptions, $TMX = \mu X'.M(X + X')$ resumptions, and so on.

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 (on a category $\mathcal{C}$);
- generality: the theory has several instantiations, including different *flavours* of monads, by choosing a different monoidal category $\hat{\mathcal{E}}$.

Readers already familiar with monoidal categories can browse through most of this section, and look only at some examples in Section 2.3.

Definition 2.1 (*Monoidal Category [28]*)**.** A *monoidal category* $\hat{\mathcal{E}}$ is a tuple $(\mathcal{E}, \otimes, \mathsf{I}, \alpha, \lambda, \rho)$, where

- $\mathcal{E}$ is a category, $\otimes : \mathcal{E} \times \mathcal{E} \longrightarrow \mathcal{E}$ is a bifunctor, $\mathsf{I} \in \mathcal{E}$ is an object
- $\alpha_{a,b,c} : a \otimes (b \otimes c) \longrightarrow (a \otimes b) \otimes c$, $\lambda_a : \mathsf{I} \otimes a \longrightarrow a$, $\rho_a : a \otimes \mathsf{I} \longrightarrow a$ are natural isomorphisms such that the diagrams (2.1) and (2.2) commute

$$\begin{tikzcd}
a \otimes (b \otimes (c \otimes d)) \arrow[r, "\alpha"] \arrow[d, "\mathsf{id} \otimes \alpha"'] & (a \otimes b) \otimes (c \otimes d) \arrow[r, "\alpha"] & ((a \otimes b) \otimes c) \otimes d \\
a \otimes ((b \otimes c) \otimes d) \arrow[rr, "\alpha"'] & & (a \otimes (b \otimes c)) \otimes d \arrow[u, "\alpha \otimes \mathsf{id}"']
\end{tikzcd} \tag{2.1}$$

$$\begin{tikzcd}
a \otimes (\mathsf{I} \otimes b) \arrow[r, "\alpha"] \arrow[d, "\mathsf{id} \otimes \lambda"'] & (a \otimes \mathsf{I}) \otimes b \arrow[d, "\rho \otimes \mathsf{id}"] \\
a \otimes b \arrow[r, equal] & a \otimes b
\end{tikzcd} \tag{2.2}$$

When the natural isomorphisms α, λ and ρ are identities, the diagrams necessarily commute, and the monoidal category is called *strict*.

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

objects are *monoids* $\hat{M} = (M, e, m)$, i.e. $\mathsf{I} \xrightarrow{\ e\ } M \xleftarrow{\ m\ } M \otimes M$ in $\mathcal{E}$ such that

$$\begin{tikzcd}
(M \otimes M) \otimes M \arrow[rr, "m \otimes \mathsf{id}"] & & M \otimes M \arrow[d, "m"] \\
M \otimes (M \otimes M) \arrow[u, "\alpha"] \arrow[r, "\mathsf{id} \otimes m"'] & M \otimes M \arrow[r, "m"'] & M
\end{tikzcd} \tag{2.3}$$

$$\begin{tikzcd}
\mathsf{I} \otimes M \arrow[r, "\lambda"] \arrow[dr, "e \oplus \mathsf{id}"'] & M & M \otimes \mathsf{I} \arrow[l, "\rho"'] \arrow[dl, "\mathsf{id} \oplus e"] \\
& M \otimes M \arrow[u, "m"] &
\end{tikzcd} \tag{2.4}$$

arrows from $\hat{M}_1$ to $\hat{M}_2$ are arrows $M_1 \xrightarrow{\ f\ } M_2$ in $\mathcal{E}$ such that

$$\begin{tikzcd}
\mathsf{I} \arrow[r, "e_1"] \arrow[d, equal] & M_1 \arrow[d, "f"] & M_1 \otimes M_1 \arrow[l, "m_1"'] \arrow[d, "f \otimes f"] \\
\mathsf{I} \arrow[r, "e_2"'] & M_2 & M_2 \otimes M_2 \arrow[l, "m_2"]
\end{tikzcd} \tag{2.5}$$

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

The forgetful functor $U : \mathsf{Mon}(\hat{\mathcal{E}}) \longrightarrow \mathcal{E}$ 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 $\hat{\mathcal{E}}$ is an object b^a together with an arrow $ev : b^a \otimes a \longrightarrow b$ satisfying the universal property

$$\forall x \in \mathcal{E}.\, \forall f : x \otimes a \longrightarrow b.\, \exists!\, \Lambda f : x \dashrightarrow b^a \text{ such that }
\begin{tikzcd}
b^a \otimes a \arrow[r, "ev"] & b \\
x \otimes a \arrow[u, dashed, "\Lambda f \otimes \mathsf{id}"] \arrow[ur, "f"'] &
\end{tikzcd} \tag{2.6}$$

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 $(T, \phi_{\mathsf{I}}, \phi)$, where

- $T : \mathcal{E} \longrightarrow \mathcal{E}'$ is a functor
- $\phi_{\mathsf{I}} : \mathsf{I}' \longrightarrow T\mathsf{I}$ is an arrow, and $\phi_{a,b} : Ta \otimes' Tb \longrightarrow T(a \otimes b)$ is a natural transformation such that

$$\begin{tikzcd}
Ta \otimes' (Tb \otimes' Tc) \arrow[r, "\mathsf{id} \otimes' \phi"] \arrow[d, "\alpha'"] & Ta \otimes' T(b \otimes c) \arrow[r, "\phi"] & T(a \otimes (b \otimes c)) \arrow[d, "T(\alpha)"] \\
(Ta \otimes' Tb) \otimes' Tc \arrow[r, "\phi \otimes' \mathsf{id}"'] & T(a \otimes b) \otimes' Tc \arrow[r, "\phi"'] & T((a \otimes b) \otimes c)
\end{tikzcd} \tag{2.7}$$

$$\begin{tikzcd}
\mathsf{I}' \otimes' Ta \arrow[rr, "\lambda'"] \arrow[d, "\phi_{\mathsf{I}} \otimes' \mathsf{id}"] & & Ta & & Ta \otimes' \mathsf{I}' \arrow[ll, "\rho'"'] \arrow[d, "\mathsf{id} \otimes' \phi_{\mathsf{I}}"] \\
T\mathsf{I} \otimes' Ta \arrow[r, "\phi"'] & T(\mathsf{I} \otimes a) \arrow[ur, "T\lambda"] & & T(a \otimes \mathsf{I}) \arrow[ul, "T\rho"'] & Ta \otimes' T\mathsf{I} \arrow[l, "\phi"]
\end{tikzcd} \tag{2.8}$$

When the arrows ϕ_{I} and $\phi_{a,b}$ are identities, the monoidal functor is called *strict*, and the commuting diagrams amount to say $\mathsf{I}' = T\mathsf{I}$, $Ta \otimes' Tb = T(a \otimes b)$, $\alpha' = T(\alpha)$, $\lambda' = T(\lambda)$ and $\rho' = T(\rho)$.

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

$$\begin{tikzcd}
\mathsf{I}' \arrow[r, equal] \arrow[d, "\phi_{\mathsf{I}}"] & \mathsf{I}' \arrow[d, "\phi'_{\mathsf{I}}"] & Ta \otimes' Tb \arrow[r, "\tau_a \otimes' \tau_b"] \arrow[d, "\phi"] & T'a \otimes' T'b \arrow[d, "\phi'"] \\
T\mathsf{I} \arrow[r, "\tau_{\mathsf{I}}"'] & T'\mathsf{I} & T(a \otimes b) \arrow[r, "\tau_{a \otimes b}"'] & T'(a \otimes b)
\end{tikzcd} \tag{2.9}$$

Theorem 2.6 (*Extension*). *A monoidal functor* $\hat{T} : \hat{\mathcal{E}} \longrightarrow \hat{\mathcal{E}}'$ *induces a functor* $T : \mathsf{Mon}(\hat{\mathcal{E}}) \longrightarrow \mathsf{Mon}(\hat{\mathcal{E}}')$, *and similarly a monoidal natural transformation* $\tau : \hat{T} \xrightarrow{\bullet} \hat{T}'$ *induces a natural transformation* $\tau : T \xrightarrow{\bullet} T'$ *such that*

$$T\hat{M} = \mathsf{I}' \xrightarrow{\phi_{\mathsf{I}}} \cdot \xrightarrow{Te} TM \xleftarrow{Tm} \cdot \xleftarrow{\phi} TM \otimes' TM \tag{2.10}$$

$$\begin{tikzcd}
\mathsf{Mon}(\hat{\mathcal{E}}) \arrow[r, bend left, "T"] \arrow[r, bend right, "T'"'] \arrow[r, phantom, "\Downarrow \tau"] & \mathsf{Mon}(\hat{\mathcal{E}}') \arrow[r, "U"] & \mathcal{E}' = \mathsf{Mon}(\hat{\mathcal{E}}) \arrow[r, "U"] & \mathcal{E} \arrow[r, bend left, "T"] \arrow[r, bend right, "T'"'] \arrow[r, phantom, "\Downarrow \tau"] & \mathcal{E}'
\end{tikzcd} \tag{2.11}$$

Proof. We prove that $(M', e', m') \hat{=} T\hat{M}$ is a monoid in $\hat{\mathcal{E}}'$, namely the analog of diagrams (2.3) and (2.4) in Definition 2.2 commute.

$$\begin{tikzcd}
(M' \otimes' M') \otimes' M' \arrow[rr, "\phi \otimes' \mathsf{id}"] & & \cdot \arrow[r, "Tm \otimes' T\mathsf{id}"] \arrow[dd, "\phi"] & M' \otimes' M' \arrow[dd, "\phi"] \\
M' \otimes' (M' \otimes' M') \arrow[u, "\alpha'"] \arrow[d, "\mathsf{id} \otimes' \phi"] \arrow[rr, phantom, "(1)"] & & \arrow[r, phantom, "(2)"] & {} \\
\cdot \arrow[r, "\phi"] \arrow[d, "T\mathsf{id} \otimes' Tm"] \arrow[r, phantom, shift right=4ex, "(2)"] & \cdot \arrow[r, "T\alpha"] \arrow[d, "T(\mathsf{id} \otimes m)"] \arrow[rr, phantom, shift right=4ex, "(3)"] & \cdot \arrow[r, "T(m \otimes \mathsf{id})"] & \cdot \arrow[d, "Tm"] \\
M' \otimes' M' \arrow[r, "\phi"'] & \cdot \arrow[rr, "Tm"'] & & M'
\end{tikzcd}$$

1. by diagram (2.7) in Definition 2.4
2. by naturality of ϕ
3. by functoriality of T and diagram (2.3) in Definition 2.2.

$$\begin{tikzcd}
\mathsf{I}' \otimes' M' \arrow[r, "\phi_{\mathsf{I}} \otimes' \mathsf{id}"] \arrow[dd, "\lambda'"] \arrow[ddr, phantom, "(1)"] & T\mathsf{I} \otimes' M' \arrow[r, "Te \otimes' T\mathsf{id}"] \arrow[d, "\phi"] \arrow[dr, phantom, "(2)"] & M' \otimes' M' \arrow[d, "\phi"] \arrow[ddr, phantom, "(4)"] & M' \otimes' \mathsf{I}' \arrow[l, "\mathsf{id} \otimes' e'"'] \arrow[dd, "\rho'"] \\
 & \cdot \arrow[r, "T(e \otimes \mathsf{id})"] \arrow[d, "T\lambda"] \arrow[dr, phantom, "(3)"] & \cdot \arrow[d, "Tm"] & \\
M' \arrow[r, equal] & M' \arrow[r, equal] & M' \arrow[r, equal] & M'
\end{tikzcd}$$

1. by diagram (2.8) in Definition 2.4
2. by naturality of ϕ
3. by functoriality of T and diagram (2.4) in Definition 2.2
4. same justifications as in the items 1–3 above, but with λ replaced by ρ and the definition of e' expanded.

We prove that $Tf : T\hat{M}_1 \longrightarrow T\hat{M}_2$ in $\mathsf{Mon}(\hat{\mathcal{E}}')$, namely the analog of diagram (2.5) in Definition 2.2 commutes, when $f : \hat{M}_1 \longrightarrow \hat{M}_2$ in $\mathsf{Mon}(\hat{\mathcal{E}})$.

$$\begin{tikzcd}
\mathsf{I}' \arrow[r, "\phi_{\mathsf{I}}"] \arrow[d, equal] & \cdot \arrow[r, "Te_1"] \arrow[d, equal] \arrow[dr, phantom, "(1)"] & TM_1 \arrow[d, "Tf"] \arrow[dr, phantom, "(1)"] & \cdot \arrow[l, "Tm_1"'] \arrow[d, "T(f \otimes f)"] \arrow[dr, phantom, "(2)"] & TM_1 \otimes' TM_1 \arrow[l, "\phi"'] \arrow[d, "Tf \otimes Tf"] \\
\mathsf{I}' \arrow[r, "\phi_{\mathsf{I}}"'] & \cdot \arrow[r, "Te_2"'] & TM_2 & \cdot \arrow[l, "Tm_2"] & TM_2 \otimes' TM_2 \arrow[l, "\phi"]
\end{tikzcd}$$

1. by functoriality of T and diagram (2.5) in Definition 2.2
2. by naturality of ϕ.

We prove that $\tau_M : T\hat{M} \longrightarrow T'\hat{M}$ in $\mathsf{Mon}(\hat{\mathcal{E}}')$, namely the analog of diagram (2.5) in Definition 2.2 commutes, for any monoid $\hat{M}$ in $\mathsf{Mon}(\hat{\mathcal{E}})$.

$$\begin{tikzcd}
\mathsf{I}' \arrow[r, "\phi_{\mathsf{I}}"] \arrow[d, equal] \arrow[dr, phantom, "(1)"] & \cdot \arrow[r, "Te"] \arrow[d, "\tau_{\mathsf{I}}"] \arrow[dr, phantom, "(2)"] & TM \arrow[d, "\tau_M"] \arrow[dr, phantom, "(2)"] & \cdot \arrow[l, "Tm"'] \arrow[d, "\tau_{M \otimes M}"] \arrow[dr, phantom, "(1)"] & TM \otimes' TM \arrow[l, "\phi"'] \arrow[d, "\tau_M \otimes \tau_M"] \\
\mathsf{I}' \arrow[r, "\phi'_{\mathsf{I}}"'] & \cdot \arrow[r, "T'e"'] & T'M & \cdot \arrow[l, "T'm"] & T'M \otimes' T'M \arrow[l, "\phi'"]
\end{tikzcd}$$

1. by diagram (2.9) in Definition 2.5
2. by naturality of τ. □

2.1. Languages for monoidal categories

It is well known (see [40,22,23]) that the simply typed λ-calculus can be interpreted in any cartesian closed category $\mathcal{C}$: types τ and type assignments Γ are interpreted by objects, and well-formed terms $\Gamma \vdash t : \tau$ by arrows (from the interpretation of Γ to the interpretation of τ). Conversely by extending the simply typed λ-calculus with types and operations representing objects and arrows of $\mathcal{C}$, one can express diagrams in $\mathcal{C}$ as (sets of) well-formed equations $\Gamma \vdash t_1 = t_2 : \tau$, and by devising a suitable notion of theory, one can establish an equivalence between a category of theories and a category of models.

In this section we introduce typed calculi for monoidal categories (with exponentials). Our aims are pragmatic, i.e. to use these calculi to express definitions, statements and proofs involving monoidal categories. In fact, expressing diagrams with equations may sometimes improve readability and simplify proofs.

Figs. 3 and 4 define the language for monoidal categories with exponentials. The language is inspired by the natural deduction system for intuitionistic non-commutative linear logic described in [38].

