Modular Models of Monoids with Operations by Lifting Functors along Fibrations

ZHIXUAN YANG
Imperial College London, United Kingdom
e-mail: s.yang20@imperial.ac.uk

NICOLAS WU
Imperial College London, United Kingdom
e-mail: n.wu@imperial.ac.uk

Abstract

Inspired by Plotkin and Power's algebraic treatment of computational effects and the principle of *notions of computations as monoids*, we propose a categorical framework for *equational theories* and *models* of monoids equipped with operations. This framework generalises Plotkin and Power's algebraic treatment of effectful operations taking or returning *values* as input or output to operations that may take or return *computations* as input or output. Additionally, to give semantic models of computational effects in a modular way, we introduce a formal theory of modular constructions of algebraic structures based on the framework of lifting functors along fibrations.

1 Introduction

In his seminal work, Moggi (1989, 1991, 1989) pioneered modelling *notions of computation*, which are now more commonly referred to as *computational effects*, using *monads* and *monad transformers*. The understanding of this approach is later deepened by Plotkin and Power (2002, 2001, 2004), who characterise many monads that model computational effects as arising from *equational theories* of some primitive effectful operations and equational laws characterising their interaction.

Later, variations of monads are proposed to model more forms of computational effects, including *arrows* (Hughes, 2000; Jacobs et al., 2009), *applicative functors* (McBride and Paterson, 2008; Paterson, 2012), *parameterised monads* (Atkey, 2009), *graded monads* (Katsumata, 2014). All these notions can be unified in the framework of *monoids in monoidal categories*, leading up to the principle of *notions of computation as monoids* (Rivas and Jaskelioff, 2017; Pieters et al., 2020).

These ideas from denotational semantics are quickly adopted by the functional programming community as an abstraction for effectful programming (Wadler, 1995; Filinski, 1994). Many refinements have been proposed for making programming with effects more *modular*. A fundamental idea is to *separate syntax from semantics*, so syntactic effectful programs can be written without resorting to a specific semantics, and (possibly many kinds of) semantics

can be given later. Building on this, there are further two kinds of modularity that are both desirable but are sometimes conflated:

Syntactic modularity is where the syntactic specification of two languages Σ_1 and Σ_2 can be combined into a larger language $\Sigma_1 +_{\text{syn}} \Sigma_2$.

Semantic modularity is where semantic models M_1 and M_2 for the sub-languages can be combined into a model $M_1 +_{\text{sem}} M_2$ of the larger language $\Sigma_1 +_{\text{syn}} \Sigma_2$.

In terms of this classification, *free monads* and their variants provide syntactic modularity (Kiselyov and Ishii, 2015; Swierstra, 2008). Typeclasses of monads with operations in Haskell (Gill and Kmett, 2012; Liang et al., 1995) also achieve syntactic modularity.

On the other hand, *monad transformers* (Moggi, 1989; Jaskelioff and Moggi, 2010), *layered monads* (Filinski, 1999), and combining monads by coproducts (Ghani and Uustalu, 2004) are about semantic modularity.

Effect Handlers Notably, *effect handlers*, introduced by Plotkin and Pretnar (2013) and further developed by many people, achieve both kinds of modularity, which explains their quick adoption. In this approach, effectful operations are described by an algebraic theory (sometimes without any equations), whose free-algebra monad is used as the monad for effectful computations. Effect handlers are (not necessarily free) algebras of this theory, so they can 'handle' effectful computations, i.e. the free algebra, using the unique homomorphism out of the free algebra. Algebraic theories and handlers are both composable so syntactic and semantic modularity are both achieved.

However, effectful operations in this framework *necessarily* fall into a kind of effectful operation known as *algebraic operations* (Plotkin and Power, 2003). An algebraic operation on a monad $M : \mathscr{C} \to \mathscr{C}$ is a natural transformation $\alpha : A \circ M \to M$ for an endofunctor $A : \mathscr{C} \to \mathscr{C}$, typically a polynomial functor, such that it is compatible with monad multiplication:

$$\mu^M \cdot (\alpha \circ M) \;=\; \alpha \cdot (A \circ \mu^M) \;:\; A \circ M \circ M \to M.$$

This intuitively says that the operation α commutes with sequential composition.

Not all effectful operations have this property though: for example, exception catching in Haskell $catch :: IO\ a \to (Exc \to IO\ a) \to IO\ a$ does not satisfy

$$(catch\ p\ h) >\!\!>\!= k \;=\; catch\ (p >\!\!>\!= k)\ (h >\!\!=\!> k)$$

since the left-hand side only catches the exception in p while the right-hand side catches the exception in p and k. Such non-algebraic operations *can* be programmed as handlers (Plotkin and Pretnar, 2013), but they do not have the benefit of (syntactic or semantic) modularity, since as handlers they themselves cannot be handled to give alternative semantics (Wu et al., 2014; Yang et al., 2022).

A line of research seeks to lift the expressivity of effect handlers (Yang et al., 2022; Wu et al., 2014; Piróg et al., 2018; Bach Poulsen and van der Rest, 2023) by considering signatures/theories of broader ranges of operations and their models. The leading example is *scoped operations* considered by Piróg et al. (2018), which are operations on monads M of the following form for some $A : \mathscr{C} \to \mathscr{C}$:

$$s : \int^{X \in \mathscr{C}} A(MX) \times (M-)^X \cong A \circ M \circ M \to M, \tag{1.1}$$

where the isomorphism is by the co-Yoneda lemma. Typically, the functor A is $(-)^n$ for some natural number n. The intuition is that s is an operation delimiting n scopes: the coend $\int^X$ is like an existential type $\exists X$, and $A(MX) = (MX)^n$ is the computation inside the scopes, returning some type X, and $(M-)^X$ is the computation after these scopes. For example, the operation of *exception catching* delimits two scopes, one for 'try' and the other for 'catch', so $A = (-)^2 \cong (- \times -)$.

However, existing work in this direction only considers syntactic modularity (with the exception (Wu et al., 2014)). The root of the difficulty with semantic modularity is that only algebraic operations have a canonical lifting along a monad morphism: given a monad morphism $f : M \to N$ and an algebraic operation $\alpha : A \circ M \to M$ on M, there is a unique algebraic operation $\overline{\alpha} : A \circ N \to N$:

$$\overline{\alpha} = (\ A \circ N \xrightarrow{A \circ \eta^M \circ N} A \circ M \circ N \xrightarrow{\alpha \circ N} M \circ N \xrightarrow{f \circ N} N \circ N \xrightarrow{\mu^M} N\ ) \tag{1.2}$$

making $f : M \to N$ an algebra homomorphism from α to $\overline{\alpha}$. In contrast, if the operation takes the form Equation 1.1 or more generally the form $\alpha : \Sigma M \to M$ for a functor $\Sigma : \textsc{Endo}(\mathscr{C}) \to \textsc{Endo}(\mathscr{C})$, we do not have a formula to define a meaningful $\overline{\alpha} : \Sigma N \to N$ from $f : M \to N$ and $\alpha : \Sigma M \to M$.

Overview Motivated by the lack of semantic modularity in existing frameworks, the present paper has two aims/parts: the first one is to develop a unifying account of equational theories of (algebraic and non-algebraic) effectful operations; the second one is to develop a framework of *modular models* that provide semantic modularity for non-algebraic operations. These two parts together achieve both modularities in the style of effect handlers, but for a much wider range of operations.

In the first part (Sections 2 to 6), we start by developing a convenient way of presenting equational theories over monoidal categories by using *equational systems* proposed by Fiore and Hur (2009) and the internal language of monoidal categories, which we call *monoidal algebraic theories*. Then we study *operation families*, which are subcategories of equational theories of *monoids with operations*. Examples include the family of *algebraic operations*, and the family of *scoped operations*, and the family of *variable-binding operations*. Syntactic modularity is achieved by taking colimits in operation families.

In the second part (Sections 7 to 8), we propose a general theory for *modularity in algebraic structures*. The main idea is to define a *modular model* M of an algebraic theory Σ to be a *lifting* of the functor $(- + \Sigma) : \mathscr{T} \to \mathscr{T}$ along the fibration $P : \mathscr{A} \to \mathscr{T}$

$$\begin{array}{ccc} \mathscr{A} & \xrightarrow{M} & \mathscr{A} \\ {\scriptstyle P}\downarrow & & \downarrow{\scriptstyle P} \\ \mathscr{T} & \xrightarrow[-+\Sigma]{} & \mathscr{T} \end{array}$$

where $\mathscr{T}$ is the category of algebraic theories in question and $P : \mathscr{A} \to \mathscr{T}$ is the fibration given by the Grothendieck construction for $(-)\text{-}\textsc{Alg} : \mathscr{T}^{\mathrm{op}} \to \mathrm{CAT}$, so $\mathscr{A}$ is the category contains all models of all theories in $\mathscr{T}$. This idea is then generalised to considering liftings $M : \mathscr{A} \to \mathscr{A}'$ of an arbitrary functor $T : \mathscr{T} \to \mathscr{T}'$ along two possibly different fibrations $P : \mathscr{A} \to \mathscr{T}$ and $P' : \mathscr{A}' \to \mathscr{T}'$. The informal intuition is that the functor T is a theory $T\Gamma$ parameterised by a generic 'future extension' Γ to the theory, and the functor $M : \mathscr{A} \to \mathscr{A}'$

is then a model of T that parameterised by a generic model of Γ, so we call the lifting M a *model transformers* of T. A number of universal constructions and concrete constructions of model transformers are then given.

Example We sketch a small concrete example here to demonstrate the ideas. Let $\mathscr{E}$ be the monoidal category $\langle \textsc{Endo}_f(\textsc{Set}), \circ, \mathrm{Id}\rangle$ of finitary endofunctors on sets. The equational theory Ec of monads M with *exception catching* has two operations besides those of monads:

$$\mathit{throw} : 1 \to M \qquad \text{and} \qquad \mathit{catch} : (\mathrm{Id} \times \mathrm{Id}) \circ M \circ M \cong (M \times M) \circ M \to M,$$

and for now let us ignore equations on these operations. The operation *throw* is for throwing an exception. The operation *catch* is for catching an exception in a scope and handling it, which is an instance of Equation 1.1 with $A = \mathrm{Id} \times \mathrm{Id}$. As we briefly mentioned earlier, the product $M \times M$ in the signature of *catch* is a pair of computations, one for the program p whose exceptions are caught, the other one for the program h handling the exception; the $\circ M$ after $M \times M$ in the signature of *catch* is an *explicit continuation* argument k. Thus $\mathit{catch}\ (\langle p, h\rangle, k)$ is understood as handling the exception in p using h, and then continues as k. The explicit continuation is exactly the trick to workaround the limitation of algebraic operations (we will say more about it in Section 6).

All theories of *monads with some scoped operations* are collected as a category $\textsc{Scp}(\mathscr{E})$, which we call the *monoidal theory family of scoped operations* over $\mathscr{E}$, whose arrows are *translations* of theories. The category $\textsc{Scp}(\mathscr{E})$ has finite coproducts by taking the coproduct of the signatures and equations. Moreover, each theory in $\textsc{Scp}(\mathscr{E})$ has free algebras, in particular initial algebras. For example, the initial algebra of Ec is the initial one among all *monads with throw and catch*. The carrier of the initial algebra $\mu\mathrm{Ec} : \textsc{Set} \to \textsc{Set}$ can be characterised as the initial solution to

$$X \ \cong\ \mathrm{Id} + 1 + (X \times X) \circ X \ \in\ \textsc{Endo}_f(\textsc{Set}).$$

The monad μEc models *syntactic programs* with exception throwing and catching.

A (strict) *modular model* of the theory Ec is a family of functors $M_{\ddot\Sigma} : \ddot\Sigma\text{-}\textsc{Alg} \to (\ddot\Sigma + \mathrm{Ec})\text{-}\textsc{Alg}$, natural in $\ddot\Sigma \in \textsc{Scp}(\mathscr{E})$. Here $(-)$-Alg is the functor $\textsc{Scp}(\mathscr{E}) \to \textsc{CAT}$ sending each theory to the category of its models. One possible modular model of Ec is to send every $\ddot\Sigma$-algebra carried by A to $A \circ (1 + \mathrm{Id})$ equipped with operations in $\ddot\Sigma$ and Ec. In the same way that there may be many handlers of the same algebraic operation, we also have choices over how *catch* and *throw* act on $A \circ (1 + \mathrm{Id})$. Besides the 'standard' semantics (detailed in Example 8.4∗6), we may also have non-standard semantics such as re-trying the program after the handling program is executed, or the semantics that the exceptions thrown by the handling program are recursively handled.

The advantage of a modular model M of Ec over an ordinary model of Ec is that M allows us to interpret syntactic programs $\mu(\ddot\Sigma + \mathrm{Ec})$ involving throwing and catching mixed with *any other operations* $\ddot\Sigma$ in $\textsc{Scp}(\mathscr{E})$. By the initiality of $\mu(\ddot\Sigma + \mathrm{Ec})$, there is a morphism

$$h : \mu(\ddot\Sigma + \mathrm{Ec}) \to M_{\ddot\Sigma}(\mu\ddot\Sigma) \qquad \text{in} \qquad (\ddot\Sigma + \mathrm{Ec})\text{-}\textsc{Alg},$$

which interprets Ec but leaves $\ddot\Sigma$-operations uninterpreted. In this way, we achieve syntactic and semantic modularity in the style of effect handlers of algebraic operations, but for the non-algebraic operation *catch*.

Paper Organisation For the first part of the paper,

- in Section 2, we review the concept of monoids in monoidal categories and their applications in computational effects;
- in Section 3, we develop Fiore and Hur (2009)'s *equational systems*, a convenient framework for presenting algebraic theories. In particular, we study morphisms between equational systems and their colimits, allowing us to construct equational systems by 'gluing' smaller ones;
- in Section 4, we develop a type theory for describing constructions and equational systems over monoidal categories;
- in Section 5, we come back to theories of monoids with operations and see some concrete examples;
- in Section 6, we classify operations on monoids into finer families and study the connections between these families.

For the second part of the paper,

- in Section 7, we introduce *modular models* of a theory the more general concept of *model transformers*;
- in Section 8, we show general constructions and examples of modular models.

Along these lines, the contribution that we make is a unified general framework for defining both *theories* and *models* of effectful operations in a modular way.

Changes from the Conference Version This paper is the extended version of the paper *Modular Models of Monoids with Operations* (Yang and Wu, 2023) presented at the 28th ACM SIGPLAN International Conference on Functional Programming (ICFP 2023). The main differences between this version and the conference version are as follows:

1. Proofs omitted in the conference version are included, along with some new theorems, remarks, and examples throughout the paper.
2. Definitions and theorems in Sections 2 to 6 are revised to assume only the existence of the free algebras of the relevant equational systems, instead of assuming chain-cocontinuity of signature functors globally as in the conference version. Consequently, these results apply to small-complete small category, which is useful when working in type theories with impredicative universes.
3. The introduction to the metalanguage of monoidal categories is refactored to a separate section, and they are treated more carefully under the name *monoidal algebraic theories*. The main technical change is that in Section 4.3 we identify a class of monoidal algebraic theories that corresponds to equational systems, while in the conference version, we use the metalanguage to present the components of equational systems in an ad-hoc way.
4. The development of modular models in Sections 7 to 8 is greatly revised and extended. The ad-hoc definition in the conference version is replaced by a definition based on lifting functors along fibrations, which increases the applicability of the theory. Also, morphisms between modular models are now discussed, which allows us to show several new constructions with universal properties and (co)limits of modular models.