We say that a typing $\Gamma \vdash t : \tau$ is well-formed, when it is derivable from the rules in Fig. 4, and an equation $\Gamma \vdash t_1 = t_2 : \tau$ is well-formed , when the typings $\Gamma \vdash t_1 : \tau$ and $\Gamma \vdash t_2 : \tau$ are well-formed. An interpretation $[\![-]\!]$ of the language in a monoidal category $\hat{\mathcal{E}}$ (with additional structure) is defined *by induction*

Variables	$x \in \mathsf{X}$
Terms	$t \in \mathsf{E} ::= \quad x \mid \mathsf{op}(t) \mid (t_1, t_2) \mid \mathsf{let}\,(x_1, x_2) = t_1 \,\mathsf{in}\, t_2 \mid$ $* \mid \mathsf{let} * = t_1 \,\mathsf{in}\, t_2 \mid \lambda x.t \mid t\ t$
Base Types	$a \in \mathsf{B}$
Types	$\tau \in \mathsf{T} ::= a \mid \tau_1 \otimes \tau_2 \mid \mathsf{I} \mid \tau_2^{\tau_1}$

Assignments $\Gamma \in (\mathsf{X} \times \mathsf{T})^*$ such that each $x \in \mathsf{X}$ occurs at most once in Γ

We write $x : \tau$ for the assignment consisting of the pair (x, τ), and Γ_1, Γ_2 for the concatenation of two assignments. The concatenation Γ_1, Γ_2 of two assignments fails to be an assignment, when a variable x occurs in both Γ_1 and Γ_2.

A term t is identified with its equivalence class modulo α-conversion. We use the derived notation $\mathsf{let}\, p = t_1 \,\mathsf{in}\, t_2$, where $p ::= x \mid * \mid (p_1, p_2)$ is a *linear* pattern.

Fig. 3. Syntax.

$$\text{var}\ \frac{}{x : \tau \vdash x : \tau} \qquad \text{map}\ \frac{\Gamma \vdash t : \tau_1}{\Gamma \vdash \mathsf{op}(t) : \tau_2}\ \mathsf{op} : \tau_1 \to \tau_2$$

$$\otimes.\mathrm{I}\ \frac{\Gamma_1 \vdash t_1 : \tau_1 \qquad \Gamma_2 \vdash t_2 : \tau_2}{\Gamma_1, \Gamma_2 \vdash (t_1, t_2) : \tau_1 \otimes \tau_2} \qquad \otimes.\mathrm{E}\ \frac{\Gamma_2 \vdash t_1 : \tau_1 \otimes \tau_2 \qquad \Gamma_1, x_1 : \tau_1, x_2 : \tau_2, \Gamma_3 \vdash t_2 : \tau}{\Gamma_1, \Gamma_2, \Gamma_3 \vdash \mathsf{let}\,(x_1, x_2) = t_1 \,\mathsf{in}\, t_2 : \tau}$$

$$\mathsf{I}.\mathrm{I}\ \frac{}{\vdash * : \mathsf{I}} \qquad \mathsf{I}.\mathrm{E}\ \frac{\Gamma_2 \vdash t_1 : \mathsf{I} \qquad \Gamma_1, \Gamma_3 \vdash t_2 : \tau}{\Gamma_1, \Gamma_2, \Gamma_3 \vdash \mathsf{let} * = t_1 \,\mathsf{in}\, t_2 : \tau}$$

$$\to.\mathrm{I}\ \frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash \lambda x : \tau_1.t : \tau_2^{\tau_1}} \qquad \to.\mathrm{E}\ \frac{\Gamma_1 \vdash t_1 : \tau_2^{\tau_1} \qquad \Gamma_2 \vdash t_2 : \tau_2}{\Gamma_1, \Gamma_2 \vdash t_1\ t_2 : \tau_2}$$

The type system is for deriving typings of the form $\Gamma \vdash t : \tau$, with Γ an assignment. Therefore, each typing rule has an implicit side-condition requiring that the concatenation of assignments in the conclusion must be an assignment.

Fig. 4. Type system.

- $[\![\tau]\!]$ is an object of $\mathcal{E}$ defined by induction on the structure of the type τ;
- $[\![\Gamma]\!]$ is an object of $\mathcal{E}$ defined by induction on the length of the assignment Γ: the empty assignment is interpreted by I, and $[\![\Gamma, x : \tau]\!] \mathrel{\hat{=}} [\![\Gamma]\!] \otimes [\![\tau]\!]$;
- $[\![\Gamma \vdash t : \tau]\!]$ is an arrow of $\mathcal{E}$ from $[\![\Gamma]\!]$ to $[\![\tau]\!]$ defined by induction on the *unique* derivation of the well-formed typing $\Gamma \vdash t : \tau$, e.g.
  if $[\![\Gamma_i \vdash t_i : \tau_i]\!] = f_i : [\![\Gamma_i]\!] \longrightarrow [\![\tau_i]\!]$, then $[\![\Gamma_1, \Gamma_2 \vdash (t_1, t_2) : \tau_1 \otimes \tau_2]\!]$ is

  $$[\![\Gamma_1, \Gamma_2]\!] \xrightarrow{\sim} [\![\Gamma_1]\!] \otimes [\![\Gamma_2]\!] \xrightarrow{f_1 \otimes f_2} [\![\tau_1]\!] \otimes [\![\tau_2]\!]$$

  where $[\![\Gamma_1, \Gamma_2]\!] \xrightarrow{\sim} [\![\Gamma_1]\!] \otimes [\![\Gamma_2]\!]$ is the *unique* isomorphism given by the coherence result for monoidal categories (see [28]).

If $\Gamma \vdash t_1 = t_2 : \tau$ is a well-formed equation and $[\![-]\!]_I$ is an interpretation of the language, as outlined above, then we write $\Gamma \vdash_I t_1 = t_2 : \tau$, when the interpretations $[\![\Gamma \vdash t_i : \tau]\!]_I$ denote the same morphism.

Definition 2.7 (*Monoid*). We express as well-formed equations Definition 2.2 of monoid $\hat{M} = (M, e, m)$ and monoid morphism $f : \hat{M}_1 \longrightarrow \hat{M}_2$

- The diagrams (2.3) and (2.4) are equivalent to the equations

$$x : M \vdash x \cdot e = x : M \tag{2.12}$$

$$x : M \vdash e \cdot x = x : M \tag{2.13}$$

$$x_1, x_2, x_3 : M \vdash (x_1 \cdot x_2) \cdot x_3 = x_1 \cdot (x_2 \cdot x_3) : M \tag{2.14}$$

  where M is a base type, $\mathsf{op}_e : \mathsf{I} \to M$ and $\mathsf{op}_m : M \otimes M \to M$ are operations, and we write e for $\mathsf{op}_e(*)$ and $t_1 \cdot t_2$ for $\mathsf{op}_m(t_1, t_2)$.
- The diagram (2.5) is equivalent to the equations

$$\vdash f\ e_1 = e_2 : M_2 \tag{2.15}$$

$$x_1, x_2 : M_1 \vdash f(x_1 \cdot_1 x_2) = (f\ x_1) \cdot_2 (f\ x_2) : M_2 \tag{2.16}$$

  where M_i, e_i and $t_1 \cdot_i t_2$ are as above, and $f : M_1 \to M_2$ is an operation.

The reduction rules of Fig. 5 induce a reduction $t_1 \Longrightarrow t_2$ (on terms modulo α-conversion) with the following properties:

$$\mathsf{let}\,(x_1, x_2) = (t_1, t_2)\,\mathsf{in}\,t \xrightarrow{\beta.\otimes} t[x_1 : t_1, x_2 : t_2]$$
$$\mathsf{let} * = * \,\mathsf{in}\,t \xrightarrow{\beta.\mathsf{I}} t$$
$$(\lambda x : \tau_1.t_2)\ t_1 \xrightarrow{\beta.\to} t_2[x : t_1]$$

$t'[x : t]$ denotes substitution of x with t in t' modulo α-conversion, namely bound variables in t' are renamed to avoid clashes with the free variables in t. We denote with $\Longrightarrow$ the compatible closure of the reduction rules given above.

Fig. 5. Reduction.

- subject reduction, i.e. $\Gamma \vdash t_1 : \tau$ and $t_1 \Longrightarrow t_2$ imply $\Gamma \vdash t_2 : \tau$
- confluence, i.e. $t_1 \Longrightarrow^* t_2$ and $t_1 \Longrightarrow^* t_3$ imply $t_2 \Longrightarrow^* t_4$ and $t_3 \Longrightarrow^* t_4$ for some t_4
- strong normalization, i.e. $\Gamma \vdash t : \tau$ implies exists n such that $m \leq n$ whenever $t \Longrightarrow^m t'$
- soundness, i.e. $\Gamma \vdash t_1 : \tau$ and $t_1 \Longrightarrow t_2$ imply $\Gamma \vdash_I t_1 = t_2 : \tau$ for any I[2].

We write Eq_0 for the set of well-formed $\Gamma \vdash t_1 = t_2 : \tau$ such that $t_1 \Longrightarrow t_2$. Given a set Eq of well-formed equations, we write $\Gamma \vdash_{Eq} t_1 = t_2 : \tau$, when the well-formed equation $\Gamma \vdash t_1 = t_2 : \tau$ is in the congruence induced by $Eq \cup Eq_0$.

Notation 2.8. To prove $\Gamma \vdash_{Eq} t = t' : \tau$ we give a stack of rewriting steps $\begin{array}{ll} C[\,\underline{t_1}\,] & \text{by justification} \\ C[\,\overline{t_2}\,] & \end{array}$ (from t down to t'), where $C[-]$ is a context with one hole and *justification* explains why $t_1 = t_2$ (more precisely $\Gamma' \vdash t_1 = t_2 : \tau'$, with Γ' and τ' inferable from Γ, τ and $C[-]$). A justification could be

- reduction, when $t_1 \Longrightarrow^* t_0$ and $t_2 \Longrightarrow^* t_0$ for some term t_0, or
- $\Gamma' \vdash eq : \tau'$ in Eq, when $t_1 = t_2$ is a substitution instance of eq.

We suppress the underlining/overlining when the context is the hole. Proofs in this style can be found in Example 2.11. □

2.2. Examples of monoids

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

Example 2.9. The **initial monoid** $\hat{\mathsf{I}}$, is given by $\mathsf{I} \xrightarrow{\mathsf{id}} \mathsf{I} \xleftarrow{\lambda} \mathsf{I} \otimes \mathsf{I}$ and is an initial object in $\mathsf{Mon}(\hat{\mathcal{E}})$. □

Example 2.10. When $\mathcal{E}$ has J-limits, i.e. limits for diagrams of shape J, then $\mathsf{Mon}(\hat{\mathcal{E}})$ 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 the unique monoid structure $\hat{1}$ on 1 yields a terminal object in $\mathsf{Mon}(\hat{\mathcal{E}})$. □

Example 2.11. When the exponential a^a exists, the **monoid** $\mathsf{K}a$ **of endomorphisms** on a is given by $\mathsf{I} \xrightarrow{\mathsf{i}_a} a^a \xleftarrow{\mathsf{c}_a} a^a \otimes a^a$ where

$$\mathsf{i}_a : a^a \mathrel{\hat{=}} \lambda x : a.x \tag{2.17}$$
$$\mathsf{c}_a(g, f : a^a) : a^a \mathrel{\hat{=}} \lambda x : a.g\ (f\ x) \tag{2.18}$$

Moreover, if $\hat{M} = (M, e, m)$ is a monoid, then one has a monoid morphism $\mathsf{to}_{\hat{M}} : \hat{M} \longrightarrow \mathsf{K}M$ given by

$$\mathsf{to}_{\hat{M}}(x : M) : M^M \mathrel{\hat{=}} \lambda x' : M.x \cdot x' \tag{2.19}$$

We show that $\mathsf{K}a$ is a monoid, i.e. it satisfies the Eqs. (2.12), (2.13) and (2.14), Let Eq be the set containing only $(\eta. \to)$, i.e. the sound equation $x' : a^a \vdash (\lambda x : a.x'\ x) = x' : a^a$ (we drop the type a of bound variables)

- $x' : a^a \vdash_{Eq} \mathsf{c}_a(x', \mathsf{i}_a) = x' : a^a$
  $\mathsf{c}_a(x', \mathsf{i}_a)$ by definition
  $\lambda x.x'\ \underline{((\lambda x.x)\ x)}$ by reduction $(\beta. \to)$
  $\lambda x.x'\ \overline{x}$ by $(\eta. \to)$ in Eq
  x'
- $x' : a^a \vdash_{Eq} \mathsf{c}_a(\mathsf{i}_a, x') = x' : a^a$ the proof is similar to the one above.

[2] The reduction is *incomplete*, since there is a well-formed $\Gamma \vdash t_1 = t_2 : \tau$ that holds in any interpretation (e.g. $x : \mathsf{I} \vdash (\mathsf{let} * = x\,\mathsf{in}\,*) = x : \mathsf{I}$), but t_1 and t_2 have different normal forms.

- $x_1, x_2, x_3 : a^a \vdash_{Eq} \mathsf{c}_a(\mathsf{c}_a(x_1, x_2), x_3) = \mathsf{c}_a(x_1, \mathsf{c}_a(x_2, x_3)) : a^a$
  $\mathsf{c}_a(\mathsf{c}_a(x_1, x_2), x_3)$ by definition
  $\lambda x.\underline{(\lambda x.x_1\ (x_2\ x))\ (x_3\ x)}$ by reduction $(\beta. \to)$
  $\lambda x.x_1\ (\overline{(\lambda x.x_2\ (x_3\ x))\ x})$ by definition
  $\mathsf{c}_a(x_1, \mathsf{c}_a(x_2, x_3))$

We show that $\mathsf{to}_{\hat{M}}$ is a monoid map, i.e. it satisfies the Eqs. (2.15) and (2.16), when $\hat{M}$ is a monoid. Let Eq be the set of equations saying that $\hat{M}$ is a monoid (we drop the type M of bound variables)

- $\vdash_{Eq} \mathsf{to}_{\hat{M}}(e) = \mathsf{i}_M : M^M$
  $\mathsf{to}_{\hat{M}}(e)$ by definition
  $\lambda x.\underline{e \cdot x}$ by (2.13) in Eq
  $\lambda x.\overline{x}$ by definition
  i_M
- $x_1, x_2 : M \vdash_{Eq} \mathsf{to}_{\hat{M}}(x_1 \cdot x_2) = \mathsf{c}_M(\mathsf{to}_{\hat{M}}(x_1), \mathsf{to}_{\hat{M}}(x_2)) : M^M$
  $\mathsf{to}_{\hat{M}}(x_1 \cdot x_2)$ by definition
  $\lambda x_3.\underline{(x_1 \cdot x_2) \cdot x_3}$ by (2.14) in Eq
  $\lambda x_3.\overline{x_1 \cdot (x_2 \cdot x_3)}$ by reduction $(\beta. \to)$
  $\lambda x_3.\overline{(\lambda x.x_1 \cdot x)\ ((\lambda x.x_2 \cdot x)\ x_3)}$ by definition
  $\mathsf{c}_M(\mathsf{to}_{\hat{M}}(x_1), \mathsf{to}_{\hat{M}}(x_2))$ $\square$

Example 2.12. When the left-adjoint $(-)^*$ to $U : \mathsf{Mon}(\hat{\mathcal{E}}) \longrightarrow \mathcal{E}$ exists, it gives **free monoids**. There are several assumptions on $\hat{\mathcal{E}}$, which imply the existence of free monoids. For instance (see [19, Page 68–69]):

1. if $\hat{\mathcal{E}}$ has exponentials, $\mathcal{E}$ has binary coproducts, and for each $a \in \mathcal{E}$ the initial algebra for the endofunctor $\mathsf{I} + a \otimes -$ exists, then a^* exists and its carrier is given the carrier $\mu x.\mathsf{I} + a \otimes x$ of the initial algebra;
2. if $\mathcal{E}$ has binary coproducts, for each $a \in \mathcal{E}$ the endofunctor $- \otimes a$ preserves colimits, and for each $a \in \mathcal{E}$ the chain a_β defined by ordinal induction

$$a_0 \mathrel{\hat{=}} \mathsf{I} \quad a_{\beta+1} \mathrel{\hat{=}} \mathsf{I} + a \otimes a_\beta \qquad a_\lambda \mathrel{\hat{=}} \operatorname*{colim}_{\beta<\lambda} a_\beta \ (\lambda \text{ limit ordinal})$$

   converges at some β, i.e. $a_\beta = a_{\beta+1}$, then a^* exists and its carrier is a_β. $\square$

Example 2.13. Given a monoid $\hat{M} = (M, e, m)$ in $\hat{\mathcal{E}}$, and a monic $M' \overset{i}{\hookrightarrow} M$ in $\mathcal{E}$, such that for some (unique) maps e' and m'

$$\begin{tikzcd} \mathsf{I} \arrow[r, "e"] \arrow[dr, "e'"'] & M & M \otimes M \arrow[l, "m"'] \\ & M' \arrow[u, "i"] & M' \otimes M' \arrow[l, "m'"] \arrow[u, "i \otimes i"] \end{tikzcd}$$

then $\hat{M}' \mathrel{\hat{=}} (M', e', m')$ is a monoid, called the **sub-monoid** of $\hat{M}$ induced by the monic i, and $\hat{M}' \overset{i}{\hookrightarrow} \hat{M}$ is a monoid monomorphism. The general definition of *quotient* of a monoid $\hat{M}$ is more involved. We give concrete descriptions of sub-monads and quotient monads in **Set**, i.e. sub-monoids and quotient monoids in $\mathsf{Endo}(\mathbf{Set})$ of Example 2.16. Given a monad $\hat{M} = (M, \eta, -^*)$ on **Set** presented as a *Kleisli triple* (see [29,32]):

- 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 $\forall X, Y. \forall f : X \longrightarrow S_Y. \forall x \in S_X. g^* x \in S_Y$ where $g = X \xrightarrow{f} S_Y \hookrightarrow MY$.
- 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
  $\forall X, Y. \forall f : X \longrightarrow R_Y. \forall (x_1, x_2) \in R_X. (g_1^* x_1, g_2^* x_2) \in R_Y$ 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. $\square$

2.3. Examples of monoidal categories

We give several examples of monoidal categories, and when possible we say whether they have exponentials. The definition of monoidal category is *self-dual*, i.e. there is a bijection between monoidal structures on $\mathcal{E}$ and on $\mathcal{E}^{op}$. Therefore, each example has a dual.

- A category with finite products (Example 2.14), like **Set**, is the most obvious example of monoidal category.
- Example 2.15 defines several full sub-categories of a monoidal category.
- For monads, the category $\mathsf{Endo}(\mathcal{C})$ of endofunctors (Example 2.16) is paradigmatic, and the other examples we give are *variations* on this.
- For strong monads, the appropriate variation on $\mathsf{Endo}(\mathcal{C})$ is the category of strong endofunctors (Example 2.17),
- For algebraic theories [29] (and collection types [30]), an appropriate choice is the category of finitary endofunctors (Example 2.18),
- The category of endofunctors expressible in $F\omega$ (Example 2.19) establishes a formal link with [17], and is paradigmatic of *syntactic* examples based on typed calculi, but it does not have exponentials.
- Realizability [26,34] is a general technique to build models for rich type structures on top of computationally expressive (untyped) applicative structures, Examples 2.20 and 2.21 define realizable endofunctors on a category of partial equivalence relations on a partial combinatory algebra and a second-order combinatory algebra, respectively.

Example 2.14. A category $\mathcal{C}$ with **finite products** (e.g. the category **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 2.15. Given a monoidal category $\hat{\mathcal{E}}$ with J-colimits (similar results hold for J-limits), we write $\mathsf{Colim}_J(\hat{\mathcal{E}})$ for the full sub-category of $\mathcal{E}$ whose objects $a \in \mathcal{E}$ **preserve** J**-colimits**, i.e. the functor $a \otimes - : \mathcal{E} \longrightarrow \mathcal{E}$ preserves J-colimits. This sub-category inherits the monoidal structure from $\hat{\mathcal{E}}$.

If $\mathcal{C}$ is a category with J-colimits and $\hat{\mathcal{E}}$ is the (strict) monoidal category of endofunctors over $\mathcal{C}$ (see Example 2.16), then $\hat{\mathcal{E}}$ has J-colimits and $\mathsf{Colim}_J(\hat{\mathcal{E}})$ is the category of endofunctors on $\mathcal{C}$ preserving J-colimits in $\mathcal{C}$. Moreover, a simple way to meet the convergence requirement in Example 2.12 is to work in $\mathsf{Colim}_\omega(\hat{\mathcal{E}})$, where all chains a_β converge at ω. □

Example 2.16. If $\mathcal{C}$ is a category, then the category $\mathsf{Endo}(\mathcal{C})$ of **endofunctors** over $\mathcal{C}$ forms a strict monoidal category $(\mathsf{Endo}(\mathcal{C}), \circ, \mathsf{Id})$, more precisely

objects are endofunctors $F : \mathcal{C} \longrightarrow \mathcal{C}$
arrows from F to G are natural transformations $\tau : F \overset{\bullet}{\longrightarrow} G$
tensor $G \circ F$ is functor composition $(G \circ F)(-) \triangleq G(F(-))$
unit Id is the identity functor $\mathsf{Id}(-) \triangleq -$.

In $\mathsf{Endo}(\mathcal{C})$ an exponential G^F is a right Kan extension of G along F, characterized by a bijection from $H \overset{\bullet}{\longrightarrow} G^F$ to $H \circ F \overset{\bullet}{\longrightarrow} G$ natural in H.

If $\mathcal{C}$ has J*-colimits*, i.e. colimits for diagrams of shape J, then so does $\mathsf{Endo}(\mathcal{C})$, these J-colimits in $\mathsf{Endo}(\mathcal{C})$ are computed pointwise and are preserved by the functors $- \circ F : \mathsf{Endo}(\mathcal{C}) \longrightarrow \mathsf{Endo}(\mathcal{C})$ (similar results hold for limits).

Also the category of **profunctors** $\mathcal{C}^{op} \times \mathcal{C} \longrightarrow$ **Set** forms a monoidal category (see [8]), and there is a monoidal functor from endofunctors to profunctors mapping F to $\mathcal{C}(-_1, F-_2)$. □

Example 2.17. If $\hat{\mathcal{C}}$ is a monoidal category, then the category $\mathsf{Endo}(\hat{\mathcal{C}})_s$ of **strong endofunctors** over $\hat{\mathcal{C}}$ forms a strict monoidal category, more precisely

objects are $\hat{F} = (F, t^F)$ with $F : \mathcal{C} \longrightarrow \mathcal{C}$ functor, $t^F_{a,b} : a \otimes Fb \longrightarrow F(a \otimes b)$ natural transformation such that

$$
\begin{tikzcd}
\mathsf{I} \otimes Fa \arrow[r, "t^F"] \arrow[dr, "\lambda"'] & F(\mathsf{I} \otimes a) \arrow[d, "F(\lambda)"] \\
 & Fa
\end{tikzcd}
\qquad
\begin{tikzcd}
a \otimes (b \otimes Fc) \arrow[r, "\mathsf{id} \otimes t^F"] \arrow[d, "\alpha"] & a \otimes F(b \otimes c) \arrow[r, "t^F"] & F(a \otimes (b \otimes c)) \arrow[d, "F(\alpha)"] \\
(a \otimes b) \otimes Fc \arrow[rr, "t^F"] & & F((a \otimes b) \otimes c)
\end{tikzcd}
$$

arrows from $\hat{F}$ to $\hat{G}$ are natural transformations $\tau : F \overset{\bullet}{\longrightarrow} G$ such that

$$\begin{tikzcd} a \otimes Fb \arrow[r, "\mathsf{id} \otimes \tau"] \arrow[d, "t^F"] & a \otimes Gb \arrow[d, "t^G"] \\ F(a \otimes b) \arrow[r, "\tau"] & G(a \otimes b) \end{tikzcd}$$

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

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

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

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

Example 2.18. We define the category $\mathsf{Endo}(\mathbf{Set})_f$ of **finitary endofunctors** on **Set**. This category inherits the monoidal structure of $\mathsf{Endo}(\mathbf{Set})$, but unlike $\mathsf{Endo}(\mathbf{Set})$ it has exponentials. These results generalize when **Set** is replaced by a *locally finitely presentable enriched* category (see [20]). A finitary endofunctor F on **Set** is *determined* by its action on finite sets (e.g. see [5]), we give two equivalent characterizations

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

We write $\mathsf{Endo}(\mathbf{Set})_f$ for the full sub-category of $\mathsf{Endo}(\mathbf{Set})$ whose objects are finitary endofunctors.

The first characterization implies that Id is finitary, composition of finitary endofunctors is finitary, and the colimit in $\mathsf{Endo}(Set)$ of a diagram in $\mathsf{Endo}(\mathbf{Set})_f$ is in $\mathsf{Endo}(\mathbf{Set})_f$. Therefore, $\mathsf{Endo}(\mathbf{Set})_f$ inherits from $\mathsf{Endo}(\mathbf{Set})$ the monoidal structure and colimits, and the inclusion of $\mathsf{Endo}(\mathbf{Set})_f$ into $\mathsf{Endo}(\mathbf{Set})$ is a strict monoidal functor, which creates and preserves colimits.

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

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

i.e. the coend (see [28, Ch. 9 and 10]) of $S : \mathbf{Set}_f^{op} \times \mathbf{Set}_f \longrightarrow \mathsf{Endo}(\mathbf{Set})$ where $S(m, n) \mathrel{\hat{=}} -^m \times (F_f n)$. In fact, S factors through $\mathsf{Endo}(\mathbf{Set})_f$, as $-^m \times A$ is finitary when $m \in \mathbf{Set}_f$ and $A \in \mathbf{Set}$, thus the coend (which is a colimit) is in $\mathsf{Endo}(\mathbf{Set})_f$, too. The monoidal structure on $\mathsf{Endo}(\mathbf{Set})_f$ induces on $\mathbf{Set}^{\mathbf{Set}_f}$ the following tensor (with unit given by the inclusion functor J)

$$(H \otimes F)a \mathrel{\hat{=}} \int^n (Fa)^n \times (Hn)$$

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

$$(G^F)a \mathrel{\hat{=}} \int_n (Gn)^{(Fn)^a}$$

i.e. the end with parameter for $T : \mathbf{Set}_f \times \mathbf{Set}_f^{op} \times \mathbf{Set}_f \longrightarrow \mathbf{Set}$ where $T(a, m, n) \mathrel{\hat{=}} (Gn)^{(Fm)^a}$. To prove that G^F is an exponential requires general properties of ends and coends, which can be found in [28, Ch. 9]. □

Example 2.19. Consider system $F\omega$ with $\beta\eta$-equivalence (see [3,12]). We define the strict monoidal category $\hat{\mathcal{E}}_{F\omega}$ of endofunctors and natural transformations **expressible** in $F\omega$ (the construction make sense also for other typed calculi). Most results in [17] can be recast as category-theoretic properties of $\hat{\mathcal{E}}_{F\omega}$. For convenience, we recall the syntax of $F\omega$

$$\begin{array}{rl} \text{kinds} & k ::= * \mid k \to k \\ \text{type constructors} & U ::= X \mid U \to U \mid \forall X{:}k.\, U \mid \lambda X{:}k.\, U \mid U\, U \\ \text{terms} & e ::= x \mid \lambda x{:}U.\, e \mid e\, e \mid \Lambda X{:}k.\, e \mid e\, U \end{array}$$

and introduce some notational conventions: we write e_U for $e\,U$ (polymorphic instantiation) and we write definitions $f_X(x : A) \mathrel{\hat{=}} t$ for $f \mathrel{\hat{=}} \Lambda X{:}*.\, \lambda x{:}A.\, t$.

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

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

where, $\mathsf{id}_X \mathrel{\hat{=}} \lambda x : X. x$ is the identity on X and $g \circ f \mathrel{\hat{=}} \lambda x : X. g\,(f\,x)$ is the composition of $g : Y \to Z$ and $f : X \to Y$

arrows from $\hat{F}$ to $\hat{G}$ are *expressible natural transformations*, i.e. $\beta\eta$-equivalence classes $[\tau]$ of closed terms $\tau : \forall X : *. FX \to GX$ such that the following $\beta\eta$-equivalence holds

$$(\mathsf{map}^G_{X,Y}\, f) \circ \tau_A = \tau_B \circ (\mathsf{map}^F_{X,Y}\, f) : FX \to GY$$

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

tensor $\hat{G} \circ \hat{F}$ is $(G \circ F, \mathsf{map})$ with $\mathsf{map}_{X,Y}\,(f : X \to Y) \mathrel{\hat{=}} \mathsf{map}^G_{FX,FY}\,(\mathsf{map}^F_{X,Y}\, f)$.

unit is the pair $(\mathsf{Id}, \mathsf{map})$ with $\mathsf{Id} \mathrel{\hat{=}} \lambda X : *. X$ and $\mathsf{map}_{A,B}\,(f : A \to B) \mathrel{\hat{=}} f$.

$\hat{\mathcal{E}}_{F\omega}$ does not have exponentials, even in the *weak* sense. More specifically, when $\hat{G}$ is the identity functor and $\hat{F}$ is the constant functor $FX = A$ (for some closed type A), there are no natural transformations from $\hat{H} \circ \hat{F}$ to $\hat{G}$, no matter what is $\hat{H}$. In fact, given $\tau : \forall X. H(FX) \to GX$, naturality of $[\tau]$ means that $X, Y : *, f : X \to Y,\ u : HA \vdash f(\tau_X\, u) = \tau_Y\, u : Y$ is a $\beta\eta$-equivalence. However, this is impossible, because the normal form of the lhs contains f free, while the normal form of the rhs does not.

Due to the lack of *weak* exponentials, also some claims in [17] are false. For instance, let $\hat{M}$ and $\hat{K}$ be the expressible functors such that $MX \mathrel{\hat{=}} X$ and $KX \mathrel{\hat{=}} \forall Z : *. (X \to Z) \to Z$, then $\mathsf{from} : \forall X : *. KX \to MX$ given by $\mathsf{from}_X(c : KX) \mathrel{\hat{=}} c_X(\mathsf{id}_X)$ is not a natural transformation from $\hat{K}$ to $\hat{M}$ (as claimed in [17, Proposition 14]). In fact, naturality of from amount to say that $c : KX\, f : X \to Y \vdash f(c_X\, \mathsf{id}_X) = c_Y\, f : Y$ is a $\beta\eta$-equivalence, but this is impossible, because the two terms are different $\beta\eta$-normal forms. $\square$

Example 2.20. Let $(A, \cdot)$ be a *partial combinatory algebra* (see e.g. [26]), i.e. a set A with a partial operation $\cdot : A \times A \rightharpoonup A$, we write $a\,b$ for $\cdot(a, b)$, and two elements $\mathsf{K} \neq \mathsf{S}$ such that $\mathsf{K}\,x\,y = x$, $\mathsf{S}\,x\,y \downarrow$, $\mathsf{S}\,x\,y\,z \simeq x\,z\,(y\,z)$. 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 $\exists x \in X \wedge (\forall a \in A. a \in X \iff aRx)$;

arrows from R_1 to R_2 are maps $f : A/R_1 \longrightarrow A/R_2$ 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 category $\mathsf{Endo}(\mathcal{P}_A)_r$ of **realizable endofunctors** and realizable natural transformations is the sub-category of $\mathsf{Endo}(\mathcal{P}_A)$ such that

objects are endofunctors $F : \mathcal{P}_A \longrightarrow \mathcal{P}_A$ 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 $\tau : F \xrightarrow{\bullet} G$ with a **realizer**, i.e. an $r \in A$ such that $r \vdash_A \tau_R$ for every object R of $\mathcal{P}_A$.