Since this extended version has become rather long, we opt in a more fine-grained numbering system to help the reader navigate in the paper more easily: (1) all theorem-like environemnts share the same counter of the format 'ABC∗XYZ' where 'ABC' is the current section number and 'XYZ' is the number of the environment within that section; (2) text paragraphs explaining one idea are grouped together and also numbered by an anonymous environment; (3) the materials that are new or received signicant changes from the conference version are numbered in green, such as Lemma 5∗13 and Lemma 6∗13 (although the colour of these hyperlinks are still the same as others for consistency).

1.1 Related Work

1.1∗1 (Effect Handlers). The most closely related work is the line of research on *handlers of algebraic effects* introduced by Plotkin and Pretnar (2009, 2013). Semantically, handlers are models of first-order algebraic theories, and are used for interpreting free algebras. As a programming construct, handlers offer a composable approach to user-defined algebraic effects, essentially relying on the fact that algebraic operations can be lifted canonically. Many implementations of handlers have been developed, both as libraries (e.g. Kammar et al. (2013)) and languages (e.g. Bauer and Pretnar (2015)). Our work is inspired by Schrijvers et al. (2019)'s *modular handlers*, which are handlers polymorphic in the unhandled effects.

Handlers of algebraic effects have been generalised in several directions. Wu et al. (2014) observe that implementing *scoped operations* as handlers leads to non-modularity issues, and they propose modelling these operations with higher-order abstract syntax and generalising handlers to this setting. Later, the categorical foundation of handlers of scoped operations were studied by Piróg et al. (2018) and Yang et al. (2022). In another direction, Pieters et al. (2020) generalised handlers from algebraic operations on monads to monoids. In comparison to the present work, Piróg et al. (2018) and Yang et al. (2022) do not consider modularity of models, and Pieters et al. (2020) only consider algebraic operations. Hence our initial motivation of the present work was to develop a clear categorical formulation of equational theories of not necessarily algebraic operations and their modular models in the generality of monoids.

A notable deviation of our definition of modular models from the standard notion of effect handlers is that our modular models are required to have a monoid structure, rather than only modelling the operations. The distinction is analogous to Arkor and Fiore (2020)'s models of *simply typed syntax* versus *simple type theories*. We believe that our deviation is a reasonable choice since (i) many practical examples of modular handler have monoid structures anyway; and (ii) many theories of non-algebraic operations inherently involve monoid operations in their equational laws, such as Example 5∗15, so a handler for such a theory without a monoid structure makes little sense.

1.1∗2 (Monad Transformers). Moggi (1989), Wadler (1990), and Spivey (1990) pioneered using *monads* to model computational effects in programming languages. To achieve modularity, Moggi (1989) introduced *monad transformers*. Later, Jaskelioff (2009) showed how to lift $\hat{\Sigma}$-operations along *functorial monad transformers*, and this result was generalised to *monoid transformers* by Jaskelioff and Moggi (2010). Our notion of *modular models* is a conceptual development of monoid transformers: besides transforming monoids into new

ones, how to equip them with new operations and lift existing operations are also part of the definition. Thus philosophically, modular models are more like Plotkin and Pretnar (2013)'s handlers, since they are meant to interpret initial algebras of equational theories that model computational effects, rather than modelling the effects themselves.

Liang et al. (1995) introduced *monad transformers* in Haskell to overcome the problem that monads do not straightforwardly compose. Their implementation includes *type classes* of monads with certain operations, which can be seen as a Haskell realisation of algebraic theories, but they need to lift existing operations along monad transformers in ad-hoc ways.

1.1∗3 (Theories and Syntax). *Algebraic theories* and their connections to *Lawvere theories* and *monads* have been studied for decades (see e.g. (Adamek et al., 2010)). In this paper, we use Fiore and Hur (2009)'s *equational systems* to define equational theories for their generality and simplicity. *Abstract syntax* can be modelled as initial algebras of theories (Goguen et al., 1977), and Fiore et al. (1999) show that abstract syntax with variable bindings can be modelled as initial algebras in a presheaf category, and they introduce Σ-*monoids*. Building on their work, we have investigated connections between families of theories of Σ-monoids, and their modular models.

Their work leads to the line of research on *second-order algebraic theories* (Fiore and Hur, 2010; Fiore and Mahmoud, 2010; Fiore, 2008; Fiore and Szamozvancev, 2022). Ghani et al. (2006) also model the syntax of *explicit substitution* in a functorial category, which Piróg et al. (2018) base on to model scoped operations. In comparison, the focus of the present work is (modular) models of abstract syntax, rather than syntax itself.

Another line of work that is very closely related to Σ-monoids is the work on using *modules* of monads/monoids as the signatures of operations; see e.g. the recent exposition by Lamiaux and Ahrens (2024).

2 Notions of Computation as Monoids in Monoidal Categories

2∗1. A *monoidal category* is a category $\mathscr{E}$ equipped with a functor $\Box : \mathscr{E} \times \mathscr{E} \to \mathscr{E}$, called the *monoidal product* (and sometimes the *tensor product*), an object $I \in \mathscr{E}$, called the *monoidal unit*, and three natural isomorphisms

$$\alpha_{A,B,C} : A \Box (B \Box C) \cong (A \Box B) \Box C, \qquad \lambda_A : I \Box A \cong A, \qquad \rho_A : A \Box I \cong A$$

satisfying some coherence axioms (Mac Lane, 1998, §VII.1) that guarantee there is a unique way using α, λ, and ρ to re-bracket an expression involving I and $\Box$ to another (Mac Lane, 1998, §VII.2) . A monoidal category is *strict* if the isomorphisms are identity morphisms. A monoidal category is *(left) closed* if all functors $- \Box A$ have right adjoints $-/A : \mathscr{E} \to \mathscr{E}$.

2∗2. A *monoid* $\langle M, \mu, \eta \rangle$ in a monoidal category $\mathscr{E}$ is an object $M \in \mathscr{E}$ with two morphisms: a *multiplication* $\mu : M \Box M \to M$ and a *unit* $\eta : I \to M$ making the following commute:

$$\begin{array}{ccccc}
(M \Box M) \Box M & & \xrightarrow{\alpha_{M,M,M}} & & M \Box (M \Box M) \\
\downarrow{\scriptstyle \mu \Box M} & & & & \downarrow{\scriptstyle M \Box \mu} \\
M \Box M & \xrightarrow[\mu]{} & M & \xleftarrow[\mu]{} & M \Box M
\end{array}$$

2∗3. In the rest of this section, we present a collection of examples of monoidal categories, in which monoids model different flavours of *notions of computations*, and will serve as the main application of the theory developed in this paper. The main messages are that

1. monoids in monoidal categories are an expressive abstraction that unifies different notions of computations;
2. by imposing suitable conditions, these monoidal categories for computations can be made closed and (co)complete, so are suitable for doing algebraic theories over them.

Some of the following examples are quite technical and my description is sketchy, but the reader does not need to fully understand the details in the examples to proceed to Section 3.

2.1 Monads and Size Issues

2.1∗1 (Monads). The category $\textsc{Endo}(\mathscr{C})$ of endofunctors $\mathscr{C} \to \mathscr{C}$ on a category $\mathscr{C}$ can be turned into a monoidal category by equipping it with functor composition $F \circ G$ as the monoidal product and the identity functor $\mathrm{Id} : \mathscr{C} \to \mathscr{C}$ as the unit. Monoids in this category are called *monads* on $\mathscr{C}$, and they are used to model *computational effects*, also called *notions of computation*, in programming languages (Moggi, 1991, 1989), where the unit $\eta : \mathrm{Id} \to M$ is understood as embedding *pure values* into computations, and the multiplication $\mu : M \circ M \to M$ is understood as flattening *computations of computations* into computations by sequentially executing them. The understanding of μ as sequential composition is better exhibited by the following co-Yoneda isomorphism:

$$(F \circ G)A = F(GA) \cong \int^{X \in \mathscr{C}} \coprod_{\mathscr{C}(X, GA)} FX$$

where $\int^X$ denotes a coend and $\coprod_{\mathscr{C}(X,GA)}$ denotes a $\mathscr{C}(X, GA)$-fold coproduct. The informal reading of the coend is that FX is the first computation, returning a value of type X, and the second computation is determined by the result of FX, given as a function $X \to GA$. So $\mu : M \circ M \to M$ is sequential composition of two computations in which the second is determined by the first one.

2.1∗2. However, the category $\textsc{Endo}(\mathscr{C})$ is usually not as well behaved as we would like for doing algebraic theories in it, even when $\mathscr{C}$ itself is a very nice category such as $\textsc{Set}$. In particular, $\textsc{Endo}(\textsc{Set})$ is not closed with respect to either cartesian products or functor composition. Moreover, $\textsc{Endo}(\textsc{Set})$ is not a nice category for doing algebraic theories; for example, some objects in $\textsc{Endo}(\textsc{Set})$, such as the covariant powerset functor $\mathscr{P}$, do not have free monads over them.

These issues about $\textsc{Endo}(\textsc{Set})$ are fundamentally related to *sizes* of sets. For instance, if $\mathscr{P} : \textsc{Set} \to \textsc{Set}$ were to have a free monad F, then F would also be *algebraically free* (nLab, 2024, Theorem 3.2), which entails that $F\emptyset$ would carry the initial $\mathscr{P}$-algebra. By Lambek's lemma (Lambek, 1968), we would then have a set $F\emptyset$ satisfying $F\emptyset \cong \mathscr{P}(F\emptyset)$, contradicting Cantor's theorem (nLab, 2024).

There are two ways to rectify the size issues: (1) we can consider some 'extremely nice' categories $\mathscr{C}$, namely, *(small-) complete small categories*, or (2) we can restrict our attention to 'reasonably nice endofunctors' on a 'reasonably nice' category $\mathscr{C}$, namely, *κ-accessible functors* on *locally κ-presentable categories*. We will describe both approaches below.

2.1∗3. When $\mathscr{C}$ is a small category that is also small-complete, the monoidal structure $\langle \circ, \mathrm{Id} \rangle$ on $\textsc{Endo}(\mathscr{C})$ is closed, with the right adjoint $-/G$ to $- \circ G$ for every $G : \mathscr{C} \to \mathscr{C}$ given by *right Kan extension* (Mac Lane, 1998, §X):

$$(F/G)A = \int_{B \in \mathscr{C}} \prod_{\textsc{Set}(A, GB)} FB, \tag{2.1}$$

which exists since the 'bound of the end' $B \in \mathscr{C}$ is small and $\mathscr{C}$ is small-complete. The unit of the adjunction $- \circ G \dashv -/G$ is given by the natural transformations $\eta_F : F \to (F \circ G)/G$ whose component at every $A \in \mathscr{C}$

$$\eta_{F,A} : FA \to \int_{B \in \mathscr{C}} \prod_{\textsc{Set}(A, GB)} F(GB)$$

is the mediating morphism induced by the wedge

$$\eta_{F,A,B} \coloneqq \langle Fk \rangle_{k \in \textsc{Set}(A, GB)} : FA \to \prod_{\textsc{Set}(A, GB)} F(GB)$$

for all $B \in \mathscr{C}$. The counit ϵ of the adjunction $- \circ G \dashv -/G$ is given by the composite

$$((F/G) \circ G)A = \int_{B \in \mathscr{C}} \prod_{\textsc{Set}(GA, GB)} FB \xrightarrow{\pi_A} \prod_{\textsc{Set}(GA, GA)} FA \xrightarrow{\pi_{id : GA \to GA}} FA$$

for all $F \in \textsc{Endo}(\mathscr{C})$ and $A \in \mathscr{C}$.

Moreover, $\textsc{Endo}(\mathscr{C})$ is also a small-complete small category: it is small because its domain and codomain categories $\mathscr{C}$ are both small; it is small-complete because limits in functor categories are computed pointwise and $\mathscr{C}$ is small-complete. We will see later in Theorem 3.1∗19 that it implies that all objects in $\textsc{Endo}(\mathscr{C})$ have free monads (and many other free algebraic structures).

2.1∗4. However, it is long known that *in classical logic* the only examples of small-complete small categories $\mathscr{C}$ are *complete preorders* (nLab, 2024). Therefore $\mathscr{C}$ being small and small-complete is classically too strong for the purpose of modelling computational effects.

However, a stunning result in categorical logic is that if we carry out category theory internally in the *effective topos* Eff, or more generally realizability toposes (Hyland, 1988; Oosten, 2008), the category Mset of *modest sets*, also known as *partial equivalence relations* (PERs), is a non-trivial small-complete small-categories, internally. The effective topos and modest sets find many application in programming language theory: for one, they provide semantics for type theories with *impredicative* polymorphism, such as System F (Girard, 1986) and the calculus of inductive constructions (Coquand and Huet, 1988; Luo, 1994) with an impredicative universe of sets, which is the type theory underlying the Rocq (previously known as Coq) proof assistant with the `-impredicative-set` option.

2.1∗5 Remark. Mathematics carried out internally in a category can also be externalised to the ambient meta-theory (Jacobs, 1999; Streicher, 2023; Phoa, 1992), telling us what an internal construction 'really means' from the external point of view. In particular, if we externalise the internal category $\textsc{Endo}(\textsc{Mset})$, we obtain a fibration $[\textsc{Mset}] \to \textsc{Eff}$; taking its fiber over $1 \in \textsc{Eff}$, we obtain an ordinary category of *realizable endofunctors* $\textsc{Endo}_r(\textsc{Mset})$, which are roughly endofunctors whose mapping on morphisms are realised

by Turing machines; see Jaskelioff and Moggi (2010, Example 2.20) or Bainbridge et al. (1990) for more information.

2.2 Finitary and Accessible Monads

2.2∗1. We have seen the approach of rectifying the size issues in $\textsc{Endo}(\mathscr{C})$ by assuming an 'extremely nice' $\mathscr{C}$ that exists only in non-classical settings. Now let us describe an alternative, classically valid, approach – restricting our attention to 'reasonably nice' endofunctors.

First we observe that the reason why the formula (2.1) does not work for arbitrary endofunctors $F, G : \textsc{Set} \to \textsc{Set}$ is that the bound of the end $B \in \textsc{Set}$ would be large while $\textsc{Set}$ is only small-complete. Therefore we can consider functors that allow us to cut $B \in \textsc{Set}$ to a small bound. An endofunctor $F \in \textsc{Endo}(\textsc{Set})$ is called *finitary* if it preserves *filtered colimits*. An informal description of finitariness is that *we lose no information if we restrict F to finite sets*. Precisely, F is finitary if we first restrict F to $F \circ V : \textsc{Fin} \to \textsc{Set}$ on the full subcategory of finite sets, where $V : \textsc{Fin} \to \textsc{Set}$ is the inclusion functor, and then take the left Kan extension of $F \circ V$ along V, the resulting functor is still isomorphic to F. In other words, we have the following equivalence, where $\textsc{Endo}_f(\textsc{Set}) \subseteq \textsc{Endo}(\textsc{Set})$ denotes the full subcategory of finitary endofunctors:

$$\textsc{Endo}_f(\textsc{Set}) \underset{-\circ V}{\overset{\textsc{Lan}_V -}{\underset{\longrightarrow}{\overset{\longleftarrow}{\cong}}}} \textsc{Set}^{\textsc{Fin}} \tag{2.2}$$

2.2∗2. The category $\textsc{Endo}_f(\textsc{Set})$ inherits the monoidal structure $\langle \circ, \mathrm{Id} \rangle$ of $\textsc{Endo}(\textsc{Set})$, which under the equivalence (2.2) corresponds to the monoidal structure $\langle \bullet, V \rangle$ on $\textsc{Set}^{\textsc{Fin}}$ (Kelly and Power, 1993; Fiore et al., 1999) where

$$Vn = n \qquad \text{and} \qquad (F \bullet G)n = \int^{m \in \textsc{Fin}} Fm \times (Gn)^m. \tag{2.3}$$

For every $G \in \textsc{Set}^{\textsc{Fin}}$, the functor $- \bullet G$ has a right adjoint

$$(F/G)n = \int_{m \in \textsc{Fin}} \prod_{\textsc{Set}(n, Gm)} Fm$$

with unit and counit similar to those in 2.1∗3. The end $\int_{m \in \textsc{Fin}}$ exists because the category $\textsc{Fin}$ of finite sets is essentially small and $\textsc{Set}$ is small complete.

Moreover, the functor $\circ : \textsc{Endo}_f(\textsc{Set}) \times \textsc{Endo}_f(\textsc{Set}) \to \textsc{Endo}_f(\textsc{Set})$ is also finitary, i.e. preserving filtered colimits in $\textsc{Endo}_f(\textsc{Set}) \times \textsc{Endo}_f(\textsc{Set})$. This is equivalent to preserving filtered colimits in both of its arguments separately: functor composition preserves (all) colimits in the first argument

$$((\mathrm{colim}_i F_i) \circ G)n = (\mathrm{colim}_i F_i)(Gn) \cong \mathrm{colim}_i (F_i(Gn))$$

because colimits of functors can be computed pointwise. It preserves filtered colimits in the second argument by the finitariness of its first argument:

$$(F \circ (\mathrm{colim}_i G_i))n = F((\mathrm{colim}_i G_i)n) \cong F(\mathrm{colim}_i G_i n) \cong \mathrm{colim}_i F(G_i n)$$

In particular, the diagram of an ω-chain is filtered, so functor composition preserves colimits of ω-chains of functors; we will use this property to construct free monads over finitary endofunctors in Section 3.1.

2.2∗3. Monoids in $\langle \textsc{Endo}_f(\textsc{Set}), \circ, \mathrm{Id}\rangle$ are called *finitary monads* on $\textsc{Set}$, and they are equivalent to (finitary) *Lawvere theories* [Power 1999, Theorem 4.2; Adamek et al. 2010, Theorem A.38] and the closely related *abstract clones* [Cohn 1981, page 132; Fiore et al. 1999, Section 3]. Computational effects modelled by Lawvere theories are usually called *algebraic effects* (Plotkin and Power, 2004, 2002).

Apart from modelling computational effects, a related but slightly different application of monoids in $\textsc{Endo}_f(\textsc{Set})$, or equivalently $\textsc{Set}^{\textsc{Fin}}$, is modelling *abstract syntax with variable binding* (Fiore et al., 1999; Fiore and Szamozvancev, 2022). In this case, a monoid $M \in \textsc{Set}^{\textsc{Fin}}$ is understood as a family of sets of terms indexed by the number of *variables* in the context. The monoid unit $V \to M$ is then embedding *variables* as M-terms, and the monoid multiplication $M \bullet M \to M$ is *simultaneous substitution* of terms for variables.

Yet another interesting reading of monoids of $\langle \bullet, V\rangle$ due to Fiore and Staton (2014) is computations supporting (i) binding a piece of code to a code pointer and (ii) jumping to a code pointer. Based on this reading, the monoidal category $\langle \textsc{Set}^{\textsc{Fin}}, \bullet, V\rangle$ provides an adequate denotational semantics of a calculus of substitution/jumps, on which algebraic effects can be encoded.

2.2∗4. The adjunction $\textsc{Endo}_f(\textsc{Set}) \cong \textsc{Set}^{\textsc{Fin}}$ can be generalised to endofunctors on categories other than $\textsc{Set}$: we can replace (1) $\textsc{Set}$ with any *locally κ-presentable* (lκp) category $\mathscr{C}$ for a regular cardinal κ, (2) $\textsc{Fin}$ with the subcategory $\mathscr{C}_\kappa$ of *κ-presentable objects* in $\mathscr{C}$, and (3) finitary functors with *κ-accessible* functors $\textsc{Endo}_\kappa(\mathscr{C})$ (Adámek and Rosicky, 1994). This results in a cocomplete closed monoidal category $\langle \textsc{Endo}_\kappa(\mathscr{C}), \circ, \mathrm{Id}\rangle$, on which $\circ$ is κ-accessible. The κ-accessibility of $\circ$ implies that there is a sufficiently large limit ordinal α such that $\circ : \textsc{Endo}_\kappa(\mathscr{C}) \times \textsc{Endo}_\kappa(\mathscr{C}) \to \textsc{Endo}_\kappa(\mathscr{C})$ preserves colimits of all α-chains, a property we will later use for constructing free monoids.

2.2∗5. All lκp categories are necessarily (small-) complete and cocomplete, and they cover a wide range of categories that are used for modelling programming languages. Examples of lκp categories include:

- all presheaf categories $\textsc{Set}^{\mathscr{D}}$ for small categories $\mathscr{D}$ and $\kappa = \aleph_0$ (an lκp category is called locally *finitely* presentable when $\kappa = \aleph_0$), and more generally all Grothendieck toposes for some κ (Borceux, 1994, 3.4.16);
- categories of models for *essentially algebraic theories* (Adámek and Rosicky, 1994, §3.D), which include the category $\textsc{Cat}$ of small categories for $\kappa = \aleph_0$, the category $\omega\textsc{Cpo}$ of ω-complete partial orders for $\kappa = \aleph_1$ (however, the category of *directed* complete partial orders is not lκp for any κ);
- functor categories $\mathscr{C}^{\mathscr{D}}$ for small categories $\mathscr{D}$ and lκp $\mathscr{C}$;
- moreover, when $\mathscr{C}$ is lκp, it is automatically lλp for any $\lambda > \kappa$.

Therefore, lκp categories provide a nice setting for algebraic theories in the context of programming language semantics.

2.2∗6 Remark. Let $\kappa < \lambda$ be two regular cardinals and $\mathscr{C}$ be an lκp category. As we mentioned above $\mathscr{C}$ is also lλp, and every κ-accessible endofunctor on $\mathscr{C}$ is also going to be λ-accessible because every λ-filtered colimit is also κ-filtered. The monoidal structure $\langle \circ, \mathrm{Id}\rangle$ in $\textsc{Endo}_\kappa(\mathscr{C})$ coincides with $\langle \circ, \mathrm{Id}\rangle$ in $\textsc{Endo}_\lambda(\mathscr{C})$, so they can be both denoted by $\circ$

unproblematically. However, the closed structure F/G in $\textsc{Endo}_\kappa(\mathscr{C})$, given by

$$(F/G)n = \int_{x \in \mathscr{C}_\kappa} \prod \mathscr{C}(n, Gx).\ Fx$$

in fact depends on κ, and it has its universal property with respect to only κ-accessible functors, so it may not coincide with the closed structure in $\textsc{Endo}_\lambda(\mathscr{C})$. In the same way, F/G computed in $\textsc{Endo}_\kappa(\mathscr{C})$ does not have to be the right Kan extension of F along G among the bigger category $\textsc{Endo}(\mathscr{C})$ of all endofunctors or the category $\textsc{Endo}_{acc}(\mathscr{C})$ of all accessible endofunctors, in which every object may have a different choice of κ. Therefore if we were pedantic about the notation, we should write $F/_\kappa G$ instead of just F/G.

2.3 Strong Monads

2.3∗1. The multiplication $\mu : M \circ M \to M$ of a monad $M : \mathscr{C} \to \mathscr{C}$ allows one to compose two effectful computations $f : A \to MB$ and $g : B \to MC$ by

$$A \xrightarrow{f} MB \xrightarrow{Mg} M(MC) \xrightarrow{\mu_C} MC.$$

However, to give semantics to effectful programming languages with a *structural* context of variables as done by Moggi (1991, 1989), what we need is slightly stronger: for all objects $\Gamma \in \mathscr{C}$ (thought of as variable contexts) and morphisms $f : \Gamma \times A \to MB$ and $g : \Gamma \times B \to MC$ (two computations under a context Γ), we would like to have a morphism $\Gamma \times A \to MC$ (the sequential composition of f and g). The structure of monads $\langle M, \mu, \eta \rangle$ is not sufficient for doing this, and we need additionally a natural transformation

$$s_{\Gamma,B} : \Gamma \times MB \to M(\Gamma \times B),$$

with which we have the composite

$$\Gamma \times A \xrightarrow{\langle \pi_1, f \rangle} \Gamma \times MB \xrightarrow{s_{\Gamma,B}} M(\Gamma \times B) \xrightarrow{Mg} M(MC) \xrightarrow{\mu_C} MC.$$

To make this way of composing effectful computations associative and the pure computation $(\eta_A \cdot \pi_2) : \Gamma \times A \to MA$ an identity, the natural transformation s must satisfy certain coherence conditions (see (Moggi, 1991, Definition 3.2)). The natural transformation s is called a *strength* for the monad M, and the tuple $\langle M, \mu, \eta, s \rangle$ is called a *strong monad*.

2.3∗2. Strong monads are monoids in the monoidal category of *strong endofunctors*. A strong endofunctor $\langle F, s \rangle$ on a category $\mathscr{C}$ with finite products is a functor $F : \mathscr{C} \to \mathscr{C}$ with a natural transformation $s_{\Gamma,B} : \Gamma \times FB \to F(\Gamma \times B)$ making the following commute:

$$\begin{array}{ccc} 1 \times FB & & \\ {\scriptstyle s_{1,B}}\downarrow & \searrow^{\lambda_{FB}} & \\ F(1 \times B) & \xrightarrow[F\lambda_B]{} & FB \end{array} \tag{2.4}$$

$$\begin{array}{ccccc} (\Gamma' \times \Gamma) \times FB & \xrightarrow{\alpha_{\Gamma',\Gamma,FB}} & \Gamma' \times (\Gamma \times FB) & \xrightarrow{\Gamma' \times s_{\Gamma,B}} & \Gamma' \times F(\Gamma \times B) \\ {\scriptstyle s_{\Gamma' \times \Gamma, B}}\downarrow & & & & \downarrow{\scriptstyle s_{\Gamma', \Gamma \times B}} \\ F((\Gamma' \times \Gamma) \times B) & & \xrightarrow[F\alpha_{\Gamma',\Gamma,B}]{} & & F(\Gamma' \times (\Gamma \times B)) \end{array} \tag{2.5}$$

where λ and α are the left unitor and associator for the cartesian monoidal structure $\langle \times, 1 \rangle$. Moreover, *strong natural transformations* between $\langle F, s^F \rangle$ and $\langle G, s^G \rangle$ are natural transformations $\tau : F \to G$ making the following commute:

$$\begin{array}{ccc} \Gamma \times FB & \xrightarrow{\Gamma \times \tau_B} & \Gamma \times GB \\ {\scriptstyle s^F_{\Gamma,B}}\downarrow & & \downarrow{\scriptstyle s^G_{\Gamma,B}} \\ F(\Gamma \times B) & \xrightarrow[\tau_{\Gamma \times B}]{} & G(\Gamma \times B) \end{array}$$

2.3∗3. Strong endofunctors on $\mathscr{C}$ and strong natural transformations can be collected into a category $\textsc{Endo}_s(\mathscr{C})$. The category $\textsc{Endo}_s(\mathscr{C})$ has a monoidal structure $\langle \circ_s, \mathrm{Id}_s \rangle$ where Id_s is the identity functor equipped with the identity strength, and $\circ_s$ is the composition of strong functors given by

$$\langle F, s^G \rangle \circ_s \langle G, s^F \rangle = \langle F \circ G,\ (F s^G_{\Gamma,B} \cdot s^F_{\Gamma,GB})_{\Gamma,B \in \mathscr{C}} \rangle.$$

Strong monads on $\mathscr{C}$ are precisely monoids in this monoidal category. When $\mathscr{C}$ is a cartesian *closed* category, strong functors and natural transformation are the same as $\mathscr{C}$*-enriched* functors and natural transformations (McDermott and Uustalu, 2022; Kock, 1972).

2.3∗4. When $\mathscr{C}$ is $\textsc{Set}$, or slightly more generally a full subcategory of $\textsc{Set}$ closed under finite products of $\textsc{Set}$, every $F : \mathscr{C} \to \mathscr{C}$ has a strength:

$$\begin{aligned} &s_{\Gamma,B} : \Gamma \times FB \to F(\Gamma \times B) \\ &s_{\Gamma,B}\ \langle \gamma, f \rangle = F\ (\lambda b.\ \langle \gamma, b \rangle)\ f \end{aligned} \tag{2.6}$$

which can be readily checked to satisfy the laws of strengths (2.4, 2.5). In fact, this is the only strength for F: let $t_{\Gamma,B} : \Gamma \times FB \to F(\Gamma \times B)$ be any strength for F, for all $\gamma \in \Gamma$ and $f \in FB$, the naturality of t implies the commutativity of

$$\begin{array}{ccc} 1 \times FB & \xrightarrow{t_{1,B}} & F(1 \times B) \\ {\scriptstyle \gamma \times FB}\downarrow & & \downarrow{\scriptstyle F(\gamma \times B)} \\ \Gamma \times FB & \xrightarrow[t_{\Gamma,B}]{} & F(\Gamma \times B) \end{array}$$

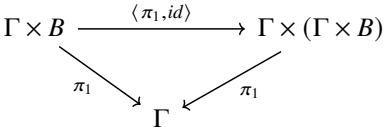
Evaluating the two paths at $\langle *, f \rangle \in 1 \times FB$, we have

$$t_{\Gamma,B} \langle \gamma, f \rangle = F\ (\gamma \times B)\ (t_{1,B}\ \langle *, f \rangle).$$

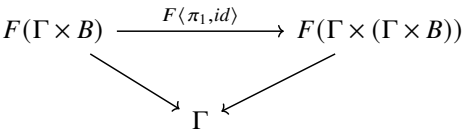
However, by the law (2.4), $t_{1,B}\ \langle *, f \rangle$ has to be $F\lambda_B^{-1} \langle *, f \rangle$. Hence $t_{\Gamma,B} \langle \gamma, f \rangle$ is equal to the canonical strength s (2.6) above. However, functors on a general category $\mathscr{C}$ may have no or non-unique strengths; see McDermott and Uustalu (2022) for some counter-examples.