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

- $a\,H(R)\,b \iff a$ and b are realizers for the same realizable natural transformation $\tau : Y_R \otimes F \xrightarrow{\bullet} G$, where Y_R is the realizable endofunctor $-^R$ given by exponentiation to R in $\mathcal{P}_A$
- an arrow $R \xrightarrow{f} S$ in $\mathcal{P}_A$ induces a realizable natural transformation $Y(f) : Y_S \xrightarrow{\bullet} Y_R$ such that $Y(f)_T \mathrel{\hat{=}} T^f$. Therefore, when $Y_R \otimes F \xrightarrow{\tau} G$ is realizable, also $Y_S \otimes F \xrightarrow{Y(f) \otimes \mathsf{id}_F} Y_R \otimes F \xrightarrow{\tau} G$ is. This induces a function $H(f) : A/H(R) \longrightarrow A/H(S)$, and by elementary considerations one can give an $a \in A$ such that $a\,r \vdash_A H(f)$ whenever $r \vdash_A f$

$ev : H \otimes F \xrightarrow{\bullet} G$ is given by $ev_R([a]) \mathrel{\hat{=}} \tau_R(\mathsf{id}_{FR})$, where $\tau : Y_{FR} \otimes F \xrightarrow{\bullet} G$ is the natural transformation realized by a, thus ev is realized by the interpretation of the combinatory term $[x]x([y]y)$. $\square$

Example 2.21. We define the strict monoidal category $\mathsf{Endo}(\mathcal{P}_{F\omega})_r$ of endofunctors and natural transformations *realizable* in $F\omega$. The definition is like that of $\mathsf{Endo}(\mathcal{P}_A)_r$ in Example 2.20, but the partial combinatory algebra $(A, \cdot)$ is replaced by $F\omega$ (more generally, one could use a *partial* second-order combinatory algebra [9]). $\mathsf{Endo}(\mathcal{P}_{F\omega})_r$, like $\mathsf{Endo}(\mathcal{P}_A)_r$, has exponentials.

In what follows 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 $\exists e \in X \wedge (\forall e' \in E(A).\, e' \in X \iff e'Re)$.

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 $f : E(R_1) \longrightarrow E(R_2)$ with a **realizer** $r \vdash f$, i.e. $r \in E(A_1 \to A_2)$ such that $\forall X \in E(R_1).\, \forall e \in X.\, r\, e \in f(X)$.

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

objects are endofunctors $F : \mathcal{P}_{F\omega} \longrightarrow \mathcal{P}_{F\omega}$ with a **realizer** $\hat{F} \vdash F$, i.e. $\hat{F}$ is a pair $(\bar{F}, \mathsf{map}^F)$ with $\bar{F} : * \to *$ closed type constructor (uniquely determined by F modulo $\beta\eta$-equivalence) such that $F(A, R) = (B, S)$ implies $B = \bar{F}A$ and $\mathsf{map}^F \in E(\forall X, Y : *.\, (X \to Y) \to \bar{F}X \to \bar{F}Y)$ such that $f : (A, R) \longrightarrow (B, S)$ in $\mathcal{P}_{F\omega}$ and $e \vdash f$ implies $\mathsf{map}^F_{A,B}\, e \vdash F(f)$;
arrows from F to G are natural transformations $\tau : F \stackrel{\bullet}{\longrightarrow} G$ with a **realizer** $r \vdash \tau$, i.e. $r \in E(\forall X : *.\, \bar{F}X \to \bar{G}X)$ such that $r_A \vdash \tau_{(A,R)}$ for any (A, R).

$\mathsf{Endo}(\mathcal{P}_{F\omega})_r$ inherits the (strict) monoidal structure of $\mathsf{Endo}(\mathcal{P}_{F\omega})$, and the inclusion functor is strict monoidal. We show (by analogy with Example 2.20) that $\mathsf{Endo}(\mathcal{P}_{F\omega})_r$ has an exponential $ev : H \otimes F \longrightarrow G$ for any F and G:

- $H(A, R) \mathrel{\hat{=}} (\forall Z : *.(A \to \bar{F}Z) \to \bar{G}Z, S)$ with $a\, S\, b \iff a$ and b are realizers for the same natural transformation $\tau : Y_{(A,R)} \otimes F \stackrel{\bullet}{\longrightarrow} G$, 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 $(\bar{H}, \mathsf{map}^H)$ with $\bar{H}X \mathrel{\hat{=}} \forall Z : *.\, (X \to \bar{F}Z) \to \bar{G}Z$ and $\mathsf{map}^H_{X,Y}(f : X \to Y, c : \bar{H}X) \mathrel{\hat{=}} \Lambda Z : *.\lambda k : Y \to \bar{F}Z.c_Z(k \circ f)$, which determines the action of H on arrows in $\mathcal{P}_{F\omega}$

$ev : H \otimes F \stackrel{\bullet}{\longrightarrow} G$ is the natural transformation realized by the element r in $E(\forall X.\bar{H}(\bar{F}X) \to \bar{G}X)$ given by $r_X(c : \bar{H}(\bar{F}X)) \mathrel{\hat{=}} c_X(\mathsf{id}_{\bar{F}X})$. $\square$

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 monoid $\hat{M} = (M, e, m)$ and a functor $H : \mathsf{Mon}(\hat{\mathcal{E}}) \longrightarrow \mathcal{E}$, an *$H$-operation* for $\hat{M}$ is a map $\mathsf{op} : H\hat{M} \longrightarrow M$ in $\mathcal{E}$.

A *first-order operation* of *arity* $A \in \mathcal{E}$ for $\hat{M}$ is a map $\mathsf{op} : A \otimes M \longrightarrow M$, i.e. an H-operation for $H(-) = A \otimes U(-)$, and such op is called *algebraic* when

$$s : A, x_1, x_2 : M \vdash \mathsf{op}(s, x_1) \cdot x_2 = \mathsf{op}(s, x_1 \cdot x_2) : M \tag{3.1}$$

Definition 3.2 (*Lifting*)**.** Given an H-operation $\mathsf{op} : H\hat{M}_1 \longrightarrow M_1$ for $\hat{M}_1$ and a monoid map $h : \hat{M}_1 \longrightarrow \hat{M}_2$, an H-operation $\overline{\mathsf{op}} : H\hat{M}_2 \longrightarrow M_2$ for $\hat{M}_2$ is a *lifting of* op *along* h when

$$\begin{tikzcd} H\hat{M}_2 \arrow[r, "\overline{\mathsf{op}}"] & M_2 \\ H\hat{M}_1 \arrow[u, "Hh"] \arrow[r, "\mathsf{op}"'] & M_1 \arrow[u, "Uh"'] \end{tikzcd} \tag{3.2}$$

Remark 3.3. Eq. (3.1) is equivalent to

$$s : A, x : M \vdash \mathsf{op}(s, x) = \mathsf{op}(s, e) \cdot x : M \tag{3.3}$$

From this it is immediate to establish a bijective correspondence between algebraic operations $\mathsf{op} : A \otimes M \longrightarrow M$ for $\hat{M}$ and maps $\mathsf{op}' : A \longrightarrow M$

$$\mathsf{op}'(s : A) : M \mathrel{\hat{=}} \mathsf{op}(s, e)$$
$$\mathsf{op}(s : A, x : M) : M \mathrel{\hat{=}} \mathsf{op}'(s) \cdot x$$

Diagram (3.2) is equivalent to the equation

$$s : A, x : M_1 \vdash h(\mathsf{op}(s, x)) = \overline{\mathsf{op}}(s, h(x)) : M_2 \tag{3.4}$$

when $H(-) = A \otimes U(-)$.

Theorem 3.4 (*Unique Algebraic Lifting*). *Given $h : \hat{M}_1 \longrightarrow \hat{M}_2$ monoid map and $\mathsf{op} : A \otimes M_1 \longrightarrow M_1$ algebraic for $\hat{M}_1$, let $\mathsf{op}^\sharp : A \otimes M_2 \longrightarrow M_2$ be*

$$\mathsf{op}^\sharp(s : A, x : M_2) : M_2 \mathrel{\hat{=}} h(\mathsf{op}(s, e_1)) \cdot_2 x \tag{3.5}$$

then $\mathsf{op}^\sharp$ is the unique lifting of op along h which is algebraic for $\hat{M}_2$.

Proof. By definition $\mathsf{op}^\sharp$ is algebraic for $\hat{M}_2$. Let Eq be the set of equations saying that $h : \hat{M}_1 \longrightarrow \hat{M}_2$ and $\mathsf{op} : A \otimes M_1 \longrightarrow M_1$ is algebraic for $\hat{M}_1$. Let $Eqop$ be Eq plus the equations saying that $\overline{\mathsf{op}} : A \otimes M_2 \longrightarrow M_2$ is algebraic for $\hat{M}_2$ and is a lifting of op along h. The claims that $\mathsf{op}^\sharp$ is a lifting of op along h and uniqueness amount to the following equations

- $s : A, x : M_1 \vdash_{Eq} \mathsf{op}^\sharp(s, h(x)) = h(\mathsf{op}(s, x)) : M_2$
  $\mathsf{op}^\sharp(s, h(x))$ — by definition
  $h(\mathsf{op}(s, e_1)) \cdot_2 h(x)$ — by (2.16) in Eq
  $h(\underline{\mathsf{op}(s, e_1) \cdot_1 x})$ — by (3.3) in Eq
  $h(\overline{\mathsf{op}(s, x)})$
- $s : A, x : M_2 \vdash_{Eqop} \overline{\mathsf{op}}(s, x) = \mathsf{op}^\sharp(s, x) : M_2$
  $\underline{\overline{\mathsf{op}}(s, x)}$ — by (3.3) in $Eqop$
  $\overline{\mathsf{op}}(s, \underline{e_2}) \cdot_2 x$ — by (2.15) in $Eqop$
  $\underline{\overline{\mathsf{op}}(s, \overline{h(e_1)})} \cdot_2 x$ — by (3.4) in $Eqop$
  $\overline{h(\mathsf{op}(s, e_1))} \cdot_2 x$ — by definition
  $\mathsf{op}^\sharp(s, x)$ □

Remark 3.5. An algebraic operation may have several liftings along a monoid map. For instance, take **Set** with the monoidal structure given by finite products (see Example 2.14), a monoid $\hat{M} = (M, e, \cdot)$ and an $\mathsf{op} : M \longrightarrow M$ algebraic for $\hat{M}$, i.e. $\mathsf{op}(x) = \mathsf{op}' \cdot x$ where $\mathsf{op}' = \mathsf{op}(e)$. Define the monoids $\hat{2} \mathrel{\hat{=}} (\{0, 1\}, 1, *)$ and $\hat{N} \mathrel{\hat{=}} \hat{M} \times \hat{2}$, and consider the monoid map $h : \hat{M} \longrightarrow \hat{N}$ given by $h(x) \mathrel{\hat{=}} (x, 1)$. The unique algebraic lifting of op along h is $\mathsf{op}^\sharp(x, b) = (\mathsf{op}' \cdot x, b)$, a different lifting of op along h is given by $\overline{\mathsf{op}}(x, b) \mathrel{\hat{=}} (\mathsf{op}' \cdot x, 1)$. □

3.1. Examples of operations

Among the different *flavours* of monads, strong monads are those needed to interpret the monadic metalanguage of [31, 32]. In this section we give examples of strong monads (on a cartesian closed category) and associated operations, saying whether the operations are algebraic, first-order or H-operations. 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 *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 $\hat{M} = (M, \mathsf{ret}^M, \mathsf{bind}^M)$ consisting of

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

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

$$\mathsf{bind}^M_{A,B}(\mathsf{ret}^M_A(a), f) = f\ a$$

$$\mathsf{bind}^M_{A,A}(u,\ \mathsf{ret}^M_A) = u$$

$$\mathsf{bind}^M_{A,C}(u,\ \lambda a : A.\ \mathsf{bind}^M_{B,C}(f\ a,\ g)) = \mathsf{bind}^M_{B,C}(\mathsf{bind}^M_{A,B}(u, f),\ g)$$

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

$$\tau_A\ (\mathsf{ret}^M_A(a)) = \mathsf{ret}^N_A(a)$$

$$\tau_B\ (\mathsf{bind}^M_{A,B}(u, f)) = \mathsf{bind}^N_{A,B}(\tau_A\, u,\ \lambda a : A.\ \tau_B\ (f\ a))$$

Remark 3.7. In the monoidal category $\mathsf{Endo}(\mathcal{C})_s$ of strong endofunctors on a cartesian closed category $\mathcal{C}$ what is usually meant by an algebraic operation for a strong monad $\hat{M}$ (e.g. see [35]) is an algebraic operation (in the sense of Definition 3.1) of arity $A(X) = J \times X^I$ (with $I, J \in \mathcal{C}$) for $\hat{M}$. For these algebraic operations there is another bijective correspondence, in addition to the one given in Remark 3.3, namely between algebraic operations $\mathsf{op}_X : J \times (MX)^I \longrightarrow MX$ for $\hat{M}$ and maps $\mathsf{op}'' : J \longrightarrow MI$ in $\mathcal{C}$

$$\mathsf{op}''(j : J) : MI \mathrel{\hat{=}} \mathsf{op}_I(j, \mathsf{ret}^M_I)$$
$$\mathsf{op}_X(j : J, f : (MX)^I) : MX \mathrel{\hat{=}} \mathsf{bind}^M_{I,X}(\mathsf{op}''(j), f)$$

This correspondence does not hold when $\mathsf{Endo}(\mathcal{C})_s$ is replaced by $\mathsf{Endo}(\mathcal{C})$, and does not give *improved* lifting results over Theorem 3.4. □

Example 3.8. The monad $\hat{M} = (M, \mathsf{ret}^M, \mathsf{bind}^M)$ of continuations in R is

$$MX \mathrel{\hat{=}} R^{(R^X)}$$
$$\mathsf{ret}^M_X(x : X) \mathrel{\hat{=}} \lambda k : R^X.\, k\, x$$
$$\mathsf{bind}^M_{X,Y}(m : MX,\ f : MY^X) \mathrel{\hat{=}} \lambda k : R^Y.\, m\, (\lambda x : X.\, f\, x\, k)$$

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

$$\mathsf{abort}_X\ (r : R) \mathrel{\hat{=}} \lambda k : R^X.\, r$$
$$\mathsf{callcc}_X(f : (MX)^{(R^{MX})}) \mathrel{\hat{=}} \lambda k : R^X.\, f\, (\lambda t : MX.\, t\, k)\, k$$

Usually, the associated operation is $\underline{\mathsf{callcc}}_{X,Y} : (MX)^{((MY)^X)} \longrightarrow MX$, which is *definable* from callcc, abort, unit and bind of the monad (see [17]). □

Example 3.9. The monad $\hat{M} = (M, \mathsf{ret}^M, \mathsf{bind}^M)$ of environments in S is

$$MX \mathrel{\hat{=}} X^S$$
$$\mathsf{ret}^M_X(x : X) \mathrel{\hat{=}} \lambda s : S.\, x$$
$$\mathsf{bind}^M_{X,Y}(m : MX,\ f : MY^X) \mathrel{\hat{=}} \lambda s : S.\, f\, (m\, s)\, s$$

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

$$\mathsf{read}_X\ (f : (MX)^S) \mathrel{\hat{=}} \lambda s : S.\, f\, s\, s$$
$$\mathsf{local}_X(f : S^S, t : MX) \mathrel{\hat{=}} \lambda s : S.\, t\, (f\, s) \quad \Box$$

Example 3.10. The monad $\hat{M} = (M, \mathsf{ret}^M, \mathsf{bind}^M)$ of side-effects on S is

$$MX \mathrel{\hat{=}} (X \times S)^S$$
$$\mathsf{ret}^M_X(x : X) \mathrel{\hat{=}} \lambda s : S.\, (x, s)$$
$$\mathsf{bind}^M_{X,Y}(m : MX,\ f : MY^X) \mathrel{\hat{=}} \lambda s : S.\, \mathsf{let}\ (a, s') = m\, s\ \mathsf{in}\ f\, a\, s'$$

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

$$\mathsf{read}_X\ (k : (MX)^S) \mathrel{\hat{=}} \lambda s : S.\, k\, s\, s$$
$$\mathsf{write}_X\ (s : S, m : MX) \mathrel{\hat{=}} \lambda s' : S.\, m\, s \quad \Box$$