2.3∗5. As a side remark to readers who know fibred category theory, the fact that functors on sets are canonically strong is not so much about sets, but more about the ability to use the value γ in the current 'context of variables' when defining $(\lambda b.\ \langle \gamma, b \rangle) : B \to \Gamma \times B$ in (2.6). Indeed, for every $\mathscr{C}$ with finite limits and a fibred endofunctor $F : \mathscr{C}^{\to} \to \mathscr{C}^{\to}$ on the fundamental fibration $\textsc{Cod} : \mathscr{C}^{\to} \to \mathscr{C}$, the restriction of F to the fiber over $1 \in \mathscr{C}$ induces a functor $\overline{F} : \mathscr{C} \cong \mathscr{C}_1^{\to} \to \mathscr{C}_1^{\to} \cong \mathscr{C}$. The functor $\overline{F}$ has a strength similar to (2.6): firstly we

have the following morphism in the fiber over every $\Gamma \in \mathscr{C}$,



which plays the same role as $(\lambda b.\, \langle \gamma, b \rangle) : B \to \Gamma \times B$ in (2.6), and then this morphism is mapped by the fibred functor F to a morphism still over Γ:



(2.7)

Because $\pi_1 : \Gamma \times B \to \Gamma$ is the pullback of $B \to 1$ along $\Gamma \to 1$, and F as a fibred functor preserves pullbacks, $F(\Gamma \times B) \to \Gamma$ is also a pullback of $FB \to 1$ (i.e. $\overline{F}B \to 1$) along $\Gamma \to 1$. Another choice of the pullback of $FB \to 1$ along $\Gamma \to 1$ is just $\pi_1 : \Gamma \times \overline{F}B \to \Gamma$. Hence there is a canonical isomorphism $F(\Gamma \times B) \cong \Gamma \times \overline{F}B$, and similarly $F(\Gamma \times (\Gamma \times B)) \cong \Gamma \times \overline{F}(\Gamma \times B)$, so the morphism $F\langle \pi_1, id \rangle$ in (2.7) gives us a morphism $\Gamma \times \overline{F}B \to \overline{F}(\Gamma \times B)$. This can be checked to be a strength for $\overline{F}$. However, from the other direction we cannot necessarily obtain a fibred endofunctor on $\textsc{Cod} : \mathscr{C}^{\to} \to \mathscr{C}$ from a strong functor on $\mathscr{C}$ in general. Instead, strong functors on $\mathscr{C}$ are equivalent to fibred endofunctors on the *simple fibration* over $\mathscr{C}$ (Jacobs, 1999, Proposition 2.6.9) (and fibred endofunctors over the fundamental fibration restrict to fibred endofunctors over the simple fibration).

2.3∗6. Similar to the setting of ordinary monads, to have a 'nice' monoidal category of strong monads, we need to either (1) consider small-complete small categories $\mathscr{C}$ or (2) κ-accessible strong monads. More precisely, denote by $\textsc{Endo}_{s\kappa}(\mathscr{C})$ the full subcategory of $\textsc{Endo}_s(\mathscr{C})$ that contains strong functors whose underlying functors are κ-accessible.

When $\mathscr{C}$ is *lκp as a cartesian closed category*, which means that $\mathscr{C}$ is lκp and cartesian closed, and that the κ-presentable objects of $\mathscr{C}$ are closed under finite products, the category $\textsc{Endo}_{s\kappa}(\mathscr{C})$ is also lκp and has a closed monoidal structure of functor composition and the identity functor. The primary example of such setting is $\mathscr{C} = \omega\textsc{Cpo}$ for modelling general recursion. We refer the reader to Kelly and Power (1993, §4) and Kelly (1982) for details.

2.4 Graded Monads

2.4∗1. Another generalisation of monads is to index the monad with some *grades* that track quantitative information about the effects performed by a computation (Katsumata, 2014; Katsumata et al., 2022; McDermott and Uustalu, 2022). Precisely, let $\langle \mathscr{G}, \cdot, 1 \rangle$ be any small strict monoidal category, whose objects we call grades. A *$\mathscr{G}$-graded monad* on a category $\mathscr{C}$ is a functor $M : \mathscr{G} \to \textsc{Endo}(\mathscr{C})$ equipped with natural transformations

$$\eta : \mathrm{Id} \to M1 \qquad\qquad \mu_{a,b} : (Ma) \circ (Mb) \to M(a \cdot b) \tag{2.8}$$

natural in $a, b \in \mathscr{G}$, satisfying laws similar to those of monads (Katsumata, 2014).

2.4∗2. For example, for tracking operations performed by a computation, $\mathscr{G}$ can be the poset of sets of operation names, ordered by inclusion, with the monoidal structure $1 = \emptyset$ and $a \cdot b = a \cup b$. And for tracking the number of nondeterministic choices made by a computation, $\mathscr{G}$ can be the poset $\langle \mathbb{N}, \leqslant \rangle$ with monoidal structure $\langle 0, + \rangle$.

2.4∗3. Again, to avoid the size issues when doing algebraic theories (2.1∗2), we can either require $\mathscr{C}$ to be small and small-complete or consider only *κ-accessible $\mathscr{G}$-graded monads* $M : \mathscr{G} \to \textsc{Endo}_\kappa(\mathscr{C})$ for $\mathrm{lkp}\ \mathscr{C}$.

2.4∗4. Similar to the ungraded situation, κ-accessible graded monads are equivalent to monoids in the functor category $\textsc{Endo}_\kappa(\mathscr{C})^{\mathscr{G}}$ equipped with the following monoidal structure similar to the Day tensor:

$$I = \coprod_{\mathscr{G}(1,-)} \mathrm{Id} \qquad\qquad F * G = \int^{a,b \in \mathscr{G}} \coprod_{\mathscr{G}(a \cdot b, -)} (Fa \circ Gb). \tag{2.9}$$

This monoidal product has right adjoints given by

$$(G \multimap F)a = \int_{b \in \mathscr{G}} \int_{m \in \mathscr{C}_\kappa} \prod_{\mathscr{C}(-,(Gb)m)} F(a \cdot b)m.$$

2.4∗5. The equivalence between κ-accessible graded monads (2.8) and monoids for the monoidal structure (2.9) seems to be unwritten folklore, so we here show a sketch of the proof using (co)end calculus (Loregian, 2021). The correspondence between $\eta : I \to M$ for the monoidal structure (2.9) and $\eta : \mathrm{Id} \to M1$ (2.8) is

$$\begin{aligned}
&\textsc{Hom}(I, M) \\
\cong\ &\{\text{We write } [A, B] \text{ for } \mathscr{C}(A, B) \text{ below}\} \\
&\textstyle\int_{c \in \mathscr{G}} \int_{x \in \mathscr{C}_\kappa} [\coprod_{\mathscr{G}(1,c)} x,\ (Mc)x] \\
\cong\ &\textstyle\int_{c \in \mathscr{G}} \int_{x \in \mathscr{C}_\kappa} [\mathscr{G}(1, c) \times x,\ (Mc)x] \\
\cong\ &\textstyle\int_{c \in \mathscr{G}} \int_{x \in \mathscr{C}_\kappa} [\mathscr{G}(1, c), [x,\ (Mc)x]] \\
\cong\ &\{[\mathscr{G}(1, c), -] \text{ preserves limits}\} \\
&\textstyle\int_{c \in \mathscr{G}} [\mathscr{G}(1, c), \int_{x \in \mathscr{C}_\kappa} [x,\ (Mc)x]] \\
\cong\ &\{\text{ Yoneda lemma }\} \\
&\textstyle\int_{x \in \mathscr{C}_\kappa} [x,\ (M1)x] \\
\cong\ &\textsc{Hom}(\mathrm{Id}, M1)
\end{aligned}$$

The correspondence between $\mu : M * M \to M$ (2.8) and families of natural transformations $\mu_{a,b} : (Ma) \circ (Mb) \to M(a \cdot b)$ (2.9) natural in $a, b \in \mathscr{G}$ is

$$\begin{aligned}
&\textsc{Hom}(M * M, M) \\
\cong\ &\textstyle\int_{c \in \mathscr{G}} \textsc{Hom}((M * M)c, Mc) \\
\cong\ &\textstyle\int_{c \in \mathscr{G}} \int_{x \in \mathscr{C}_\kappa} [\int^{a,b \in \mathscr{G}} \coprod_{\mathscr{G}(a \cdot b, -)} (Ma(Mbx)), (Mc)x] \\
\cong\ &\textstyle\int_{c \in \mathscr{G}} \int_{x \in \mathscr{C}_\kappa} \int_{a,b \in \mathscr{G}} [\mathscr{G}(a \cdot b, -), [(Ma(Mbx)), (Mc)x]] \\
\cong\ &\textstyle\int_{x \in \mathscr{C}_\kappa} \int_{a,b \in \mathscr{G}} \int_{c \in \mathscr{G}} [\mathscr{G}(a \cdot b, -), [(Ma(Mbx)), (Mc)x]] \\
\cong\ &\textstyle\int_{x \in \mathscr{C}_\kappa} \int_{a,b \in \mathscr{G}} [Ma(Mbx), (M(a \cdot b))x] \\
\cong\ &\textstyle\int_{a,b \in \mathscr{G}} \textsc{Hom}(Ma \circ Mb, M(a \cdot b))
\end{aligned}$$

We will not torture the reader with checking the correspondence of the laws here.

2.4∗6. We have seen monads and several variations – realizable, accessible, strong, graded – all formulated as monoids in monoidal categories. Despite their differences, they all model notions computations that support sequential compositions in a sense. In the following, let us have a look at some other monoids that are conceptually quite different from monads.

2.5 Cartesian Monoids

2.5∗1. Every cartesian category $\mathscr{C}$, i.e. a category with finite products, can be equipped with the product $\times$ as the monoidal product and the terminal object $1 \in \mathscr{C}$ as the monoidal unit. When $\mathscr{C}$ has all exponentials B^A, $\mathscr{C}$ is then a cartesian closed category. Particularly, the category Set is a closed monoidal category in this way. Monoids in Set are precisely the usual notion of monoids, such as the set of lists with concatenation and empty list.

2.5∗2. For lκp and cartesian closed $\mathscr{C}$, the category $\textsc{Endo}_\kappa(\mathscr{C})$ is cartesian closed. The cartesian unit and product in $\textsc{Endo}_\kappa(\mathscr{C})$ are defined pointwise:

$$1n = 1_{\mathscr{C}} \qquad\qquad (F \times G)n = Fn \times Gn$$

The exponential is given by

$$(F^G)n = \int_{m \in \mathscr{C}_\kappa} \prod_{\mathscr{C}(n,m)} (Fm)^{Gm}.$$

A computational interpretation of cartesian monoids in $\textsc{Endo}_\kappa(\mathscr{C})$ is that they model *notions of independent computations*: cartesian multiplication $M \times M \to M$ composes two computations that have no dependency, whereas monadic multiplication $M \circ M \to M$ composes a computation with another that depends on the result of the former.

2.6 Applicative Functors

2.6∗1. Between the two extremes of $M \circ M$ and $M \times M$, there are monoidal structures on $\textsc{Endo}_\kappa(\mathscr{C})$ that allow computations to have restricted dependency. One of them is the Day convolution (Day, 1970) induced by cartesian products: the Day monoidal structure on $\textsc{Endo}_\kappa(\textsc{Set})$ has as unit the identity functor, and the monoidal product is given by the following coend formula:

$$(F * G)n = \int^{m,k \in \textsc{Set}_\kappa \times \textsc{Set}_\kappa} Fm \times Gk \times n^{(m \times k)}. \tag{2.10}$$

Informally, $F * G$ models two computations Fn and Gm that are *almost independent* except that their return values are combined by a pure function $n^{(m \times k)}$. This structure is symmetric and closed, with the right adjoint to $- * G$ given by

$$G \multimap F = \int_{n \in \textsc{Set}_\kappa} (F(- \times n))^{Gn}.$$

More generally, we can replace Set with any *lκp as a cartesian closed* category $\mathscr{V}$ (Kelly, 1982) and also the coend in (2.10) with a $\mathscr{V}$-enriched coend, which allows us to give a more accurate formulation of applicatives in functional languages with general recursion by setting $\mathscr{V} = \omega\textsc{Cpo}$. Alternatively, we can consider small-complete small $\mathscr{C}$ to work around the size issues as in 2.1∗3.

2.6∗2. Monoids for $\langle *, \mathrm{Id}\rangle$ are called *applicative functors* or simply *applicatives*. They are equivalent to *lax monoidal functors* $\langle \textsc{Set}, \times, 1\rangle \to \langle \textsc{Set}, \times, 1\rangle$ (McBride and Paterson, 2008; Paterson, 2012). A practical application of them is *build systems* (Mokhov et al., 2018), since usually the result of a building task does not affect what the next building task is.

2.6∗3. Applicatives are a weaker notion than strong monads, as intuitively independence is a special case of dependence. Precisely, for any two functors $F, G \in \textsc{Endo}_\kappa(\textsc{Set})$, there are canonical strengths (2.6):

$$s^F_{XY} : FX \times Y \to F(X \times Y), \qquad s^G_{XY} : GX \times Y \to G(X \times Y),$$

and then there is a natural transformation $e : F * G \to F \circ G$ as follows:

$$\begin{aligned}(F * G)n = \int^{mk} Fm \times Gk \times n^{(m\times k)} &\to \int^{mk} F(G(m \times k \times n^{(m\times k)})) \\ &\to \int^{mk} F(Gn) \cong F(Gn)\end{aligned}$$

where the first arrow is repeated uses of the strengths of F and G, and the second arrow is function evaluation. Consequently, for any monad $\langle M, \mu, \eta\rangle$ on $\textsc{Set}$, it induces an applicative functor with unit η and multiplication

$$M * M \xrightarrow{e} M \circ M \xrightarrow{\mu} M.$$

However, there are many applicative functors that are not obtained from monads in this way (Paterson, 2012; McBride and Paterson, 2008).

2.7 Hughes Arrows

2.7∗1. Between applicatives and monads, there is a middle ground of notions of computations that allows *data dependency* but not *control dependency*, known as *Hughes arrows*, or simply *arrows* (Hughes, 2000; Lindley et al., 2011; Jacobs et al., 2009), or *strong promonads* (Román, 2022). Unlike monads and applicatives, arrows are not monoids in $\textsc{Endo}(\mathscr{C})$, but in the category of *strong endoprofunctors*.

An *endoprofunctor* P on a small category $\mathscr{C}$ is just a functor $P : \mathscr{C}^{\mathrm{op}} \times \mathscr{C} \to \textsc{Set}$. Informally, the set $P(a, b)$ is P-computations from type a to type b. Assuming $\mathscr{C}$ has finite products, a *strong* endoprofunctor on $\mathscr{C}$ is additionally equipped with a family s of morphisms, called a *strength*,

$$s_{abc} : P(a, b) \to P(a \times c, b \times c),$$

natural in a, b and dinatural in c satisfying certain coherence conditions (Rivas and Jaskelioff, 2017, Def. 7.1). The strength s_{abc} informally means every computation in $P(a, b)$ can also be run alongside some unused data c. A *strong natural transformation* $\tau : \langle P, s^P\rangle \to \langle Q, s^Q\rangle$ is a natural transformation $\tau : P \to Q$ making the following diagram commute:

$$\begin{array}{ccc} P(a,b) & \xrightarrow{s^P_{abc}} & P(a\times c, b\times c) \\ {\scriptstyle \tau_{a,b}}\downarrow & & \downarrow{\scriptstyle \tau_{a\times c, b\times c}} \\ Q(a,b) & \xrightarrow[s^Q_{abc}]{} & P(a\times c, b\times c) \end{array}$$

2.7∗2. Strong endoprofunctors and strong natural transformations between them form a category $\textsc{EndoPro}_s(\mathscr{C})$ (Rivas and Jaskelioff, 2017), which can be equipped with a monoidal structure $\langle I, \otimes \rangle$:

$$I(a,b) = \mathscr{C}(a,b) \qquad (P \otimes Q)(a,b) = \int^{x \in \mathscr{C}} P(a,x) \times Q(x,b) \tag{2.11}$$

where the associated strength of I is

$$s^I_{abc} := (- \times c) : \mathscr{C}(a,b) \to \mathscr{C}(a \times c, b \times c),$$

and the strength of $P \otimes Q$ is the composite

$$\begin{aligned} \int^x P(a,x) \times Q(x,b) &\longrightarrow \int^x P(a \times c, x \times c) \times Q(x \times c, b \times c) \\ &\longrightarrow \int^y P(a \times c, y) \times Q(y, b \times c) \end{aligned}$$

where the first arrow is $\int^x s^P_{axb} \times s^Q_{xbc}$ and the second arrow is the coprojection morphism for $x \times c$ of the coend.

Informally, the product $P \otimes Q$ are two computations P and Q with some type of data x flowing from P to Q, so it allows more dependency than applicatives, but unlike $M \circ M$, it does not allow the second computation to *dynamically* depend on the return value of P (see Pieters et al. (2020) and Lindley et al. (2011) for more detailed comparisons).

2.7∗3. While the category of endoprofunctors *without* strengths has a closed monoidal structure $\langle I, \otimes \rangle$ with the same definition of I and $\otimes$ as those in (2.11), with the right adjoints $P \multimap -$ to $- \otimes P$ given by

$$(P \multimap Q)(a,b) = \int_{x \in \mathscr{C}} [P(b,x), Q(a,x)],$$

We do not know whether endofunctors *with* strengths $\textsc{EndoPro}_s(\mathscr{C})$ is also closed.

2.7∗4 Remark. In this section, we have paid special attention to the cocompleteness and closedness of monoidal categories. However, in this paper cocompleteness will play a much more predominant role than closedness: we need cocompleteness to construct free algebras of algebraic theories, while closedness will only be used for a few concrete constructions, such as the concrete construction of the free monoid over A as the list object $\mu X.\, 1 + A \,\square\, X$.

3 Equational Systems and Translations

3∗1. We have seen monoids in various monoidal categories, but if the only thing that we know about a monoid is its unit and multiplication, then it is barely interesting. Instead, monoids in practice usually come with operations: for example, the state monad $(- \times S)^S$ comes with operations for reading and writing the mutable state; the exception monad $- + E$ has operations for throwing and catching exceptions; the ordinary monoid in $\textsc{Set}$ of lists with concatenation has the operation of appending an element to a list.

Therefore, we shall have a way to talk about algebraic theories in monoidal categories systematically. In this paper, we will use Fiore and Hur's [2007; 2009] *equational systems*, a simple but powerful framework subsuming (enriched) algebraic theories. Importantly, the model theory of equational systems is well developed: Fiore and Hur established conditions for the existence of free algebras, cocompleteness of the category of models, and monadicity.

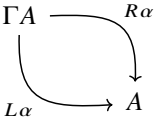
3∗2. In the following, we first briefly introduce equational systems and Fiore and Hur's theorem for the existence of free algebras of equational systems, and we also show a *constructively valid* proof of the theorem for small-complete small categories (Section 3.1). We then introduce *functorial translations* between equational systems, making them a category, and discuss some basic properties of this category (Sections 3.2 and 3.3).

3.1 Equational Systems

3.1∗1. Generally speaking, an algebraic theory consists of the *signature* and *equations* of its operations. A concise way to specify a signature over a category $\mathscr{C}$ is just a functor $\Sigma : \mathscr{C} \to \mathscr{C}$, and then a Σ-*algebra* is an object $A \in \mathscr{C}$, called the *carrier*, together with a *structure map* $\alpha : \Sigma A \to A$. For example, the signature functor of the *theory of monoids* in a monoidal category $\mathscr{E}$ with binary coproducts is $\Sigma_{\text{Mon}} = (- \square -) + I$. A Σ_{Mon}-algebra $\langle A, \alpha \rangle$ is an object A with a morphism $\alpha : (A \square A) + I \to A$, or equivalently two morphisms $A \square A \to A$ and $I \to A$.

3.1∗2. We denote the category of Σ-algebras by Σ-Alg, whose morphisms from $\langle A, \alpha \rangle$ to $\langle B, \beta \rangle$ are *algebra homomorphisms*, i.e. morphisms $h : A \to B$ in $\mathscr{C}$ such that $h \cdot \alpha = \beta \cdot \Sigma h : \Sigma A \to B$. The forgetful functor dropping the structure map is denoted by $\mathrm{U}_\Sigma : \Sigma\text{-Alg} \to \mathscr{C}$ or just U when it is not ambiguous.

3.1∗3. Equations on a signature $\Sigma : \mathscr{C} \to \mathscr{C}$ are typically presented as commutative diagrams, such as the diagrams in 2∗2 for monoids. Pictorially, a diagram looks like the following:



which contains a pair of paths $L\alpha$ and $R\alpha$ from some formal object ΓA to some formal object A, parameterised by a formal morphism $\alpha : \Sigma A \to A$ of the operation. The starting node ΓA can be called the *context* of the diagram.

A categorical way to make precise such as a diagram is a functor $\Gamma : \mathscr{C} \to \mathscr{C}$ and a pair of functors $L, R : \Sigma\text{-Alg} \to \Gamma\text{-Alg}$. For example, let $\langle \mathscr{E}, \square, I \rangle$ be a monoidal category and $\Sigma = (- \square -) : \mathscr{E} \to \mathscr{E}$ be the signature of a binary operation. To encode the associativity diagram in 2∗2 for Σ-algebras, we let the functor $\Gamma : \mathscr{E} \to \mathscr{E}$ be $(- \square -) \square -$, and the two paths are represented by two functors $L, R : (- \square -)\text{-Alg} \to ((- \square -) \square -)\text{-Alg}$:

$$L\langle A, \mu : A \square A \to A \rangle = \langle A, \mu \cdot (\mu \square A) \rangle$$
$$R\langle A, \mu : A \square A \to A \rangle = \langle A, \mu \cdot (A \square \mu) \cdot \alpha_{A,A,A} \rangle$$

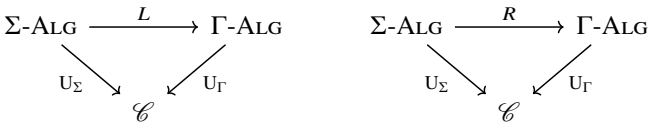
The functors $L, R : \Sigma\text{-Alg} \to \Gamma\text{-Alg}$ must satisfy $\mathrm{U}_\Gamma \circ L = \mathrm{U}_\Sigma$ and $\mathrm{U}_\Gamma \circ R = \mathrm{U}_\Sigma$.

3.1∗4 Definition (Fiore and Hur (2009)). An *equational system* on a category $\mathscr{C}$

$$\dot{\Sigma} := (\Sigma \triangleright \Gamma \vdash L = R)$$

consists of four functors: a *(functorial) signature* $\Sigma : \mathscr{C} \to \mathscr{C}$, a *(functorial) context* $\Gamma : \mathscr{C} \to \mathscr{C}$, and a pair of two *(functorial) terms* $L, R : \Sigma\text{-Alg} \to \Gamma\text{-Alg}$ making the following

diagrams commute:



(3.1)

An *algebra*, or a *model*, of $\dot{\Sigma}$ is a Σ-algebra $\langle A \in \mathscr{C}, \alpha : \Sigma A \to A\rangle$ such that

$$L\langle A, \alpha\rangle = R\langle A, \alpha\rangle \in \Gamma\text{-Alg}.$$

We denote by $\dot{\Sigma}$-Alg the full subcategory of Σ-Alg containing $\dot{\Sigma}$-algebras.

3.1∗5 Notation. In the definition above, the functors $L, R : \Sigma\text{-Alg} \to \Gamma\text{-Alg}$ always send objects $\langle A, \alpha : \Sigma A \to A\rangle$ to $\langle A, \beta : \Gamma A \to A\rangle$, keeping carriers A unchanged, so we will simply write $L\alpha : \Gamma A \to A$ to mean $\pi_2(L\langle A, \alpha\rangle)$.

3.1∗6 Example. Let $\mathscr{E}$ be a monoidal category with binary coproducts. The concept of monoids in $\mathscr{E}$ (2∗2) can be defined as an equational system on $\mathscr{E}$

$$\text{Mon} := (\Sigma_{\text{Mon}} \triangleright \Gamma_{\text{Mon}} \vdash L_{\text{Mon}} = R_{\text{Mon}}) \tag{3.2}$$

with functorial signature and contexts

$$\Sigma_{\text{Mon}}\, M = (M \square M) + I$$
$$\Gamma_{\text{Mon}}\, M = ((M \square M) \square M) + (I \square M) + (M \square I),$$

and $L_{\text{Mon}}, R_{\text{Mon}} : \Sigma_{\text{Mon}}\text{-Alg} \to \Gamma_{\text{Mon}}\text{-Alg}$ given by

$$L_{\text{Mon}}\,\beta = [L_1, L_2, L_3] \qquad R_{\text{Mon}}\,\beta = [R_1, R_2, R_3],$$

where we define $\mu := (\beta \cdot \iota_1) : M \square M \to M$, $\eta := (\beta \cdot \iota_2) : I \to M$ and

$$L_1 := \mu \cdot (\mu \square M) \qquad R_1 := \mu \cdot (M \square \mu) \cdot \alpha_{M,M,M}$$
$$L_2 := \mu \cdot (\eta \square M) \qquad R_2 := \lambda_M$$
$$L_3 := \mu \cdot (M \square \eta) \qquad R_3 := \rho_M$$

Each pair L_i and R_i encodes a commutative diagram for monoids in 2∗2. An algebra of this equational system is precisely a monoid in $\mathscr{E}$.

3.1∗7 Notation. From the example above we see that although the definition of equational systems has exactly one operation and one equation, multiple operations/equations can be encoded via coproducts. We will say *an equational system with a set of operations* $\{\Sigma_o : \mathscr{C} \to \mathscr{C}\}_{o \in O}$ and *a set of equations* $\{\Gamma_e \vdash L_e = R_e\}_{e \in E}$, where $\Gamma_e : \mathscr{C} \to \mathscr{C}$ and $L_e, R_e : (\coprod_{o \in O} \Sigma_o)\text{-Alg} \to \Gamma_e\text{-Alg}$, provided that $\mathscr{C}$ has O-indexed and E-indexed coproducts.

Given an equational system $\dot{\Sigma} = (\Sigma \triangleright \Gamma \vdash L = R)$ on a category $\mathscr{C}$ with binary coproducts, we denote by $\dot{\Sigma} \curlywedge \Sigma'$ the extension of $\dot{\Sigma}$ with a new operations of signature $\Sigma' : \mathscr{C} \to \mathscr{C}$:

$$\dot{\Sigma} \curlywedge \Sigma' := (\Sigma + \Sigma' \triangleright \Gamma \vdash L \circ \mathrm{U} = R \circ \mathrm{U})$$

where $\mathrm{U} : (\Sigma + \Sigma')\text{-Alg} \to \Sigma\text{-Alg}$ is the forgetful functor dropping Σ' operations. Similarly, we denote by $\dot{\Sigma} \curlywedge (\Gamma' \vdash L' = R')$ the extension of $\dot{\Sigma}$ with a new equation:

$$\dot{\Sigma} \curlywedge (\Gamma' \vdash L' = R') := (\Sigma \triangleright \Gamma + \Gamma' \vdash [L, L'] = [R, R']),$$

where $[L, L'] : \Sigma\text{-Alg} \to (\Gamma + \Gamma')\text{-Alg}$ is the functor mapping $\alpha : \Sigma X \to X$ to $[L\alpha, L'\alpha] : (\Gamma + \Gamma')X \to X$, and $[R, R']$ is similar.

3.1∗8 Example. The concept of Eilenberg-Moore algebras of a monad can be defined as an equational system. Let M be a monad on some $\mathscr{C}$ with coproducts. The equational system M-Alg has as signature M itself and two equations: $\mathrm{Id}_{\mathscr{C}} \vdash L_1 = R_1$ and $M \circ M \vdash L_2 = R_2$ where for all $\langle X \in \mathscr{C}, \alpha : MX \to X \rangle$,

$$L_1\alpha = (X \xrightarrow{\eta_X} MX \xrightarrow{\alpha} X) \qquad R_1\alpha = id_X$$
$$L_2\alpha = (M(MX) \xrightarrow{\mu_X} MX \xrightarrow{\alpha} X) \qquad R_2\alpha = (M(MX) \xrightarrow{M\alpha} MX \xrightarrow{\alpha} X)$$

The category of algebras for this equational system is precisely the Eilenberg-Moore category of the monad M.

3.1∗9. Equational systems can also express *inequations* using a standard trick in enriched algebraic theories (Robinson, 2002). Let $\mathscr{C}$ be a category with an enrichment over Poset, which means that every hom-set $\mathscr{C}(A, B)$ is equipped with a partial order and composition of $\mathscr{C}$-morphisms $\mathscr{C}(A, B) \times \mathscr{C}(B, C) \to \mathscr{C}(A, C)$ is monotone, such that $\mathscr{C}$ is *copowered* (also called *tensored*) over Poset, which means that for every $A \in \mathscr{C}$ and $P \in \text{Poset}$, there is an object $P \cdot A \in \mathscr{C}$ and a natural isomorphism between posets:

$$\mathscr{C}(P \cdot A, B) \;\cong\; \text{Poset}(P, \mathscr{C}(A, B)).$$

Let $\mathfrak{S}$ be the two-element poset $\{\bot \sqsubseteq \top\}$. For every endofunctor $\Sigma : \mathscr{C} \to \mathscr{C}$ (not necessarily a Poset-enriched functor), an operation $\alpha : \mathfrak{S} \cdot \Sigma A \to A$ is then two operations $\alpha_\bot, \alpha_\top : \Sigma A \to A$ such that $\alpha_\bot \sqsubseteq \alpha_\top$ in the poset $\mathscr{C}(\Sigma A, A)$. Therefore in this case we can impose orders on operations of an equational system. Moreover, we can encode an inequational axiom $l(x) \sqsubseteq r(x)$ by introducing two operations $\bar{l} \sqsubseteq \bar{r}$ and equations stating that $l(x) = \bar{l}(x)$ and $r(x) = \bar{r}(x)$. This trick can also be generalised to Cat-enriched categories, i.e. 2-categories, to express lax-equations or pseudo-equations.