Example 3.11. The monad $\hat{M} = (M, \mathsf{ret}^M, \mathsf{bind}^M)$ of complexity on a monoid $(W, 0, +)$ in $\mathcal{C}$ is

$$MX \mathrel{\hat{=}} X \times W$$
$$\mathsf{ret}^M_X(x : X) \mathrel{\hat{=}} (x, 0)$$
$$\mathsf{bind}^M_{X,Y}((x, w) : MX,\ f : MY^X) \mathrel{\hat{=}} \mathsf{let}\ (y, w') = f\, x\ \mathsf{in}\ (y, w + w')$$

It has an algebraic operation for the functor $A_{\mathsf{add}}X = X \times W$ and H-operations for the functors $H_{\mathsf{collect}_A}\hat{M}X = MA \times X^{(A \times W)}$, namely

$$\mathsf{add}_X\ (t : MX, w : W) \mathrel{\hat{=}} \mathsf{let}\ (x, w') = t\ \mathsf{in}\ (x, w' + w)$$
$$\mathsf{collect}_{A,X}(t : MA, f : X^{(A \times W)}) \mathrel{\hat{=}} \mathsf{let}\ (y, w) = t\ \mathsf{in}\ (f\, t, w)$$

Usually the associated operation is $\underline{\mathsf{collect}}_X : MX \longrightarrow M(X \times W)$, which is *definable* from the operations $\mathsf{collect}_A$, unit and bind of the monad. □

Example 3.12. When $\mathcal{C}$ has binary sums, the monad $\hat{M} = (M, \mathsf{ret}^M, \mathsf{bind}^M)$ of exceptions in E is

$$MX \hat{=} X + E$$
$$\mathsf{ret}^M_X(x : X) \hat{=} \mathsf{inl}\, x$$
$$\mathsf{bind}^M_{X,Y}(m : MX, f : MY^X) \hat{=} [f, \mathsf{inr}]\, m$$

It has an algebraic operation for the functor $A_{\mathsf{throw}}X = E$ and a first-order operation (but not algebraic) for the functor $A_{\mathsf{handle}}X = X \times X^E$, namely

$$\mathsf{throw}_X\, (e : E) \hat{=} \mathsf{inr}\, e$$
$$\mathsf{handle}_X\, (m : MX,\ h : (MX)^E) \hat{=} [\mathsf{inl}, h](m) \quad \square$$

Example 3.13. Algebraic theories [29] are presented by operations and equations. More precisely, an algebraic theory $T = (\Sigma, Eq)$ consists of a signature $\Sigma = (O_n \mid n \in N)$, where O_n is the set of operations of arity n, and a set Eq of equations (between Σ-terms). They are a way to define monads and associated operations (see [20] for generalizations of equational theories that go beyond **Set**). In fact, an algebraic theory T induces a monoid $\hat{M}_T$ in $\mathsf{Endo}(\mathbf{Set})_f$ (see Example 2.18), i.e. a finitary monad[3] on **Set**. Conversely, every monoid in $\mathsf{Endo}(\mathbf{Set})_f$ is isomorphic to some $\hat{M}_T$. The monad $\hat{M}_T$ has an algebraic operation $o_X : (M_T X)^n \longrightarrow M_T X$ for each $o \in O_n$, where o_X is the interpretation of o in the free T-algebra over X. These operations can be collected in one algebraic operation $\mathsf{op}_X : \Sigma(M_T X) \longrightarrow M_T X$, where Σ is the finitary endofunctor $\Sigma(X) \hat{=} \coprod_{n\in N} O_n \times X^n$.

All monads for *collection types* (such as lists, bags, sets) arise from *balanced* finitary algebraic theories [30]. The monad in Example 3.8 is finitary when the set R has at most one element. The monads of Examples 3.9 and 3.10 are finitary when the set S is finite. For instance, the monad $MX = (X \times S)^S$ corresponds to the algebraic theory [36] given by an operation read of arity $|S|$, unary operations write_s for $s \in S$, and equations

$$t = \mathsf{read}(t \mid i \in S)$$
$$\mathsf{read}(\mathsf{read}(t_{i,j} \mid j \in S) \mid i \in S) = \mathsf{read}(t_{i,i} \mid i \in S)$$
$$\mathsf{read}(t_i \mid i \in S) = \mathsf{read}(\mathsf{write}_i(t_i) \mid i \in S)$$
$$\mathsf{write}_i(\mathsf{read}(t_j \mid j \in S)) = \mathsf{write}_i(t_i) \quad \text{with } i \in S$$
$$\mathsf{write}_i(\mathsf{write}_j(t)) = \mathsf{write}_j(t) \quad \text{with } i, j \in S$$

The monads of Examples 3.11 and 3.12 are always finitary. When $\hat{M}$ is the free monad on Σ, i.e. the monad induced by the algebraic theory $T = (\Sigma, \emptyset)$, one can associate to $\hat{M}$ two other operations

- $\mathsf{elim}_X : X^{\Sigma X} \times X^A \longrightarrow X^{MA}$ captures *initiality* of MA among the Σ-algebras over A, namely $\mathsf{elim}_X(\alpha, f)$ is the unique Σ-homomorphism f^* from $\Sigma(MA) \xrightarrow{\mathsf{op}_A} MA$ (the free algebra over A) to $\Sigma X \xrightarrow{\alpha} X$ such that $f^* \circ \mathsf{ret}^M_A = f$. elim generalises $\mathsf{bind}^M_{A,X}$ (see the try construct in [37]), and usually cannot be presented as an H-operation.
- $\underline{\mathsf{case}}_X : MA \times X^A \times X^{\Sigma(MA)} \longrightarrow X$ does case analysis on MA, which is isomorphic to $A + \Sigma(MA)$. The instance of $\underline{\mathsf{case}}$ obtained by replacing X with MX, i.e. $\mathsf{case}_X : MA \times (MX)^A \times (MX)^{\Sigma(MA)} \longrightarrow MX$, can be presented as an H-operation for $H\hat{N}X \hat{=} NA \times (NX)^A \times (NX)^{\Sigma(MA)}$, provided the M in contravariant position is fixed. $\square$

4. Monoid transformers

This section introduces a taxonomy of *monoid transformers* in the setting of a monoidal category $\hat{\mathcal{E}}$ and gives examples of monoid transformers motivated by the incremental approach to monadic semantics. The main motivation for the taxonomy are the solutions to the lifting problem given in Section 5, which depend on where a transformer fits in the taxonomy.

The minimum requirement on a monoid transformer T is to map a monoid $\hat{M} \in \mathsf{Mon}(\hat{\mathcal{E}})$ to a monoid $T\hat{M}$ (and a monoid morphism $\hat{M} \longrightarrow T\hat{M}$). The maximum requirement is when the monoid transformer T is *induced* by a monoidal endofunctor $\hat{T}$ on $\hat{\mathcal{E}}$. In the rest of this section we call monoid transformers simply transformers.

Definition 4.1 (*Monoid Transformers*). Let $\hat{\mathcal{E}}$ be a monoidal category, and $\mathcal{M}$ be the category $\mathsf{Mon}(\hat{\mathcal{E}})$ of monoids in $\hat{\mathcal{E}}$, then

1. A *basic* transformer (T, in) is a 2-cell $|\mathcal{M}| \underset{T}{\overset{\mathsf{In}}{\rightrightarrows}} \mathcal{M}$ with $\Downarrow \mathsf{in}$ (in the 2-category of categories), where $|\mathcal{M}|$ is the *discrete sub-category* of $\mathcal{M}$ and In is the *inclusion functor*
2. A *covariant* transformer (T, in) is a 2-cell $\mathcal{M} \underset{T}{\overset{\mathsf{Id}}{\rightrightarrows}} \mathcal{M}$ with $\Downarrow \mathsf{in}$

[3] In **Set** every monad is strong.

3. A *functorial* transformer is a covariant transformer (T, in) and a 2-cell $\mathcal{E}\ \underset{T}{\overset{\mathsf{Id}}{\Downarrow \mathsf{in}}}\ \mathcal{E}$ such that $U \circ T = T \circ U$ and $U(\mathsf{in}_{-}) = \mathsf{in}_{U(-)}$, i.e. $\mathcal{M}\ \underset{T}{\overset{\mathsf{Id}}{\Downarrow \mathsf{in}}}\ \mathcal{M} \xrightarrow{U} \mathcal{E} = \mathcal{M} \xrightarrow{U} \mathcal{E}\ \underset{T}{\overset{\mathsf{Id}}{\Downarrow \mathsf{in}}}\ \mathcal{E}$
4. A *monoidal* transformer is a 2-cell $\hat{\mathcal{E}}\ \underset{\hat{T}}{\overset{\hat{\mathsf{Id}}}{\Downarrow \mathsf{in}}}\ \hat{\mathcal{E}}$ (in the 2-category of monoidal categories), i.e. $\hat{T}$ is a monoidal functor and in is a monoidal natural transformation.

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

monoidal $\Longrightarrow$ *functorial* $\Longrightarrow$ *covariant* $\Longrightarrow$ *basic.*

Proof. Immediate from the definitions and Theorem 2.6 □

Remark 4.3. Also the monad/theory combinations proposed in [27,15] have a natural generalization in the setting of a monoidal category, namely a *monoid combination* is a bifunctor $\otimes_C : \mathcal{M} \times \mathcal{M} \longrightarrow \mathcal{M}$, which makes $\mathcal{M}$ into a monoidal category with $\hat{\mathrm{I}}$ as unit. Since $\hat{\mathrm{I}}$ is the initial monoid, one can define a pair of 2-cells $\mathcal{M} \times \mathcal{M}\ \underset{\otimes_C}{\overset{\pi_i}{\Downarrow \mathsf{in}_i}}\ \mathcal{M}$ for $i = 1, 2$. Thus, every monoid $\hat{M}$ *induces* a covariant transformer $T(-) \mathrel{\hat{=}} \hat{M} \otimes_C -$, by fixing the first monoid in the combination. However, there are functorial transformers, which are not of the form $\hat{M} \otimes_C -$, for some choice of $\otimes_C$ and $\hat{M}$. A simple counter-example in the category $\mathcal{M}$ of finitary monads on **Set** (or equivalently algebraic theories) is the list transformer $TMX = \mu X'.M(1{+}X{\times}X')$, described in Example 4.9. At the level of algebraic theories (see Example 3.13) the list transformer T maps a presentation (Σ, Eq) to the presentation obtained by adding to (Σ, Eq) a binary (infix) operation @, a constant nil, and the equations

$$\mathsf{nil}@x = x = x@\mathsf{nil} \qquad (x@y)@z = x@(y@z)$$
$$\mathsf{op}(x_i|i \in n)@y = \mathsf{op}(x_i@y|i \in n) \quad \text{for any } \mathsf{op} \in \Sigma \text{ of arity } n$$

We are unaware of simple conditions on $\otimes_C$ and $\hat{M}$ implying that the induced transformer $T(-) = \hat{M} \otimes_X -$ is functorial or monoidal. Such implications would be of interest to extend our lifting results to combinations. □

4.1. Examples of transformers

We give examples of strong monad transformers, i.e. monoid transformers on the monoidal category $\mathsf{Endo}(\mathcal{C})_s$ with $\mathcal{C}$ cartesian closed, and say where they fit in the taxonomy. Some examples require additional assumptions on $\mathcal{C}$ and use a monoidal sub-category of $\mathsf{Endo}(\mathcal{C})_s$.

- The transformers $TMX = MX^S$ (Example 4.5), $TMX = M(X \times S)^S$ (Example 4.6) and $TMX = M(X \times W)$ (Example 4.7) are monoidal.
- The transformer $TMX = \mu X'.M(X + SX')$ (Example 4.8) is functorial, but not monoidal. By a suitable choice of S this transformer becomes $TMX = M(X + E)$ for exceptions, $TMX = \mu X'.M(X + X')$ for resumptions, $TMX = \mu X'.M(X + V \times X' + X'^V)$ for interactive I/O.
- The transformer $TMX = \mu X', M(1 + X \times X')$ (Example 4.9) is covariant, but not functorial.

Finally, the monoid transformers in Example 4.10 show that the implications in Proposition 4.2 cannot be reversed.

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

Definition 4.4 (*Strong Endofunctor*). A *strong endofunctor* on a cartesian closed category $\mathcal{C}$ is a pair $\hat{F} = (F, \mathsf{map}^F)$ consisting of

- a map $F : |\mathcal{C}| \longrightarrow |\mathcal{C}|$ on the objects of $\mathcal{C}$
- a family $\mathsf{map}^F_{X,Y} : Y^X \times FX \longrightarrow FY$ of maps with $X, Y \in \mathcal{C}$

such that for every $u : FA, f : B^A$ and $g : C^B$:

$$\mathsf{map}^F_{A,A}(\mathsf{id}_A,\ u) = u$$
$$\mathsf{map}^F_{A,C}(g \circ f,\ u) = \mathsf{map}^F_{B,C}(g,\ \mathsf{map}^F_{A,B}(f,\ u))$$

A *strong natural transformation* $\tau : \hat{F} \longrightarrow \hat{G}$ is a family $\tau_X : FX \longrightarrow GX$ of maps with $X \in \mathcal{C}$ such that for every $u : FA$ and $f : B^A$

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

Example 4.5. The transformer (T, in) for adding *environments* in $S \in \mathcal{C}$ is defined as follows:

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

$$NX \mathrel{\hat{=}} MX^S$$
$$\mathsf{ret}^N_X(x) \mathrel{\hat{=}} \lambda s : S.\, \mathsf{ret}^M_X(x)$$
$$\mathsf{bind}^N_{X,Y}(c,\ f) \mathrel{\hat{=}} \lambda s : S.\, \mathsf{bind}^M_{X,Y}(c\, s,\ \lambda x : X.\, f\, x\, s)$$

- in maps a strong monad $\hat{M}$ to $\tau : \hat{M} \longrightarrow T\hat{M}$ given by

$$\tau_X(c : MX) \mathrel{\hat{=}} \lambda s : S.\, c$$

This transformer is monoidal. More precisely, it is induced by the following monoidal functor $\hat{T} = (T, \phi_{\mathsf{I}}, \phi)$ and monoidal natural transformation in

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

$$GX \mathrel{\hat{=}} (FX)^S$$
$$\mathsf{map}^G_{X,Y}(f,\ u) \mathrel{\hat{=}} \lambda s : S.\, \mathsf{map}^F_{X,Y}(f,\ u\, s)$$

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

$$(T\tau)_X(u) \mathrel{\hat{=}} \lambda s : S.\, \tau_X(u\, s)$$

- $\phi_{\mathsf{I}} : \mathsf{Id} \xrightarrow{\bullet} T(\mathsf{Id})$ and $\phi_{\hat{F}_2,\hat{F}_1} : T\hat{F}_2 \circ T\hat{F}_1 \xrightarrow{\bullet} T(\hat{F}_2 \circ \hat{F}_1)$ are

$$\phi_{\mathsf{I},X}(x : X) \mathrel{\hat{=}} \lambda s : S.\, x$$
$$\phi_{\hat{F}_2,\hat{F}_1,X}(u : F_2((F_1X)^S)^S) \mathrel{\hat{=}} \lambda s : S.\, \mathsf{map}^{F_2}_{(F_1X)^S,F_1X}(\lambda f : (F_1X)^S.\, f\, s,\ u\, s)$$

- $\mathsf{in}_{\hat{F}} : \hat{F} \xrightarrow{\bullet} T\hat{F}$ is $\mathsf{in}_{\hat{F},X}(u : FX) \mathrel{\hat{=}} \lambda s : S.\, u$ □

Example 4.6. The transformer (T, in) for adding *side-effects* on $S \in \mathcal{C}$ is defined as follows:

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

$$NX \mathrel{\hat{=}} M(X \times S)^S$$
$$\mathsf{ret}^N_X(x) \mathrel{\hat{=}} \lambda s : S.\, \mathsf{ret}^M_{X\times S}(x, s)$$
$$\mathsf{bind}^N_{X,Y}(c,\ f) \mathrel{\hat{=}} \lambda s : S.\, \mathsf{bind}^M_{X\times S,Y\times S}(c\, s,\ \lambda(x : X, s' : S).\, f\, x\, s')$$

- in maps a strong monad $\hat{M}$ to $\tau : \hat{M} \longrightarrow T\hat{M}$ given by

$$\tau_X(c : MX) \mathrel{\hat{=}} \lambda s : S.\, \mathsf{bind}^M_{X,X\times S}(c,\ \lambda x : X.\, \mathsf{ret}^M_{X\times S}(x, s))$$

Also this transformer is monoidal. More precisely, it is induced by the following monoidal functor $\hat{T}$ and monoidal natural transformation in

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

$$GX \mathrel{\hat{=}} F(X \times S)^S$$
$$\mathsf{map}^G_{X,Y}(f,\ u) \mathrel{\hat{=}} \lambda s : S.\, \mathsf{map}^F_{X\times S,Y\times S}(\lambda(x : X, s' : S).\, (f\, x, s'),\ u\, s)$$

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

$$(T\tau)_X(u) \mathrel{\hat{=}} \lambda s : S.\, \tau_{X\times S}(u\, s)$$

- $\phi_{\mathsf{I}} : \mathsf{Id} \xrightarrow{\bullet} T(\mathsf{Id})$ and $\phi_{\hat{F}_2,\hat{F}_1} : T\hat{F}_2 \circ T\hat{F}_1 \xrightarrow{\bullet} T(\hat{F}_2 \circ \hat{F}_1)$ are

$$\phi_{\mathsf{I},X}(x : X) \mathrel{\hat{=}} \lambda s : S.\, (x, s)$$
$$\phi_{\hat{F}_2,\hat{F}_1,X}(u : (F_2((F_1X \times S)^S \times S))^S) \mathrel{\hat{=}} \lambda s : S.\, \mathsf{map}^{F_2}_{F_1(X\times S)^S\times S,F_1(X\times S)}(\lambda(f : F_1(X \times S)^S, s' : S).\, f\, s',\ u\, s)$$

- $\mathsf{in}_{\hat{F}} : \hat{F} \xrightarrow{\bullet} T\hat{F}$ is $\mathsf{in}_{\hat{F},X}(u : FX) \mathrel{\hat{=}} \lambda s : S.\, \mathsf{map}^F_{X,X\times S}(\lambda x : X.\, (x, s),\ u)$ □

Example 4.7. The transformer (T, in) for adding *complexity* on a monoid $(W, 0, +)$ in $\mathcal{C}$ is defined as follows:

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

$$NX \mathrel{\hat{=}} M(X \times W)$$
$$\mathsf{ret}^N_X(x) \mathrel{\hat{=}} \mathsf{ret}^M_{X\times W}(x, 0)$$
$$\mathsf{bind}^N_{X,Y}(c, f) \;\mathrel{\hat{=}}\; \mathsf{bind}^M(c, \lambda(x : X, w : W).\, \mathsf{bind}^M(f\,x,\ \lambda(y : Y, w' : W).\, \mathsf{ret}^M(y, w + w')))$$

- in maps a strong monad $\hat{M}$ to $\tau : \hat{M} \longrightarrow T\hat{M}$ given by

$$\tau_X(c : MX) \mathrel{\hat{=}} \mathsf{bind}^M_{X,X\times W}(c,\ \lambda x : X.\, \mathsf{ret}^M_{X\times W}(x, 0))$$

Also this transformer is monoidal (we skip the details). □

Example 4.8. In this example we need additional assumptions on $\mathcal{C}$, namely

- existence of binary sums

$$\begin{tikzcd} A_1 \arrow[r, "\mathsf{inl}"] \arrow[dr, "f_1"'] & A_1 + A_2 \arrow[d, dashed, "{[f_1, f_2]}"] & A_2 \arrow[l, "\mathsf{inr}"'] \arrow[dl, "f_2"] \\ & A & \end{tikzcd}$$

  (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 $\hat{F}$.

In order to satisfy the last assumption one could take as $\mathcal{C}$ the cartesian closed category $\mathcal{P}_A$ of partial equivalence relations, and replace $\mathsf{Endo}(\mathcal{P}_A)_s$ with the more restricted category $\mathsf{Endo}(\mathcal{P}_A)_r$ of realizable endofunctors and realizable natural transformations (see Example 2.20). Alternatively, one could take the category of finitary endofunctors (see Example 2.18) or the category of containers [1] which are also closed under initial algebras. Given a realizable endofunctor $\hat{S}$, the transformer (T, in) for adding $\hat{S}$-*steps* is defined as follows:

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

$$NX \mathrel{\hat{=}} \mu X'.\, M(X + SX')$$
$$\mathsf{ret}^N_X(x) \mathrel{\hat{=}} \alpha(\mathsf{ret}^M_{X+S(NX)}(\mathsf{inl}\, x))$$
$$\mathsf{step}_X : S(NX) \longrightarrow NX$$
$$\mathsf{step}_X(u) \mathrel{\hat{=}} \alpha(\mathsf{ret}^M_{X+S(NX)}(\mathsf{inr}\, u))$$
$$\mathsf{bind}^N_{X,Y}(c, f) \mathrel{\hat{=}} h\, c$$

  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

$$\beta(c) \mathrel{\hat{=}} \alpha(\mathsf{bind}^M_{X+S(NY),Y+S(NY)}(c,\ \alpha^{-1} \circ [f, \mathsf{step}_Y]))$$

- in maps a realizable monad $\hat{M}$ to $\tau : \hat{M} \longrightarrow \hat{N} = T\hat{M}$ given by

$$\tau_X(c : MX) \mathrel{\hat{=}} \alpha(\mathsf{bind}^M_{X,X+S(NX)}(c,\ \alpha^{-1} \circ \mathsf{ret}^N_X))$$

This transformer is functorial. More precisely, the underlying realizable endofunctor transformer (T, in) is

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

$$GX \mathrel{\hat{=}} \mu X'.\, F(X + SX')$$
$$\mathsf{map}^G_{X,Y}(f, u) \mathrel{\hat{=}} h\, u$$

  where $GX \xrightarrow{h} GY$ is the unique $F(X + S-)$-algebra morphism from the initial algebra to $\beta : F(X + S(GY)) \longrightarrow GY$ given by

$$\beta(u) \mathrel{\hat{=}} \alpha(\mathsf{map}^F_{X+S(GY),Y+S(GY)}(f + \mathsf{id}_{S(GY)},\ u))$$

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

$$(T\tau)_X(u) \mathrel{\hat{=}} h\, u$$

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

$$\beta(u) \mathrel{\hat{=}} \alpha(\tau_{X+S(G_2X)}(u))$$

- in maps a realizable endofunctor $\hat{F}$ to $\tau : \hat{F} \longrightarrow \hat{G} = T\hat{F}$ given by

$$\tau_X(u : FX) \mathrel{\hat{=}} \alpha(\mathsf{map}^F_{X,X+S(GX)}(\mathsf{inl},\ u))$$

This transformer may fail to be monoidal (see Example 4.10). □

Example 4.9. We define the list transformer, which needs additional assumptions, like those identified in Example 4.8. Therefore, we take as $\mathcal{C}$ the cartesian closed category $\mathcal{P}_A$ of partial equivalence relations, and replace $\mathsf{Endo}(\mathcal{P}_A)_s$ with the more restricted category $\mathsf{Endo}(\mathcal{P}_A)_r$ of realizable endofunctors and realizable natural transformations. The *list* transformer (T, in) is defined as follows:

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

$$\begin{aligned}
&NX \mathrel{\hat{=}} \mu X'.\, M(1 + X \times X') \\
&\mathsf{nil}_X : NX \\
&\mathsf{nil}_X \mathrel{\hat{=}} \alpha(\mathsf{ret}^M_{1+X\times NX}(\mathsf{inl}\, *)) \\
&\mathsf{cons}_X : X \times NX \longrightarrow NX \\
&\mathsf{cons}_X(x, l) \mathrel{\hat{=}} \alpha(\mathsf{ret}^M_{1+X\times NX}(\mathsf{inr}(x, l))) \\
&\mathsf{ret}^N_X(x) \mathrel{\hat{=}} \mathsf{cons}_X(x, \mathsf{nil}_X) \\
&\mathsf{bind}^N_{X,Y}(c, f) \mathrel{\hat{=}} h\, c
\end{aligned}$$

  where $NX \xrightarrow{h} NY$ is the unique $M(1+X\times -)$-algebra morphism from the initial algebra to $\beta : M(1+X\times NY) \longrightarrow NY$ given by

$$\beta(c) \mathrel{\hat{=}} \alpha(\mathsf{bind}^M_{1+X\times NY,1+Y\times NY}(c,\ \alpha^{-1} \circ [\mathsf{nil}_Y,\ \lambda(x, l).\, \mathsf{app}_Y((f\, x), l)]))$$

  with $NX \xrightarrow{\Lambda \mathsf{app}_X} (NX)^{NX}$ the unique $M(1+X\times -)$-algebra from the initial algebra to $\Lambda\beta : M(1+X\times (NX)^{NX}) \longrightarrow (NX)^{NX}$ where β is given by

$$\beta(c, l) \mathrel{\hat{=}} \alpha(\mathsf{bind}^M_{1+X\times(NX)^{NX},1+X\times NX}(c,\ \alpha^{-1} \circ [\mathsf{nil}_X,\ \lambda(x, f).\, \mathsf{cons}_X(x, f\, l)]))$$

  To prove that ret^N and bind^N satisfy the equations in Definition 3.6, one can use the following properties of nil_X, cons_X and app_X

$$\begin{aligned}
&\mathsf{app}_X(\mathsf{nil}_X, l) = l = \mathsf{app}_X(l, \mathsf{nil}_X) \\
&\mathsf{app}_X(\mathsf{cons}_X(x, l_1), l_2) = \mathsf{cons}_X(x, \mathsf{app}_X(l_1, l_2)) \\
&\mathsf{app}_X(\mathsf{app}_X(l_1, l_2), l_3) = \mathsf{app}_X(l_1, \mathsf{app}_X(l_2, l_3))
\end{aligned}$$

- in maps a realizable monad $\hat{M}$ to $\tau : \hat{M} \longrightarrow \hat{N} = T\hat{M}$ given by

$$\tau_X(c : MX) \mathrel{\hat{=}} \alpha(\mathsf{bind}^M_{X,1+X\times NX}(c,\ \alpha^{-1} \circ \mathsf{ret}^N_X))$$

This transformer is covariant, but not functorial. In fact, take the endofunctor $MX = X \times N$, where $N \in \mathcal{C}$ is the natural numbers object. Consider the two monoid $\hat{N}_1 \mathrel{\hat{=}} (N, 0, +)$ and $\hat{N}_2 \mathrel{\hat{=}} (N, 1, *)$ with N as carrier, they induce different monads $\hat{M}_i$ with M as underlying endofunctor. The natural transformations $\mathsf{in}_{\hat{M}_i} : MX \longrightarrow TMX$ are different, and so they are not determined by the underlying endofunctor (as required in the definition of functorial transformer).

We conjecture that the list transformer is a quotient of the *binary tree* transformer, which adds $\hat{B}$-steps for the functor $B(X) \mathrel{\hat{=}} 1 + X \times X$ (see Example 4.8). A more precise statement requires the equational systems of [11].

Example 4.10. We give four (strong) monad transformers on **Set**, which show that the implications in Proposition 4.2 cannot be reversed. When convenient, we use the fact that every endofunctor/monad on **Set** is strong (see Section 3.1).

1. The transformer (T, in) 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 3.8)

$$NX \hat{=} (MR)^{((MR)^X)}$$
$$\mathsf{ret}^N_X(x) \hat{=} \lambda k : (MR)^X.\, k\, x$$
$$\mathsf{bind}^N_{X,Y}(c,\ f) \hat{=} \lambda k : (MR)^Y.\, c\,(\lambda x : X.\, f\, x\, k)$$

and in maps $\hat{M}$ to the morphism $\tau : \hat{M} \longrightarrow T\hat{M}$ given by

$$\tau_X(c : MX) \hat{=} \lambda k : (MR)^X.\, \mathsf{bind}^M_{X,R}(c,\ k)$$

This transformer **is not covariant**, because M is used also 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 $c_1; c_2 \hat{=} \mathsf{bind}^M_{X,X}(c_1,\ \lambda x : X.\, c_2)$.

   The transformer (T, in) **making computations idempotent** is defined as follows, T maps a strong monad $\hat{M}$ to the smallest quotient monad (see Example 2.13) generated by the family of relations

$$R_X \hat{=} \{(c, c; c) \mid c \in MX\}$$

and $\mathsf{in}_{\hat{M}}$ is the epimorphism from $\hat{M}$ to the quotient monad.

   This transformer is covariant, because $\tau_X(c; c) = \tau_X(c); \tau_X(c) : NX$ for any strong monad morphism $\tau : \hat{M} \longrightarrow \hat{N}$ and $c : MX$, but it **is not functorial**. In fact, there are two monads $\hat{M}$ and $\hat{N}$ of complexity (see Example 3.11) with the same underlying endofunctor $F(-) \hat{=} - \times \mathsf{bool}$, with bool the set of booleans, such that $T\hat{M} = \hat{M}$ and $T\hat{N} = \hat{\mathsf{Id}}$:
   - $\hat{M}$ is the strong monad induced by the monoid $(\mathsf{bool}, \mathsf{false}, \mathsf{or})$ in **Set**. Since this monoid is idempotent, all computations in MX are already idempotent, therefore $T\hat{M} = \hat{M}$.
   - $\hat{N}$ is the strong monad induced by the monoid $(\mathsf{bool}, \mathsf{false}, \mathsf{xor})$ in **Set**. Since $\mathsf{xor}(\mathsf{true}, \mathsf{true}) = \mathsf{false}$, the quotient monad $T\hat{N}$ must identify (x, false) and (x, true) for any $x : X$ (and this suffices to make all computations idempotent).

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

$$NX \hat{=} M(X + E)$$
$$\mathsf{ret}^N_X(x) \hat{=} \mathsf{ret}^M_{X+E}(\mathsf{inl}\, x)$$
$$\mathsf{throw}_X(e : E) \hat{=} \mathsf{ret}^M_{X+E}(\mathsf{inr}\, e)$$
$$\mathsf{bind}^N_{X,Y}(c,\ f) \hat{=} \mathsf{bind}^M_{X+E,Y+E}(c,\ [f, \mathsf{throw}_X])$$

and in maps $\hat{M}$ to the morphism $\tau : \hat{M} \longrightarrow T\hat{M}$ given by

$$\tau_X(c : MX) \hat{=} \mathsf{bind}^M_{X,X+E}(c,\ \mathsf{ret}^N_X)$$

This transformer is functorial (since it is the instance of Example 4.8 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

$$\phi_{G,F} : G(F(- + E) + E) \xrightarrow{\bullet} G(F(- + E)).$$

However, this is impossible, when $E = 1$, $GX = X$ and $FX = 0$.

4. The **identity transformer**, which maps $\hat{M}$ to itself, is monoidal. □

5. Transformers and liftings

Theorem 3.4 gives a unique way to lift algebraic operations along any monoid map. Therefore, given a basic transformer (T, in) and a monoid $\hat{M}$, every algebraic operation $A \otimes M \xrightarrow{\mathsf{op}} M$ for $\hat{M}$ can be lifted along $\mathsf{in}_{\hat{M}}$. In this section, we exploit the structure of monoidal and functorial transformers to provide liftings for more general classes of operations, including first-order operations.

Going back to Fig. 1, when one moves from top to bottom the operations become more general, but the lifting theorems need additional assumptions on the transformers or the monoidal category $\hat{\mathcal{E}}$.