3.1∗10 Example. The category Poset is enriched over itself with the pointwise order on $\text{Poset}(A, B)$, and it is tensored with $P \cdot A$ given by cartesian product $P \times A$. The equational system Bot over the category Poset expresses posets with a bottom element: it has the signature functor $\Sigma_{\text{Bot}} := 1 + (\mathfrak{S} \cdot -)$ and two equations $\{\mathrm{Id} \vdash L_i = R_i\}_{i=1,2}$ where for every $\alpha : 1 + \mathfrak{S} \cdot A \to A$ the functors L_i and R_i are given by

$$L_1\alpha = (A \xrightarrow{\iota_\bot} \mathfrak{S} \cdot A \xrightarrow{\alpha \cdot \iota_2} A) \qquad R_1\alpha = (A \to 1 \xrightarrow{\alpha \cdot \iota_1} A)$$
$$L_2\alpha = (A \xrightarrow{\iota_\top} \mathfrak{S} \cdot A \xrightarrow{\alpha \cdot \iota_2} A) \qquad R_2\alpha = (A \xrightarrow{id_A} A)$$

An algebra of Bot is a preorder $\langle A, \sqsubseteq \rangle$ with an element $b \in A$ and two monotone functions $\bar{l} : A \to A$, and $\bar{r} : A \to A$ such that $\bar{l} \sqsubseteq \bar{r}$. The two equations of Bot state that $\bar{l}(x) = b$ and $\bar{r}(x) = x$ for all $x \in A$, so an algebra of Bot is exactly a preorder with a bottom element b.

3.1∗11. Among the algebras of an equational system $\dot{\Sigma}$ over $\mathscr{C}$, the *free algebras* are particularly useful since they represent *abstract syntax* of terms built from variables and operations of the theory. The abstract syntax can be *interpreted* with another model using the free-forgetful adjunction:

$$\phi : \mathscr{C}(X, A) \;\cong\; \dot{\Sigma}\text{-Alg}(\mathrm{F}\, X, \langle A, \alpha \rangle)$$

Given any model $\langle A, \alpha\rangle$ of $\dot{\Sigma}$ and $g : X \to A$, the morphism $\phi(g) : \mathrm{F}\, X \to \langle A, \alpha\rangle$ interprets the free algebra with the semantic model $\langle A, \alpha\rangle$, with generators X interpreted by $g : X \to A$. Fiore and Hur (2009) showed several sufficient conditions for the existence of free algebras, which we record below.

3.1∗12 Theorem (Fiore and Hur (2009)). *Let $\dot{\Sigma} = (\Sigma \triangleright \Gamma \vdash L = R)$ be an equational system over $\mathscr{C}$. If $\mathscr{C}$ is (small-) cocomplete and one of the following holds*:

- *Σ and Γ preserve colimits of α-chains for a limit ordinal α;*
- *Σ preserves colimits of α-chains for a limit ordinal α, and both Σ and Γ preserve epimorphisms in $\mathscr{C}$;*
- *Σ preserves colimits of α-chains for a limit ordinal α, and Σ preserves epimorphisms, and $\mathscr{C}$ has no transfinite chains of proper epimorphisms,*

then there are left adjoints to the inclusion functor $\dot{\Sigma}$-Alg $\hookrightarrow \Sigma$-Alg and the forgetful functor Σ-Alg $\to \mathscr{C}$:

$$\dot{\Sigma}\text{-Alg} \underset{\hookrightarrow}{\overset{\longleftarrow}{\bot}} \Sigma\text{-Alg} \underset{}{\overset{\longleftarrow}{\bot}} \mathscr{C} \tag{3.3}$$

Moreover, $\dot{\Sigma}$-Alg is cocomplete and the composite of the adjunction is monadic.

3.1∗13 Notation. We denote the composite adjunction of (3.3) by $\mathrm{F}_{\dot{\Sigma}} \dashv \mathrm{U}_{\dot{\Sigma}}$, or $\mathrm{F} \dashv \mathrm{U}$ when $\dot{\Sigma}$ is clear from context. Moreover, the initial $\dot{\Sigma}$-algebra is denoted by $\langle \mu\dot{\Sigma}, \alpha^{\dot{\Sigma}} : \Sigma\mu\dot{\Sigma} \to \mu\dot{\Sigma}\rangle$.

3.1∗14. Fiore and Hur's proof of this result is quite technical, but we will not rely on the specifics of their construction. For concreteness, we provide some basic intuition here: the free Σ-algebra on some $A \in \mathscr{C}$ is first constructed by a transfinite iteration of $A + \Sigma -$ on 0

$$0 \xrightarrow{\ !\ } A + \Sigma 0 \xrightarrow{A+\Sigma !} A + \Sigma(A + \Sigma 0) \longrightarrow \cdots$$

and taking colimits for limit ordinals. The iteration will stop at some $X \cong A + \Sigma X$ in α steps, giving the carrier of the free Σ-algebra (Adámek, 1974). Then it is quotiented by the equation $L = R$ and the congruence rule, using Fiore and Hur's *algebraic coequalisers*. The quotienting may also need to be repeated α times when Σ or Γ does not preserve epimorphisms. The result of quotienting is the free $\dot{\Sigma}$-algebra.

3.1∗15 Example. Let $\mathscr{E}$ be a small-cocomplete monoidal category such that $\square : \mathscr{E} \times \mathscr{E} \to \mathscr{E}$ preserves α-chains for some limit ordinal α. For example, $\mathscr{E}$ can be $\langle \textsc{Endo}_\kappa(\mathscr{C}), \circ, \mathrm{Id}\rangle$ for an lκp $\mathscr{C}$ from Section 2.2 or $\langle \textsc{Endo}_\kappa(\textsc{Set}), *, \mathrm{Id}\rangle$ from Section 2.6. Then the first condition of Theorem 3.1∗12 is applicable to the equational system Mon from Example 3.1∗6, so every $A \in \mathscr{E}$ has a free monoid.

Moreover, Fiore and Hur (2009) showed that when $\mathscr{E}$ is left closed, there is a simple formula for free monoids: for every $A \in \mathscr{E}$, the free monoid over A is the initial algebra $\mu X.\, I + A \square X$ (sometimes called the *list object* for A) equipped with appropriate monoid operations. This formula is useful in practice: when $\mathscr{E}$ is $\langle \textsc{Set}, \times, 1\rangle$, it is exactly the usual definition $\mu X.\, 1 + A \times X$ of lists of A-elements; and when $\mathscr{E}$ is $\langle \textsc{Endo}_\kappa(\mathscr{C}), \circ, \mathrm{Id}\rangle$ or $\langle \textsc{Endo}_\kappa(\textsc{Set}), *, \mathrm{Id}\rangle$, this gives formulas for free monads and free applicatives that are suitable for implementation (Rivas and Jaskelioff, 2017). Note that this formula needs closedness to work: to define the monoid multiplication

$$(\mu X.\, I + A \square X) \square (\mu X.\, I + A \square X) \to (\mu X.\, I + A \square X),$$

we need closedness to shift one $(\mu X.\, I + A \,\square\, X)$ to the right-hand side

$$(\mu X.\, I + A \,\square\, X) \to (\mu X.\, I + A \,\square\, X)/(\mu X.\, I + A \,\square\, X)$$

and then we can use the universal property of the initial algebra.

3.1∗16 Remark. When $\mathscr{E}$ is not closed, the initial algebra $\mu X.\, 1 + A \,\square\, X$ may not be the free monoid over A. For a counterexample, let $\mathscr{E}$ be the cartesian monoidal category $\langle \textsc{Mon}, \times, 1\rangle$ of monoids in $\langle \textsc{Set}, \times, 1\rangle$. By the Eckmann-Hilton argument, a monoid object in the monoidal category $\langle \textsc{Mon}, \times, 1\rangle$ is precisely an ordinary commutative monoid in sets (so the *structure* of a monoid object degenerates into a *property* in this case). The free monoid object over some $M \in \textsc{Mon}$ is then obtained by quotienting M with commutativity.

On the other hand, the initial algebra $\mu X.\, 1 + M \times X$ in $\textsc{Mon}$ is the monoid whose elements are finite lists $\langle m_1, m_2, \ldots, m_n\rangle$ of M-elements, considered up to trailing zeros, by which we mean $\langle m_1, m_2, \ldots, m_n\rangle$ and $\langle m_1, m_2, \ldots, m_n, e, \ldots, e\rangle$ are considered the same, where e is the unit element of M. Its unit element is the empty sequence $\langle\rangle$, and its multiplication is the *pairwise* multiplication using the multiplication of M (after padding with enough trailing zeros):

$$\langle m_1, m_2, \ldots, m_n\rangle \cdot \langle m'_1, m'_2, \ldots, m'_n\rangle = \langle m_1 \cdot_M m'_1, \ldots, m_n \cdot_M m'_n\rangle. \tag{3.4}$$

The associated $(1 + M \times -)$-algebra is still given by the empty list and *cons* as usual. Now we observe that the multiplication (3.4) is not commutative when M is not commutative, so the initial algebra $\mu X.\, 1 + M \times X$ may not be a monoid object in $\langle \textsc{Mon}, \times, 1\rangle$, let alone the free monoid object over M.

3.1∗17. An appealing aspect of Theorem 3.1∗12 is that the base category $\mathscr{C}$ is only required to be cocomplete rather than locally κ-presentable. Therefore the theorem applies to the category $\textsc{dCpo}$ of *directed complete partial orders*, generalising the construction of free dcpo-algebras for operations of finite arities and (in)equations (Abramsky and Jung, 1995, Theorem 6.1.2). Allowing an endofunctor $\Sigma : \textsc{dCpo} \to \textsc{dCpo}$ as the signature comes in handy: in particular, it allows us to express operations that takes an ascending chain $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \cdots$ as arguments by choosing $\Sigma := (-)^\omega$ where $\omega := \{0 \sqsubseteq 1 \sqsubseteq 2 \sqsubseteq \cdots\}$. This functor preserves colimits of all Ω-chains, where Ω is the first uncountable ordinal.

3.1∗18. We do not know if any of the sufficient conditions for the existence of free algebras given by Theorem 3.1∗12 (or possibly some of their classically equivalent conditions) is constructively true, and in particular, whether they are applicable to small-complete small categories in realizability toposes (2.1∗4). Anyway, we have an alternative constructive proof for small-complete small categories using *impredicative encodings* (Awodey et al., 2018), a refinement of the well known *Church encoding* of inductive datatypes.

Remarkably, the theorem below imposes no constraints on the functorial signature Σ and context Γ, generalising the result that every endofunctor on a small-complete small category has an initial algebra (Hyland, 1988, §3.1).

3.1∗19 Theorem. *Let $\mathscr{C}$ be a small-complete small category. For every equational system $\dot{\Sigma} = (\Sigma \triangleright \Gamma \vdash L = R)$ over $\mathscr{C}$, there is a monadic adjunction $\mathrm{F}_{\dot{\Sigma}} \dashv \mathrm{U}_{\dot{\Sigma}}$. Moreover, the category $\dot{\Sigma}$-$\textsc{Alg}$ is a small-complete-and-cocomplete small category.*

Proof. An easy proof goes by first noticing that $\dot{\Sigma}$-Alg is also a small-complete small category for all equational systems $\dot{\Sigma}$-Alg over $\mathscr{C}$, and then the free algebra of $\dot{\Sigma}$ over an object $X \in \mathscr{C}$ can be constructed from the initial algebra of $\dot{\Sigma} \uparrow \mathrm{K}_X$, where $\mathrm{K}_X : \mathscr{C} \to \mathscr{C}$ is the constant functor mapping to X. The initial algebra of $\dot{\Sigma} \uparrow \mathrm{K}_X$ can be then constructed as the limit of the identity functor $\mathrm{Id} : (\dot{\Sigma} \uparrow \mathrm{K}_X)\text{-Alg} \to (\dot{\Sigma} \uparrow \mathrm{K}_X)\text{-Alg}$ (Mac Lane, 1998, §X.1 Lemma 1), which exists since $(\dot{\Sigma} \uparrow \mathrm{K}_X)$-Alg is a small-complete small category. □

3.1∗20. Theorem 3.1∗12 and Theorem 3.1∗19 give different sufficient conditions for the existences of free algebras of equational systems, and there are some other constructive techniques such as the one by Fiore et al. (2022). In this paper, we only rely on the existence of free algebras but not those specific conditions guaranteeing the existence of free algebras. The following definition will be used for abstracting over those different conditions.

3.1∗21 Definition. Let $\mathscr{C}$ be a category. A relation $\mathscr{A} \subseteq \mathrm{Endo}(\mathscr{C}) \times \mathrm{Endo}(\mathscr{C})$ of endofunctors is called a *freeness condition* if every equational system $\Sigma \triangleright \Gamma \vdash L = R$ with $\langle \Sigma, \Gamma \rangle \in \mathscr{A}$ has the free-forgetful adjunction.

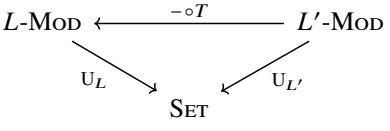
We denote by $\mathrm{Eqs}_{\mathscr{A}}(\mathscr{C})$ the full subcategory of $\mathrm{Eqs}(\mathscr{C})$ containing equational systems whose functorial signature and context are in $\mathscr{A}$.

3.1∗22 Example. (1) If $\mathscr{C}$ is lκp then the relation $\mathrm{Endo}_\kappa(\mathscr{C}) \times \mathrm{Endo}_\kappa(\mathscr{C})$ containing all pairs of κ-accessible endofunctors (Section 2.2) is a freeness condition by the first item of Theorem 3.1∗12. (2) If $\mathscr{C}$ is small-complete and small, the entire relation $\mathrm{Endo}(\mathscr{C}) \times \mathrm{Endo}(\mathscr{C})$ is a freeness condition (2.1∗3) by Theorem 3.1∗19. (3) For every $\mathscr{C}$, there is always the largest freeness condition $\mathscr{F}$ containing all pairs $\langle \Sigma, \Gamma \rangle$ such that all equational systems with signature Σ and context Γ have the free-forgetful adjunction. However, $\mathscr{F}$ is less useful than it may appear: we may not know whether $\mathscr{F}$ has good closure properties, for example, whether $\langle \Sigma + \Sigma', \Gamma + \Gamma' \rangle$ is in $\mathscr{F}$ when $\langle \Sigma, \Gamma \rangle$ and $\langle \Sigma', \Gamma' \rangle$ are in $\mathscr{F}$.

3.2 Functorial Translations

3.2∗1. Morphisms between equational systems are not studied by Fiore and Hur (2007, 2009), but we need them later for talking about combinations of equational systems. A natural idea for morphisms from an equational system $\dot{\Sigma}$ to another $\dot{\Psi}$ is *translations*, which map operations in $\dot{\Sigma}$ to *terms* of $\dot{\Psi}$, preserving equations in a suitable sense. However, a technical difficulty is that equational systems $\dot{\Psi}$ may not have terms, i.e. free algebras.