Remark 5.1. For covariant transformers we have no lifting result which improves over Theorem 3.4. However, for specific transformers, one may find liftings which are *ad hoc* in the transformer, but *uniform* in the operations (e.g. for the list transformer there is a simple way to lift any first-order operation). In general one should first try to exploit general lifting results, only when these results are not applicable, one should resort to more ad hoc methods. □

Theorem 5.2 (*Monoidal Lifting*)**.** *If $(\hat{T}, \mathsf{in})$ is a monoidal transformer, with $\hat{T} = (T, \phi_{\mathsf{I}}, \phi)$, and $\mathsf{op} : A \otimes M \longrightarrow M$ is a first-order operation for $\hat{M}$, then there is a lifting of op along $\mathsf{in}_{\hat{M}}$ given by*

$$\overline{\mathsf{op}} \mathrel{\hat{=}} A \otimes TM \xrightarrow{\mathsf{in}_A \otimes \mathsf{id}} TA \otimes TM \xrightarrow{\phi} T(A \otimes M) \xrightarrow{T(\mathsf{op})} TM \tag{5.1}$$

More generally, if $H(-) = (A \otimes U(-)) \otimes F$, with $A, F \in \mathcal{E}$, and $\mathsf{op} : H\hat{M} \longrightarrow M$ is an H-operation for $\hat{M}$, then there is a lifting $\overline{\mathsf{op}}$ of op along $\mathsf{in}_{\hat{M}}$ given by

$$\begin{tikzcd}
(TA \otimes TM) \otimes F \arrow[r, "\phi \otimes \mathsf{in}_F"] & T(A \otimes M) \otimes TF \arrow[r, "\phi"] & T((A \otimes M) \otimes F) \arrow[d, "T(\mathsf{op})"] \\
(A \otimes TM) \otimes F \arrow[u, "(\mathsf{in}_A \otimes \mathsf{id}) \otimes \mathsf{id}"] \arrow[rr, dashed, "\overline{\mathsf{op}}"'] & & TM
\end{tikzcd} \tag{5.2}$$

Proof. The first-order case reduces to the more general case when $F = \mathsf{I}$. We need to show that diagram (3.2) commutes, i.e. $\overline{\mathsf{op}} \circ ((\mathsf{id} \otimes \mathsf{in}_M) \otimes \mathsf{id}) = \mathsf{in}_M \circ \mathsf{op}$. We expand the definition of $\overline{\mathsf{op}}$ and prove that the following diagram commutes

$$\begin{tikzcd}
(TA \otimes TM) \otimes F \arrow[r, "\phi \otimes \mathsf{in}_F"] \arrow[dr, "\phi \otimes id"] & T(A \otimes M) \otimes TF \arrow[r, "\phi"] \arrow[dr, phantom, "(1)"] & T((A \otimes M) \otimes F) \arrow[d, "T(\mathsf{op})"] \\
(A \otimes TM) \otimes F \arrow[u, "(\mathsf{in}_A \otimes \mathsf{id}) \otimes \mathsf{id}"] \arrow[r, phantom, "(1)"] & T(A \otimes M) \otimes F \arrow[u, "\mathsf{id} \otimes \mathsf{in}_F"] \arrow[dr, phantom, "(2)"] & TM \\
 & (A \otimes M) \otimes F \arrow[ul, "(id \otimes \mathsf{in}_M) \otimes id"] \arrow[u, "\mathsf{in}_{A \otimes M} \otimes \mathsf{id}"] \arrow[uur, "\mathsf{in}_{(A \otimes M) \otimes F}"'] \arrow[r, "\mathsf{op}"] & M \arrow[u, "\mathsf{in}_M"']
\end{tikzcd}$$

1. because in is a monoidal natural transformation
2. because in is a natural transformation. □

5.1. Functorial lifting

We now focus on functorial transformers. Before proving the main result (Theorem 5.5), we need to establish two lemmas.

Lemma 5.3 (*Derived Lifting*)**.** *Given a functorial transformer (T, in), two H-operations $\mathsf{op}_2 : H\hat{N} \longrightarrow N$ and $\overline{\mathsf{op}}_2 : H(T\hat{N}) \longrightarrow TN$ with $\overline{\mathsf{op}}_2$ a lifting of op_2 along $\mathsf{in}_{\hat{N}}$, a monoid map $t : \hat{M} \longrightarrow \hat{N}$ and a map $f : N \longrightarrow M$, let*

$$\mathsf{op}_1 \mathrel{\hat{=}} H\hat{M} \xrightarrow{Ht} H\hat{N} \xrightarrow{\mathsf{op}_2} N \xrightarrow{f} M \tag{5.3}$$

$$\overline{\mathsf{op}}_1 \mathrel{\hat{=}} H(T\hat{M}) \xrightarrow{H(Tt)} H(T\hat{N}) \xrightarrow{\overline{\mathsf{op}}_2} TN \xrightarrow{Tf} TM \tag{5.4}$$

then $\overline{\mathsf{op}}_1$ is a lifting of op_1 along $\mathsf{in}_{\hat{M}}$.

Proof. The claim amounts to the outer square of the commuting diagram

$$\begin{tikzcd}
H(T\hat{M}) \arrow[rrr, "\overline{\mathsf{op}}_1"] \arrow[dr, "H(Tt)"] & & & TM \\
 & H(T\hat{N}) \arrow[r, "\overline{\mathsf{op}}_2"] \arrow[u, phantom, "(1)"] & TN \arrow[ur, "Tf"] & \\
 & H\hat{N} \arrow[u, "H(\mathsf{in}_{\hat{N}})"] \arrow[r, "\mathsf{op}_2"'] \arrow[l, phantom, "(2)"] \arrow[r, phantom, "(3)", shift left=4ex] & N \arrow[u, "\mathsf{in}_N"'] \arrow[dr, "f"] \arrow[r, phantom, "(2)"] & \\
H\hat{M} \arrow[uuu, "H(\mathsf{in}_{\hat{M}})"] \arrow[ur, "Ht"] \arrow[rrr, "\mathsf{op}_1"'] \arrow[rrr, phantom, "(1)", shift left=4ex] & & & M \arrow[uuu, "\mathsf{in}_M"']
\end{tikzcd}$$

1. by definition of op_1 and $\overline{\mathsf{op}}_1$
2. because in is a natural transformation
3. because, by assumption, $\overline{\mathsf{op}}_2$ is a lifting of op_2 along $\mathsf{in}_{\hat{N}}$. $\square$

Consider Lemma 5.3 when $H(-) = A \otimes U(-)$ and $\mathsf{op}_2 : A \otimes N \longrightarrow N$ is algebraic for $\hat{N}$, then one can take as $\overline{\mathsf{op}}_2$ the algebraic lifting of op_2 along $\mathsf{in}_{\hat{N}}$ (see Theorem 3.4). When $\hat{\mathcal{E}}$ has exponentials, we show that every $\mathsf{op}_1 : A \otimes M \longrightarrow M$ can be expressed (as in Lemma 5.3) using an algebraic op_2, and thus op_1 has a lifting along $\mathsf{in}_{\hat{M}}$.

Lemma 5.4 (*Additional Properties of* $\mathsf{K}M$)**.** *If $\hat{\mathcal{E}}$ has exponentials, $\hat{M}$ is a monoid and $\mathsf{op} : A \otimes M \longrightarrow M$ is a first-order operation for $\hat{M}$, let*

$$\mathsf{from}_{\hat{M}}(f : M^M) : M \mathrel{\hat{=}} f\ e \tag{5.5}$$

$$\widetilde{\mathsf{op}}(s : A, f : M^M) : M^M \mathrel{\hat{=}} \lambda x : M.\mathsf{op}(s, f\ x) \tag{5.6}$$

then the following claims hold (where $\mathsf{K}M$ and $\mathsf{to}_{\hat{M}}$ are given in Example 2.11)

(a) $M \xrightarrow{\mathsf{to}_{\hat{M}}} M^M \xrightarrow{\mathsf{from}_{\hat{M}}} M$ *is the identity on* M

(b) $\widetilde{\mathsf{op}} : A \otimes M^M \longrightarrow M^M$ *is algebraic for* $\mathsf{K}M$ *and* $\mathsf{op} = A \otimes M \xrightarrow{\mathsf{to}_{\hat{M}}} A \otimes M^M \xrightarrow{\widetilde{\mathsf{op}}} M^M \xrightarrow{\mathsf{from}_{\hat{M}}} M$

(c) op *algebraic for* $\hat{M}$ *implies* $\widetilde{\mathsf{op}}$ *is the algebraic lifting of* op *along* $\mathsf{to}_{\hat{M}}$.

Proof. Let Eq be the set of equations saying that $\hat{M}$ is a monoid (Definition 2.7) and $Eqop$ be Eq plus (3.1) saying that op is algebraic for $\hat{M}$, then the claims amount to the equations (we drop the type M of bound variables)

(a) $x : M \vdash_{Eq} \mathsf{from}_{\hat{M}}(\mathsf{to}_{\hat{M}}(x)) = x : M$

$$\begin{array}{ll}
\mathsf{from}_{\hat{M}}(\mathsf{to}_{\hat{M}}(x)) & \text{by definition} \\
(\lambda x'.x \cdot x')\ e & \text{by reduction } (\beta.\to) \\
x \cdot e & \text{by (2.12) in } Eq \\
x &
\end{array}$$

(b) $s : A, x'_1, x'_2 : M^M \vdash_{Eq} \mathsf{c}_M(\widetilde{\mathsf{op}}(s, x'_1), x'_2) = \widetilde{\mathsf{op}}(s, \mathsf{c}_M(x'_1, x'_2)) : M^M$

$$\begin{array}{ll}
\mathsf{c}_M(\widetilde{\mathsf{op}}(s, x'_1), x'_2) & \text{by definition} \\
\lambda x.(\lambda x.\mathsf{op}(s, x'_1\ x))\ (x'_2\ x) & \text{by reduction } (\beta.\to) \\
\lambda x.\mathsf{op}(s,\ (\lambda x.x'_1\ (x'_2\ x))\ x) & \text{by definition} \\
\widetilde{\mathsf{op}}(s, \mathsf{c}_M(x'_1, x'_2)) &
\end{array}$$

$s : A, x : M \vdash_{Eq} \mathsf{from}_{\hat{M}}(\widetilde{\mathsf{op}}(s, \mathsf{to}_{\hat{M}}(x))) = \mathsf{op}(s, x) : M$

$$\begin{array}{ll}
\mathsf{from}_{\hat{M}}(\overline{\mathsf{op}}(s, \mathsf{to}_{\hat{M}}(x))) & \text{by definition} \\
(\lambda x'.\mathsf{op}(s,\ (\lambda x'.x \cdot x')\ x'))\ e & \text{by reduction } (\beta.\to) \\
\mathsf{op}(s, x \cdot e) & \text{by (2.12) in } Eq \\
\mathsf{op}(s, x) &
\end{array}$$

(c) $s : A, x : M \vdash_{Eqop} \widetilde{\mathsf{op}}(s, \mathsf{to}_{\hat{M}}(x)) = \mathsf{to}_{\hat{M}}(\mathsf{op}(s, x)) : M^M$

$$\begin{array}{ll}
\widetilde{\mathsf{op}}(s, \mathsf{to}_{\hat{M}}(x)) & \text{by definition} \\
\lambda x'.\mathsf{op}(s,\ (\lambda x'.x \cdot x')\ x')) & \text{by reduction } (\beta.\to) \\
\lambda x'.\mathsf{op}(s, x \cdot x') & \text{by (3.1) in } Eqop \\
\lambda x'.\mathsf{op}(s, x) \cdot x' & \text{by definition} \\
\mathsf{to}_{\hat{M}}(\mathsf{op}(s, x)) &
\end{array}$$

$\square$

Theorem 5.5 (*Functorial Lifting*)**.** *If (T, in) is a functorial transformer, and $\mathsf{op} : A \otimes M \longrightarrow M$ is a first-order operation for $\hat{M}$, then there is a lifting $\overline{\mathsf{op}}$ of op along $\mathsf{in}_{\hat{M}}$ given by*

$$\overline{\mathsf{op}} \mathrel{\hat{=}} A \otimes TM \xrightarrow{\mathsf{id} \otimes T(\mathsf{to}_{\hat{M}})} A \otimes T(M^M) \xrightarrow{\widetilde{\mathsf{op}}^\sharp} T(M^M) \xrightarrow{T(\mathsf{from}_{\hat{M}})} TM \tag{5.7}$$

where $\widetilde{\mathsf{op}}$ is defined in (5.6) and $\widetilde{\mathsf{op}}^\sharp$ is the unique algebraic lifting of $\widetilde{\mathsf{op}}$ along $\mathsf{in}_{(\mathsf{K}M)}$ given by Theorem 3.4.

Proof. The lifting $\overline{\mathsf{op}}$ is the $\overline{\mathsf{op}}_1$ given in Lemma 5.3 when one takes $\hat{N} = \mathsf{K}M$, $\mathsf{op}_2 = A \otimes N \xrightarrow{\widetilde{\mathsf{op}}} N$, thus op_2 is algebraic for $\hat{N}$ (by Lemma 5.4), $\overline{\mathsf{op}}_2$ the unique algebraic lifting $A \otimes (TN) \xrightarrow{\mathsf{op}_2^\sharp} TN$ of op_2 along $\mathsf{in}_{\hat{N}}$, $t = \mathsf{to}_{\hat{M}}$, $f = \mathsf{from}_{\hat{M}}$, and thus $\mathsf{op}_1 = \mathsf{op}$ (again by Lemma 5.4). $\square$

5.2. Coincidence of liftings

For some pair operation–transformer two (or more) of the lifting theorems summarized in Fig. 1 are applicable. For instance, if op is an algebraic operation for $\hat{M}$ and $(\hat{T}, \mathsf{in})$ is a monoidal transformer, then one can apply both the algebraic lifting (Theorem 3.4) and the monoidal lifting (Theorem 5.2). We prove that when two lifting theorems are applicable, they yield the same result.

Theorem 5.6 (*Algebraic/Monoidal*)**.** *If $(\hat{T}, \mathsf{in})$ is a monoidal transformer and $\mathsf{op} : A \otimes M \longrightarrow M$ is algebraic for $\hat{M}$, then the monoidal lifting (Theorem 5.2) and the algebraic lifting (Theorem 3.4) of op along $\mathsf{in}_{\hat{M}}$ coincide.*

Proof. Eq. (3.3), saying that op is algebraic for $\hat{M} = (M, e, m)$, amounts to $\mathsf{op} = m \circ (\mathsf{op}' \otimes \mathsf{id})$, where $\mathsf{op}'(s : A) : M \mathrel{\hat{=}} \mathsf{op}(s, e)$. The coincidence follows by the commuting diagram below, where the top path from $A \otimes TM$ to TM is the monoidal lifting of op, and the bottom path is the algebraic lifting of op along $\mathsf{in}_{\hat{M}} : \hat{M} \longrightarrow T\hat{M}$ (the multiplication of $T\hat{M}$ is $(Tm) \circ \phi$, see Theorem 2.6)

$$\begin{tikzcd}
A \otimes TM \arrow[r, "\mathsf{in}_A \otimes \mathsf{id}"] \arrow[d, "\mathsf{op}' \otimes \mathsf{id}"'] \arrow[dr, phantom, "(1)"] & TA \otimes TM \arrow[r, "\phi"] \arrow[d, "T(\mathsf{op}') \otimes T(\mathsf{id})"] \arrow[dr, phantom, "(2)"] & T(A \otimes M) \arrow[r, "T(\mathsf{op})"] \arrow[d, "T(\mathsf{op}' \otimes \mathsf{id})"] \arrow[dr, phantom, "(3)"] & TM \arrow[d, equal] \\
M \otimes TM \arrow[r, "\mathsf{in}_M \otimes \mathsf{id}"'] & TM \otimes TM \arrow[r, "\phi"'] & T(M \otimes M) \arrow[r, "Tm"'] & TM
\end{tikzcd}$$

1. because in is a natural transformation
2. because ϕ is a natural transformation
3. because $\mathsf{op} = m \circ (\mathsf{op}' \otimes \mathsf{id})$ and functoriality of T. □

Theorem 5.7 (*Algebraic/Functorial*)**.** *If (T, in) is a functorial transformer on a monoidal category with exponentials and $\mathsf{op} : A \otimes M \longrightarrow M$ is algebraic for $\hat{M}$, then the functorial lifting (Theorem 5.5) and the algebraic lifting (Theorem 3.4) of op along $\mathsf{in}_{\hat{M}}$ coincide.*

Proof. Since op is algebraic for $\hat{M}$, we can define the following algebraic liftings

- $\mathsf{op}^\sharp : A \otimes TM \longrightarrow TM$ the algebraic lifting of op along $\mathsf{in}_{\hat{M}}$
- $\widetilde{\mathsf{op}} : A \otimes M^M \longrightarrow M^M$ the algebraic lifting of op along $\mathsf{to}_{\hat{M}}$, which is given by (5.6) of Lemma 5.4
- $\widetilde{\mathsf{op}}^\sharp : A \otimes T(M^M) \longrightarrow T(M^M)$ the algebraic lifting of $\widetilde{\mathsf{op}}$ along $\mathsf{in}_{(KM)}$.

The coincidence follows by the commuting diagram below, where the bottom path from $A \otimes TM$ to TM is the functorial lifting of op given by Theorem 5.5

$$\begin{tikzcd}
A \otimes TM \arrow[r, "\mathsf{op}^\sharp"] \arrow[d, "\mathsf{id} \otimes T(\mathsf{to}_{\hat{M}})"'] \arrow[dr, phantom, "(1)"] & TM \arrow[d, "T(\mathsf{to}_{\hat{M}})"] \arrow[dr, equal] \arrow[dr, phantom, "(2)"] & \\
A \otimes T(M^M) \arrow[r, "\widetilde{\mathsf{op}}^\sharp"'] & T(M^M) \arrow[r, "T(\mathsf{from}_{\hat{M}})"'] & TM
\end{tikzcd}$$

1. because, $\widetilde{\mathsf{op}}^\sharp$ is the unique algebraic of $\mathsf{op}^\sharp$ along $T(\mathsf{to}_{\hat{M}})$, in fact
   - $\mathsf{op}^\sharp$ is the unique algebraic lifting of op along $\mathsf{in}_{\hat{M}}$
   - $\widetilde{\mathsf{op}}^\sharp$ is the unique algebraic lifting of op along $\mathsf{in}_{(KM)} \circ \mathsf{to}_{\hat{M}}$
   - $T(\mathsf{to}_{\hat{M}}) \circ \mathsf{in}_{\hat{M}} = \mathsf{in}_{(KM)} \circ \mathsf{to}_{\hat{M}}$ by naturality of in
2. by Lemma 5.4(a) and functoriality of T. □

Theorem 5.8 (*Functorial/Monoidal*)**.** *If $(\hat{T}, \mathsf{in})$ is a monoidal transformer on a monoidal category with exponentials and $\mathsf{op} : A \otimes M \longrightarrow M$, then the functorial lifting (Theorem 5.5) and the monoidal lifting (Theorem 5.2) of op along $\mathsf{in}_{\hat{M}}$ coincide.*

Proof. The functorial lifting of op is given by $A \otimes TM \xrightarrow{\mathsf{id} \otimes T(\mathsf{to}_{\hat{M}})} A \otimes T(M^M) \xrightarrow{\widetilde{\mathsf{op}}^\sharp} T(M^M) \xrightarrow{T(\mathsf{from}_{\hat{M}})} TM$, where $\widetilde{\mathsf{op}}^\sharp$ is the algebraic lifting of $\widetilde{\mathsf{op}}$ along $\mathsf{in}_{(KM)}$ (see Theorem 5.5), or equivalently (by Theorem 5.6) $\widetilde{\mathsf{op}}^\sharp$ is the monoidal lifting of $\widetilde{\mathsf{op}}$ along $\mathsf{in}_{(KM)}$, i.e. $\widetilde{\mathsf{op}}^\sharp = A \otimes T(M^M) \xrightarrow{\mathsf{in}_A \otimes \mathsf{id}} TA \otimes T(M^M) \xrightarrow{\phi} T(A \otimes M^M) \xrightarrow{T(\widetilde{\mathsf{op}})} T(M^M)$. The coincidence follows by the commuting diagram below, where the top path from $A \otimes TM$ to TM is the monoidal lifting of op, and the bottom path is the functorial lifting of op

$$\begin{tikzcd}
A \otimes TM \arrow[r, "\mathsf{in}_A \otimes \mathsf{id}"] \arrow[d, "\mathsf{id} \otimes T(\mathsf{to}_{\hat{M}})"'] & TA \otimes TM \arrow[r, "\phi"] \arrow[d, "T(\mathsf{id}) \otimes T(\mathsf{to}_{\hat{M}})"'] \arrow[dr, phantom, "(1)"] & T(A \otimes M) \arrow[r, "T(\mathsf{op})"] \arrow[d, "T(\mathsf{id} \otimes \mathsf{to}_{\hat{M}})"'] \arrow[dr, phantom, "(2)"] & TM \\
A \otimes T(M^M) \arrow[r, "\mathsf{in}_A \otimes \mathsf{id}"'] & TA \otimes T(M^M) \arrow[r, "\phi"'] & T(A \otimes M^M) \arrow[r, "T(\widetilde{\mathsf{op}})"'] & T(M^M) \arrow[u, "T(\mathsf{from}_{\hat{M}})"']
\end{tikzcd}$$

1. because ϕ is a natural transformation
2. by Lemma 5.4(b) and functoriality of T. □

6. Conclusions

2-categories versus monoidal categories. Category-theoretic notions, such as monads and adjunctions, can be recast in the setting of a 2-category [21], in fact for monads 2-categories with one object suffice. A 2-category $\mathcal{C}$ with one object corresponds to a strict monoidal category $\hat{\mathcal{E}}$, and the correspondence induces a bijection between monads in $\mathcal{C}$ and monoids in $\hat{\mathcal{E}}$. Moreover, we can drop the strictness assumption on $\hat{\mathcal{E}}$ (and replace 2-categories with bicategories [6]). Therefore, the move from monads to monoids is a natural generalization. What is not obvious, is the possibility of addressing the lifting problem (for monad transformers) at this level of generality, indeed this is the main novelty w.r.t. [17].

Relation with the companion paper [17]. The main results in the companion paper are *instances* of the algebraic and functorial lifting (Theorems 3.4 and 5.5) for the monoidal category $\hat{\mathcal{E}}_{F\omega}$ of endofunctors expressible in $F\omega$ (see Example 2.19). Theorem 5.5 is not applicable to $\hat{\mathcal{E}}_{F\omega}$, because it does not have exponentials (in addition some claims in [17] are wrong). However, this problem is overcome by replacing $\hat{\mathcal{E}}_{F\omega}$ with $\mathsf{Endo}(\mathcal{P}_{F\omega})_r$ of Example 2.21. Finally, the companion paper works with *expressible* monad transformers, a proper subset of the monoid transformers on $\hat{\mathcal{E}}_{F\omega}$, which are more amenable to *implementation* in a programming language.

Generalizations of algebraic theories. [11] has proposed a notion of (iterated) equational system on a category $\mathcal{C}$, which provides a significant generalization of algebraic theories and constructions of free algebras. The definition of *functorial term* of arity A given in [11] is closely related to the definition of *algebraic operation* of arity A 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 $\mathcal{C}^{\hat{M}}$ of Eilenberg–Moore algebras for the monad $\hat{M}$, then there is a bijective correspondence between natural transformations $\mathsf{op} : A \xrightarrow{\bullet} M$, i.e. algebraic operations of arity A for $\hat{M}$, and functorial terms T of arity A, i.e. functors $T : \mathcal{C}^{\hat{M}} \longrightarrow A\text{-}\mathbf{Alg}$ such that $\mathcal{C}^{\hat{M}} \xrightarrow{U} \mathcal{C} = \mathcal{C}^{\hat{M}} \xrightarrow{T} A\text{-}\mathbf{Alg} \xrightarrow{U} \mathcal{C}$.

$$T(MX \xrightarrow{\alpha} X) = AX \xrightarrow{\mathsf{op}_X} MX \xrightarrow{\alpha} X$$
$$\mathsf{op}_X = AX \xrightarrow{A\eta_X} A(MX) \xrightarrow{T(\mu_X)} MX$$

This correspondence suggests a reinterpretation (and generalization) of the notions introduced in [11]:

Equational Systems [11]	Monoidal Category $\hat{\mathcal{E}}$
iterated equational system (IES)	monoid $\hat{M} \in \mathsf{Mon}(\hat{\mathcal{E}})$
functorial signature F (IES with $n = 0$)	object $F \in \mathcal{E}$
category F-**Alg** of F-algebras	free monoid F^* over F
functorial term T of arity D	map $\mathsf{op} : D \longrightarrow U(\hat{M})$
adding an equation to an IES IES $\vdash T_1 = T_2 : D$	taking a quotient of $\hat{M}$ $D \underset{\mathsf{op}_2}{\overset{\mathsf{op}_1}{\rightrightarrows}} \hat{M} \dashrightarrow \hat{N}$

Future work. A topic of future work is to investigate the use of free constructions for equational systems to define strong monad transformers that add to a pre-existing monad new operations satisfying certain equations (Example 4.9 should be an instance of this). Another line of research (mentioned in the Introduction) is the use of monoid transformers for an incremental approach for *arrows* [14] (viewed as monoids [13]) or other generalizations of monads proposed in the literature.

Acknowledgements

We would like to thank the anonymous referees for their constructive criticisms and suggestions. The second author is partially supported by Italian PRIN 2008 "Metodi Costruttivi in Topologia, Algebra e Fondamenti dell'Informatica".

References

[1] M. Abbott, T. Altenkirch, N. Ghani, Categories of containers, in: A.D. Gordon (Ed.), FoSSaCS, in: Lect. Notes in Comput. Sci., vol. 2620, Springer, 2003, pp. 23–38.
[2] R. Atkey, What is a categorical model of arrows? in: Mathematically Structured Functional Programming, Electron. Notes Theor. Comput. Sci, 2008 (in press).
[3] H. Barendregt, Lambda calculi with types, in: S. Abramsky, D.M. Gabbay, T.S.E. Maibaum (Eds.), Handbook of Logic in Computer Science, Oxford University Press, 1992, pp. 117–309.
[4] M. Barr, C. Wells, Toposes, Triples and Theories, in: Grundlehren der mathematischen Wissenschaften, vol. 278, Springer-Verlag, New York, 1985.
[5] M. Barr, C. Wells, Category Theory for Computer Science, Prentice Hall, 1995.
[6] J. Bénabou, Introduction to bicategories, in: Reports of the Midwest Category Seminar, in: Springer Lecture Notes in Mathematics, vol. 47, Springer-Verlag, 1967, pp. 1–77.
[7] N. Benton, J. Hughes, E. Moggi, Monads and effects, in: International Summer School On Applied Semantics APPSEM2000, Springer-Verlag, 2000, pp. 42–122.
[8] F. Borceux, Handbook of Categorical Algebra, Cambridge University Press, 1994.
[9] K.B. Bruce, A.R. Meyer, J.C. Mitchell, The semantics of the second-order lambda calculus, Inf. Comput. 85 (1990) 76–134.
[10] G. Castagna (Ed.), Programming Languages and Systems, 18th European Symposium on Programming, ESOP 2009, in: Lect. Notes in Comput. Sci., vol. 5502, Springer, 2009.
[11] M. Fiore, C.-K. Hur, On the construction of free algebras for equational systems, Theoret. Comput. Sci. 410 (18) (2009) 1704–1729.
[12] N. Ghani, Eta-expansions in F-omega, in: Proceedings of CSL'96, in: Lecture Notes in Computer Science, vol. 1258, Springer-Verlag, 1996, pp. 182–197.
[13] C. Heunen, B. Jacobs, Arrows, like monads, are monoids, Electron. Notes Theor. Comput. Sci. 158 (2006) 219–236.
[14] J. Hughes, Generalising monads to arrows, Sci. Comput. Programming 37 (1–3) (2000) 67–111.
[15] M. Hyland, G.D. Plotkin, J. Power, Combining effects: Sum and tensor, Theor. Comput. Sci. 357 (1–3) (2006) 70–99.
[16] M. Jaskelioff, Monatron: an extensible monad transformer library, in: Implementation and Application of Functional Languages, IFL08, 2008 (in press).
[17] M. Jaskelioff, Modular monad transformers, in: G. Castagna [10], pp. 64–79.
[18] S.L. Peyton Jones, P. Wadler, Imperative functional programming, in: 20th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 1993, pp. 71–84.
[19] G.M. Kelly, A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on, Bull. Aust. Math. Soc. 22 (01) (1980) 1–83.
[20] G.M. Kelly, J. Power, Adjunctions whose counits are coequalizers and presentations of finitary monads, J. Pure Appl. Algebra 89 (1–2) (1993) 163–179.
[21] G.M. Kelly, R.H. Street, Review of the elements of 2-categories, in: A. Dold, B. Eckmann (Eds.), Category Seminar, in: Lect. Notes in Math., vol. 420, Springer, 1974.
[22] J. Lambek, From lambda calculus to cartesian closed categories, in: To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Academic Press, 1980, pp. 375–402.
[23] J. Lambek, P.J. Scott, Introduction to Higher Order Categorical Logic, Cambridge University Press, 1986.
[24] S. Liang, P. Hudak, Modular denotational semantics for compiler construction, in: H.R. Nielson (Ed.), ESOP, in: Lect. Notes in Comput. Sci., vol. 1058, Springer, 1996, pp. 219–234.
[25] S. Liang, P. Hudak, M. Jones, Monad transformers and modular interpreters, in: 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 1995, pp. 333–343.
[26] J.R. Longley, Realizability toposes and language semantics, Ph.D. Thesis, University of Edinburgh, 1994.
[27] C. Lüth, N. Ghani, Composing monads using coproducts, in: 7th ACM SIGPLAN International Conference on Functional Programming, 2002, pp. 133–144.
[28] S. Mac Lane, Categories for the Working Mathematician, second edition, in: Graduate Texts in Mathematics, vol. 5, Springer-Verlag, 1998.
[29] E.G. Manes, Algebraic Theories, Springer-Verlag, 1976.
[30] E.G. Manes, Implementing collection classes with monads, Math. Structures Comput. Sci. 8 (3) (1998) 231–276.
[31] E. Moggi, Computational lambda-calculus and monads, in: 4th Annual Symposium on Logic in Computer Science, IEEE Computer Society, 1989, pp. 14–23.
[32] E. Moggi, Notions of computation and monads, Inf. Comput. 93 (1) (1991) 55–92.
[33] E. Moggi, Metalanguages and applications, in: Semantics and Logics of Computation, Publications of the Newton Institute, CUP, 1997.
[34] Andrew M. Pitts, Tripos theory in retrospect, Math. Structures Comput. Sci. 12 (3) (2002) 265–279.
[35] G.D. Plotkin, J. Power, Semantics for algebraic operations, Electron. Notes Theor. Comput. Sci. 45 (2001).
[36] G.D. Plotkin, J. Power, Notions of computation determine monads, in: M. Nielsen, U. Engberg (Eds.), FoSSaCS, in: Lect. Notes in Comput. Sci., vol. 2303, Springer, 2002, pp. 342–356.
[37] G.D. Plotkin, M. Pretnar, Handlers of algebraic effects. in: G. Castagna [10], pp. 80–94.
[38] J. Polakow, F. Pfenning, Natural deduction for intuitionistic non-communicative linear logic, in: J.-Y. Girard (Ed.), TLCA, in: Lect. Notes in Comput. Sci., vol. 1581, Springer, 1999, pp. 295–309.
[39] J. Power, E. Robinson, Premonoidal categories and notions of computation, Math. Structures Comput. Sci. 7 (5) (1997) 453–468.
[40] D.S. Scott, Relating theories of the λ-calculus, in: R. Hindley, J. Seldin (Eds.), To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, Academic Press, 1980.
[41] P. Wadler, Comprehending monads, Math. Structures Comput. Sci. 2 (4) (1992) 461–493.
[42] P. Wadler, The essence of functional programming, in: 9th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 1992, pp. 1–14.