In the following, we avoid this by introducing a more indirect definition which we call *functorial translations* between equational systems. For some motivation, consider Lawvere theories: every morphism $T : L \to L'$ between Lawvere theories induces a functor $- \circ T : L'\text{-Mod} \to L\text{-Mod}$ between their category of models from the opposite direction. Moreover this functor commutes with the forgetful functors from models of L and L' to sets:



This mapping from morphisms T between Lawvere theories to functors $- \circ T$ between the categories of models (from the opposite direction) that commute with U_L and $\mathrm{U}_{L'}$ is in fact

an equivalence of categories (Adamek et al., 2010, 11.38). Mimicking this situation, we will define morphisms between equational systems also as functors between their categories of models from the opposite direction that commute with the forgetful functors.

From a more informal viewpoint, this is justified by that if we can 'translate' operations of an equational system Σ to terms of operations of another equational system Σ', then given a model of Σ', we can construct a model of Σ.

3.2∗2. In the rest of this section, we fix a category $\mathscr{C}$ and a freeness condition $\mathscr{A} \subseteq \textsc{Endo}(\mathscr{C}) \times \textsc{Endo}(\mathscr{C})$ of endofunctors (Definition 3.1∗21).

3.2∗3 Definition. A *functorial translation* of equational systems on $\mathscr{C}$ from $\dot{\Sigma} = (\Sigma \triangleright \Gamma \vdash L = R)$ to $\dot{\Sigma}' = (\Sigma' \triangleright \Gamma' \vdash L' = R')$ is a functor $T : \dot{\Sigma}'\text{-}\textsc{Alg} \to \dot{\Sigma}\text{-}\textsc{Alg}$ such that $U_{\dot{\Sigma}} \circ T = U_{\dot{\Sigma}'}$, where $U_{\dot{\Sigma}} : \dot{\Sigma}\text{-}\textsc{Alg} \to \mathscr{C}$ and $U_{\dot{\Sigma}'} : \dot{\Sigma}'\text{-}\textsc{Alg} \to \mathscr{C}$ are the forgetful functors. Equational systems on $\mathscr{C}$ and translations form a category $\textsc{Eqs}(\mathscr{C})$ whose identity morphisms are the identity functors $\dot{\Sigma}\text{-}\textsc{Alg} \to \dot{\Sigma}\text{-}\textsc{Alg}$ and composition of translations $T \circ T'$ is functor composition.

3.2∗4 Example. Let $\mathscr{C}$ be a category with finite coproducts and products. The theory $\textsc{Grp}$ of groups over $\mathscr{C}$ is the theory $\textsc{Mon}$ of monoids in $\langle \mathscr{C}, \times, 1\rangle$ from Example 3.1∗6 extended with a new operation i with signature $\mathrm{Id} : \mathscr{C} \to \mathscr{C}$ and a new equation $\mathrm{Id} \vdash L = R$ where

$$L\langle M, \eta, \mu, i\rangle = \langle M, M \xrightarrow{\langle id_X, i\rangle} M \times M \xrightarrow{\mu} M\rangle$$
$$R\langle M, \eta, \mu, i\rangle = \langle M, M \xrightarrow{!} 1 \xrightarrow{\eta} M\rangle$$

Then there is a translation $T : \textsc{Mon} \to \textsc{Grp}$ that maps every $\langle M, \eta, \mu, i\rangle$ in $\textsc{Grp}\text{-}\textsc{Alg}$ to an object $\langle M, \eta, \mu\rangle$ in $\textsc{Mon}\text{-}\textsc{Alg}$ by forgetting the newly added operation. We call translations like $T : \textsc{Mon} \to \textsc{Grp}$ that simply forget some operations and equations *inclusion translations*.

3.2∗5. For equational systems with free-algebras, the following lemma shows that functorial translations coincide with the expected notion of translations: maps between monads.

3.2∗6 Lemma. Let $\dot{\Sigma}, \dot{\Psi} \in \textsc{Eqs}_{\mathscr{A}}(\mathscr{C})$. There is a bijection between functorial translations $T : \dot{\Sigma} \to \dot{\Psi}$ and monad morphisms $m : U_{\dot{\Sigma}}F_{\dot{\Sigma}} \to U_{\dot{\Psi}}F_{\dot{\Psi}}$.

Proof. By Fiore and Hur (2009, Proposition 6.4), the free-forgetful adjunction for an equational system is always monadic, so functorial translations $T : \dot{\Sigma} \to \dot{\Psi}$, i.e. functors $T : \dot{\Psi}\text{-}\textsc{Alg} \to \dot{\Sigma}\text{-}\textsc{Alg}$ such that $U_{\dot{\Sigma}} \circ T = U_{\dot{\Psi}}$, are in bijection with functors $S : \mathscr{C}^{U_{\dot{\Psi}}F_{\dot{\Psi}}} \to \mathscr{C}^{U_{\dot{\Sigma}}F_{\dot{\Sigma}}}$ between the corresponding Eilenberg-Moore categories that commute with the forgetful functors. By Borceux (1994, Proposition 4.5.9), those functors S are in bijection with monad morphisms $m : U_{\dot{\Sigma}}F_{\dot{\Sigma}} \to U_{\dot{\Psi}}F_{\dot{\Psi}}$. □

3.2∗7 Corollary. The two functorial terms $L, R : \Sigma\text{-}\textsc{Alg} \to \Gamma\text{-}\textsc{Alg}$ of an equational system can also be viewed as translations between the equational systems of signatures Γ and Σ without equations. Therefore when Σ and Γ have free algebras, L and R are in bijection with two monad morphisms $U_\Gamma F_\Gamma \to U_\Sigma F_\Sigma$. When $\mathscr{C}$ is locally small and small-complete, the monad $U_\Gamma F_\Gamma$ for free Γ-algebras is also the free monad over the endofunctor Γ (nLab, 2024, Theorem 3.2). In this case, monad morphisms $U_\Gamma F_\Gamma \to U_\Sigma F_\Sigma$ are in bijection with natural transformations $\Gamma \to U_\Sigma F_\Sigma$. Moreover, it can be checked that a Σ-algebra $\alpha : \Sigma A \to A$

satisfies a functorial equation $\Gamma \vdash L = R$ if and only if the following commutes:

$$\Gamma A \underset{\tilde{R}_A}{\overset{\tilde{L}_A}{\rightrightarrows}} \mathrm{U}_\Sigma \mathrm{F}_\Sigma A \xrightarrow{\mathrm{U}_\Sigma \epsilon_{\langle A,\alpha\rangle}} A$$

where $\tilde{L}$ and $\tilde{R}$ are the natural transformations corresponding to L and R.

3.2∗8 Theorem. *Let $\mathscr{C}$ be a category with binary coproducts and $\mathrm{Eqs}_f(\mathscr{C}) \subseteq \mathrm{Eqs}(\mathscr{C})$ be the full subcategory containing all equational systems admitting the free-forgetful adjunction. Then we have an equivalence of categories*

$$\mathrm{Eqs}_f(\mathscr{C}) \quad \cong \quad \mathrm{Mon}(\mathscr{C}),$$

where $\mathrm{Mon}(\mathscr{C})$ is the category of monads over $\mathscr{C}$ and monad morphisms.

Proof. Example 3.1∗8 shows that every monad M induces an equational system M-Alg whose category of algebras is precisely the Eilenberg-Moore category of M. Thus the equational system M-Alg is in $\mathrm{Eqs}_f(\mathscr{C})$ since free Eilenberg-Moore algebras always exist (which are simply $\langle X, \mu_X : M(MX) \to MX\rangle$ for all $X \in \mathscr{C}$). By Lemma 3.2∗6 above, this construction extends to a fully faithful functor $\mathrm{Mon}(\mathscr{C}) \to \mathrm{Eqs}_f(\mathscr{C})$. Moreover, the forgetful functor for every equational system $\dot{\Sigma} \in \mathrm{Eqs}_f(\mathscr{C})$ is monadic (Fiore and Hur, 2009, Proposition 6.4), so $\dot{\Sigma}$ is isomorphic to the equational system $(\mathrm{F}_{\dot{\Sigma}}\mathrm{U}_{\dot{\Sigma}})$-Alg. Therefore we have an essentially surjective fully faithful functor $\mathrm{Mon}(\mathscr{C}) \to \mathrm{Eqs}_f(\mathscr{C})$, and thus an equivalence $\mathrm{Mon}(\mathscr{C}) \cong \mathrm{Eqs}_f(\mathscr{C})$. □

3.2∗9. The theorem above shows that equational systems subsume not just *monads with ranks* but *all monads*. Two natural questions are then

- Is the category $\mathrm{Eqs}(\mathscr{C})$ some kind of completion of $\mathrm{Mon}(\mathscr{C})$?
- Given a functor $\mathrm{U} : \mathscr{D} \to \mathscr{C}$, under what conditions U is the forgetful functor for an equational system on $\mathscr{C}$?

We leave answering these questions as future work.

3.2∗10. Below, we turn our attention to the property of translations $T : \dot{\Sigma} \to \dot{\Psi}$ as functors $T : \dot{\Psi}\text{-Alg} \to \dot{\Sigma}\text{-Alg}$, and show a condition for T to have left adjoints. We start with an interesting lemma saying that (when $\dot{\Psi}$ has free algebras) translations $T : \dot{\Sigma} \to \dot{\Psi}$ can be extended in a canonical way to translations $\tilde{T} : \Sigma \to \Psi$ between the respective equational systems *without equations*. Note, however, such an extension may not be unique: when the equation of $\dot{\Psi}$ is inconsistent in the sense that $\dot{\Psi}$ only has the trivial model 1, there is a unique trivial T, but there can still be different translations $\Sigma \to \Psi$.

3.2∗11 Lemma. Let $\mathscr{C}$ be a category and $\dot{\Sigma}, \dot{\Psi} \in \mathrm{Eqs}(\mathscr{C})$ such that $\dot{\Psi}$ has the free-forgetful adjunction. Every functorial translation $\dot{\Sigma} \to \dot{\Psi}$, i.e. a functor $T : \dot{\Psi}\text{-Alg} \to \dot{\Sigma}\text{-Alg}$ such that $\mathrm{U}_{\dot{\Sigma}} \circ T = \mathrm{U}_{\dot{\Psi}}$ can be extended to a functor $\tilde{T} : \Psi\text{-Alg} \to \Sigma\text{-Alg}$ such that $\mathrm{U}_\Sigma \circ T = \mathrm{U}_\Psi$, and $\tilde{T}$ coincides with T on $\dot{\Psi}$-Alg.

Proof. Given any Ψ-algebra $\langle A, \alpha : \Psi A \to A\rangle$, there is a free $\dot{\Psi}$-algebra over A with structure map $o : \Psi(\mathrm{F}_{\dot{\Psi}}A) \to \mathrm{F}_{\dot{\Psi}}A$. It is mapped by T to a $\dot{\Sigma}$-algebra $To : \Sigma(\mathrm{F}_{\dot{\Psi}}A) \to \mathrm{F}_{\dot{\Psi}}A$. We now

define the functor $\tilde{T} : \Psi\text{-ALG} \to \Sigma\text{-ALG}$ by

$$\tilde{T}\langle A, \alpha\rangle = \langle A,\ \Sigma A \xrightarrow{\Sigma\eta_A} \Sigma(\mathrm{F}_{\dot\Psi} A) \xrightarrow{T o} \mathrm{F}_{\dot\Psi} A \xrightarrow{\epsilon_{A,\alpha}} A\rangle.$$

To see that $\tilde{T}$ restricts to T on $\dot\Psi$-ALG, supposing $\langle A, \alpha\rangle \in \dot\Psi$, T maps the counit $\epsilon_{A,\alpha} : \langle \mathrm{F}_{\dot\Psi} A, o\rangle \to \langle A, \alpha\rangle$ to the following commutative diagram:

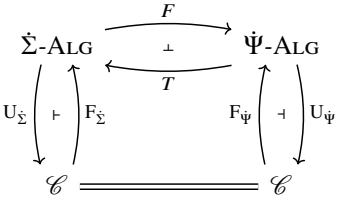
$$\begin{array}{ccc} \Sigma(\mathrm{F}_{\dot\Psi} A) & \xrightarrow{T o} & \mathrm{F}_{\dot\Psi} A \\ {\scriptstyle \Sigma\epsilon_{A,\alpha}}\downarrow & & \downarrow{\scriptstyle \epsilon_{A,\alpha}} \\ \Sigma A & \xrightarrow[T\alpha]{} & A \end{array}$$

Therefore $\tilde{T}\alpha = \epsilon_{A,\alpha} \cdot T o \cdot \Sigma\eta_A = T\alpha \cdot \Sigma\epsilon \cdot \Sigma\eta_A = T\alpha$ since $\epsilon_{A,\alpha} \cdot \eta_A = id$. □

3.2∗12. In the adjunction $\mathrm{F}_{\dot\Sigma} \dashv \mathrm{U}_{\dot\Sigma} : \dot\Sigma\text{-ALG} \to \mathscr{C}$ for free algebras, the category $\mathscr{C}$ can be viewed as the category $\emptyset$-ALG for the empty theory $\emptyset$ with no operations or equations, and $\mathrm{U}_{\dot\Sigma} : \dot\Sigma\text{-ALG} \to \emptyset\text{-ALG}$ is the unique translation from $\emptyset$ to $\dot\Sigma$. The fact that $\mathrm{U}_{\dot\Sigma}$ always has a left adjoint can be generalised to any functorial translations, giving us *relative free algebras*.

3.2∗13 Definition. Letting J be a set, we say that the freeness condition $\mathscr{A}$ is *closed under J-indexed coproducts* if whenever $\langle \Sigma_j, \Gamma_j\rangle \in \mathscr{A}$ for all $j \in J$ then $\coprod_j \Sigma_j$ and $\coprod_j \Gamma_j$ exist and are related by $\mathscr{A}$. Similarly, $\mathscr{A}$ is said to be *closed under constant functors* if $\langle \mathrm{K}_A, \mathrm{K}_B\rangle \in \mathscr{A}$ for all $A, B \in \mathscr{C}$.

3.2∗14 Theorem. *Assuming that $\mathscr{A}$ is closed under binary coproducts and constant functors, every functorial translation $T : \dot\Sigma \to \dot\Psi$ in $\mathrm{EQS}_{\mathscr{A}}(\mathscr{C})$ as a functor $T : \dot\Psi\text{-ALG} \to \dot\Sigma\text{-ALG}$ has a left adjoint F and the adjunction is monadic*:



Proof. The idea of the proof is to construct the left adjoint F via the initial algebra of some other equational system, similarly to how the free algebra for a functor $\Sigma : \mathscr{C} \to \mathscr{C}$ over an object $A \in \mathscr{C}$ can be constructed via the initial algebra of the functor $A + \Sigma -$, except that we need to take equations into account.

Let $\dot\Sigma = (\Sigma \triangleright \Gamma \vdash L = R)$. To construct the free $\dot\Psi$-algebra over a $\dot\Sigma$-algebra $\langle A, \alpha : \Sigma A \to A\rangle$, we consider the equational system

$$\dot\Psi_A := \dot\Psi \pitchfork \mathrm{K}_A \pitchfork (\mathrm{K}_{\Sigma A} \vdash L' = R') \tag{3.5}$$

where K_X is the constant endofunctor mapping to $X \in \mathscr{C}$, and the two functorial terms $L', R' : (\Psi + \mathrm{K}_A)\text{-ALG} \to \mathrm{K}_{\Sigma A}\text{-ALG}$ are

$$\begin{aligned} L'\langle B, \beta : \Psi B \to B, i : A \to B\rangle &= \langle B, \Sigma A \xrightarrow{\Sigma i} \Sigma B \xrightarrow{\tilde{T}\beta} B\rangle, \\ R'\langle B, \beta : \Psi B \to B, i : A \to B\rangle &= \langle B, \Sigma A \xrightarrow{\alpha} A \xrightarrow{i} B\rangle. \end{aligned}$$

where $\tilde{T} : \Psi\text{-Alg} \to \Sigma\text{-Alg}$ is obtained from T by Lemma 3.2∗11. Since $\mathscr{A}$ is assumed to closed under binary coproducts and constant functors, the equational system $\dot{\Psi}_A$ is in $\mathrm{Eqs}_{\mathscr{A}}(\mathscr{C})$ and thus has an initial algebra

$$\langle B_0, \beta : \Psi B_0 \to B_0, i : A \to B_0 \rangle.$$

We note that the functorial equation $L' = R'$ encodes a Σ-algebra homomorphism:

$$\begin{array}{ccc} \Sigma A & \xrightarrow{\Sigma i} & \Sigma B \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\tilde{T}\beta} \\ A & \xrightarrow{i} & B \end{array} \tag{3.6}$$

Since $\tilde{T}$ and T coincide on $\dot{\Psi}$ algebras, $\tilde{T}\beta$ is the same as $T\beta$. Therefore $i : A \to B_0$ is a $\dot{\Sigma}$-algebra homomorphism from $\langle A, \alpha \rangle$ to $\langle B_0, T\beta \rangle$.

Next we show that the arrow $i : \langle A, \alpha \rangle \to T\langle B_0, \beta \rangle$ is a universal arrow from $\langle A, \alpha \rangle$ to the functor $T : \dot{\Psi}\text{-Alg} \to \dot{\Sigma}\text{-Alg}$. For every $\langle C, \delta \rangle \in \dot{\Psi}\text{-Alg}$ and an arrow $f : \langle A, \alpha \rangle \to \langle C, T\delta \rangle$, we need to find a unique $\dot{\Psi}$-homomorphism $h : \langle B_0, \beta \rangle \to \langle C, \delta \rangle$ such that $Th \cdot i = f$:

$$\begin{array}{ccc} \langle A, \alpha \rangle & \xrightarrow{\ i\ } & T\langle B_0, \beta \rangle \\ & {\scriptstyle f}\searrow & \downarrow{\scriptstyle Th} \\ & & T\langle C, \delta \rangle \end{array} \qquad\qquad \begin{array}{c} \langle B_0, \beta \rangle \\ \vdots \\ \downarrow{\scriptstyle \exists! h} \\ \langle C, \delta \rangle \end{array}$$

We observe that $\langle C, \delta, f \rangle$ is a model of $\dot{\Psi}_A$ (3.5), so by the initiality of $\langle B_0, \beta, i \rangle$, there is an $h : \langle B_0, \beta, i \rangle \to \langle C, \delta, f \rangle$. Since h is a $\dot{\Psi}_A$-homomorphism, we have $h \cdot i = f$. Hence $Th \cdot i = f$ as the translation T preserves homomorphisms.

It remains to show the uniqueness of such $h : \langle B_0, \beta \rangle \to \langle C, \delta \rangle \in \dot{\Psi}\text{-Alg}$ with $h \cdot i = f$. Assuming there is such an h', then h' is also a $\dot{\Psi}_A$-homomorphism from $\langle B_0, \beta, i \rangle$ to $\langle C, \delta, i \rangle$. Therefore $h' = h$ by the initiality of $\langle B_0, \beta, i \rangle$.

We have shown that for every $\dot{\Sigma}$-algebra $\langle A, \alpha \rangle$, there is a universal arrow from $\langle A, \alpha \rangle$ to the functor $T : \dot{\Psi}\text{-Alg} \to \dot{\Sigma}\text{-Alg}$. This extends to an adjunction $F \dashv T$ uniquely by (Mac Lane, 1998, §IV.1 Theorem 2).

For the monadicity of the adjunction $F \dashv T$, by Beck's monadicity theorem (Borceux, 1994, Theorem 4.4.4), we need to show that (1) the functor T reflects isomorphisms and (2) for a pair of morphisms $h, g : \langle A, \alpha \rangle \to \langle B, \beta \rangle \in \dot{\Psi}\text{-Alg}$ such that Th and Tg have a split coequaliser in $\dot{\Sigma}\text{-Alg}$, h and g have a coequaliser in $\dot{\Psi}\text{-Alg}$ and that is preserved by T.

For (1), the image of a morphism $h : \langle A, \alpha \rangle \to \langle B, \beta \rangle \in \dot{\Psi}\text{-Alg}$ under the translation functor T is still h (as a $\dot{\Sigma}$-homomorphism). If h has an inverse h^{-1} in $\dot{\Sigma}\text{-Alg}$, then h^{-1} is a $\dot{\Psi}$-homomorphism as well:

$$h \cdot \alpha = \beta \cdot \Psi h \iff h \cdot \alpha \cdot \Psi h^{-1} = \beta \iff \alpha \cdot \Psi h^{-1} = h^{-1} \cdot \beta$$

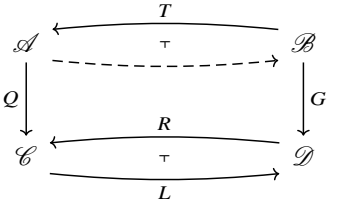
So h is also an isomorphism in $\dot{\Psi}\text{-Alg}$.

For (2), let $e : \langle B, T\beta \rangle \to \langle C, \gamma \rangle$ be the split coequaliser of Th and Tg. By the definition of split coequalisers, $\mathrm{U}_{\dot{\Sigma}} e$ is a split coequaliser of $h, g : A \to B$ in $\mathscr{C}$. By the monadicity of $\mathrm{F}_{\dot{\Psi}} \dashv \mathrm{U}_{\dot{\Psi}}$, there is a coequaliser $e' : \langle B, \beta \rangle \to \langle C', \gamma' \rangle$ of h and g in $\dot{\Psi}\text{-Alg}$ that is preserved by $\mathrm{U}_{\dot{\Psi}}$. Since $Th \cdot Te' = Tg \cdot Te'$, there is a morphism $i : \langle C, \gamma \rangle \to \langle C', T\gamma' \rangle$ such that $i \cdot e = Te'$ in $\dot{\Sigma}\text{-Alg}$. But in the category $\mathscr{C}$, both $\mathrm{U}_{\Sigma} e : B \to C$ and $\mathrm{U}_{\Gamma} e' = \mathrm{U}_{\Sigma} Te : B \to C'$

are coequalisers of $h, g : A \to B$, so $\mathrm{U}_\Sigma i$ is the unique isomorphism between C and C'. As a monadic functor, U_Σ reflects isomorphisms, so i is also an isomorphism in $\dot{\Sigma}$-Alg too, and Te' is also a coequaliser of Th and Tg. □

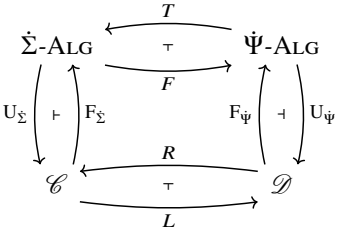
3.2∗15. The two examples of freeness conditions $\mathscr{A}$ in Example 3.1∗22 both satisfy the assumption in the theorem. In particular, the theorem applied to the translation Mon → Grp in Example 3.2∗4 constructs free groups over monoids. This theorem will later be used for constructing *free modular models*.

3.2∗16. The construction of relative free algebras is a special case of the *adjoint lifting problem* (Borceux, 1994, §4.5), which asks given functors $Q \circ T = R \circ G$ such that R has a left adjoint, whether T has a left adjoint as well?



The situation of Theorem 3.2∗14 is then the case where $L \dashv R$ is the identity adjunction $\mathrm{Id} \dashv \mathrm{Id} : \mathscr{C} \to \mathscr{C}$. An answer given by Borceux (1994, Theorem 4.5.6) is that if G and P are monadic and $\mathscr{B}$ has coequalisers, then T has a left adjoint. Therefore Theorem 3.2∗14 can be generalised as follows.

3.2∗17 Theorem. *Let $L \dashv R : \mathscr{D} \to \mathscr{C}$ be an adjunction, $\mathscr{A}$ be a freeness condition on $\mathscr{D}$, $\dot{\Sigma} \in \mathrm{Eqs}_f(\mathscr{C})$, and $\dot{\Psi} \in \mathrm{Eqs}_{\mathscr{A}}(\mathscr{D})$. A functor $T : \dot{\Psi}$-Alg $\to \dot{\Sigma}$-Alg such that $\mathrm{U}_{\dot{\Psi}} \circ T = R \circ \mathrm{U}_{\dot{\Psi}}$ has a left adjoint $F : \dot{\Sigma}$-Alg $\to \dot{\Psi}$-Alg if either* (1) *$\dot{\Psi}$-Alg has coequalisers or* (2) *$\mathscr{A}$ is closed under binary coproducts and constant functors.*



Proof sketch. Free-forgetful adjunctions of equational systems are always monadic (Fiore and Hur, 2009, Proposition 6.4), so if $\dot{\Psi}$ has coequalisers we have the left adjoint F by Borceux (1994, Theorem 4.5.6). Alternatively, if $\mathscr{A}$ satisfies the required property, we can construct the left adjoint in the same way as in the proof of Theorem 3.2∗14, inserting L and R in suitable places to go back and forth between $\mathscr{C}$ and $\mathscr{D}$. For example, the crucial diagram (3.6) in the proof of Theorem 3.2∗14 should be modified to

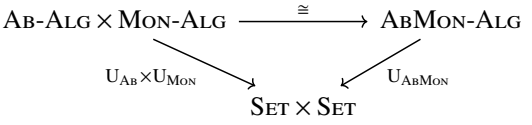
$$\begin{array}{ccc} \Sigma A & \xrightarrow{\Sigma i} & \Sigma RB \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle \tilde{T}\beta} \\ A & \xrightarrow{i} & RB \end{array}$$

where $\langle A \in \mathscr{C}, \alpha : \Sigma A \to A\rangle$ and $\langle B \in \mathscr{D}, \beta : \Psi B \to B\rangle$, and the functor $\tilde{T} : \Psi\text{-Alg} \to \Sigma\text{-Alg}$ from Lemma 3.2∗11 is modified to

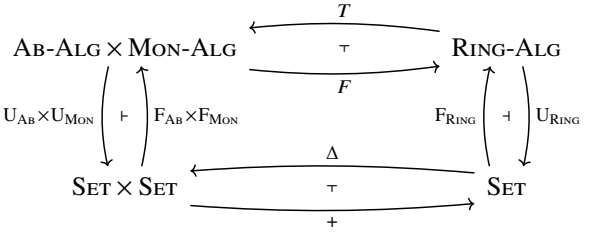
$$\tilde{T}\langle B, \beta\rangle = \langle RB,\ \Sigma RB \xrightarrow{\Sigma R\eta_B} \Sigma R(\mathrm{F}_{\dot\Psi} B) \xrightarrow{To} R\mathrm{F}_{\dot\Psi} B \xrightarrow{R\epsilon_{B,\beta}} RB\rangle.$$

where $o : \dot\Psi \mathrm{F}_{\dot\Psi} B \to \mathrm{F}_{\dot\Psi} B$ is the algebra structure on the free $\dot\Psi$-algebra. □

3.2∗18 Example. Instantiate the adjunction $L \dashv R$ in Theorem 3.2∗17 to be $+ \dashv \Delta : \textsc{Set} \to \textsc{Set} \times \textsc{Set}$. We have an equational system $\textsc{Ring} \in \textsc{Eqs}(\textsc{Set})$ of rings, and we can define an equational system $\textsc{AbMon} \in \textsc{Eqs}(\textsc{Set} \times \textsc{Set})$ such that an algebra of AbMon is a pair $\langle A, M\rangle$ of sets with an abelian group on A and a monoid on M, so we have the following commutative triangle:



Let $T : \textsc{Ring-Alg} \to \textsc{Ab-Alg} \times \textsc{Mon-Alg}$ be the functor that projecting out the additive group and multiplicative monoid structure of rings. It satisfies $\mathrm{U}_{\textsc{Ab}} \times \mathrm{U}_{\textsc{Mon}} \circ T = \Delta \circ \mathrm{U}_{\textsc{Ring}}$. By Theorem 3.2∗17, T has a left adjoint F:



The functor F generates a free ring *simultaneously* over an abelian group A and a monoid M. Generally, whenever we have a cospan of equational systems $\dot\Sigma \to \dot\Psi \leftarrow \dot\Phi$, we can construct an algebra of $\dot\Psi$ out of a $\dot\Sigma$-algebra and a $\dot\Phi$-algebra in a similar fashion. We will later use this technique to turn an ordinary model of an equational system to a modular model.

3.3 Colimits of Equational Systems

3.3∗1. Colimits in $\textsc{Eqs}(\mathscr{C})$ allow us to construct bigger equational systems by 'gluing' smaller ones. For example, the equational system for rings can be obtained by first taking the coproduct of Grp and Mon and then taking a suitable coequaliser $L \rightrightarrows \textsc{Grp} + \textsc{Mon}$ encoding the interaction of operations.

3.3∗2 Theorem. *The category* $\textsc{Eqs}_{\mathscr{A}}(\mathscr{C})$ *is small-cocomplete when the freeness condition* $\mathscr{A}$ *is closed under small-coproducts and if* $\langle \Psi, \Theta\rangle \in \mathscr{A}$ *then* $\langle 0, \Psi\rangle \in \mathscr{A}$.

Proof sketch. Arbitrary colimits can be constructed from coproducts and coequalisers, so it is sufficient to show that $\textsc{Eqs}_{\mathscr{A}}(\mathscr{C})$ has small-set-indexed coproducts and coequalisers. We sketch the constructions here without proof.

(i) The coproduct of a (small) set of equational systems is obtained by taking the coproduct of signatures and equations. Precisely, if $\langle \Sigma_i \triangleright \Gamma_i \vdash L_i = R_i\rangle_{i \in I}$ is a set of equational systems

with each of them in $\mathrm{Eqs}_{\mathscr{A}}(\mathscr{C})$, their coproduct is

$$\coprod_i \Sigma_i \triangleright \coprod_i \Gamma_i \vdash L = R$$

where $L, R : (\coprod_{i\in I} \Sigma_i)\text{-Alg} \to (\coprod_{i\in I} \Gamma_i)\text{-Alg}$ are

$$L\alpha = [L_i(\alpha \cdot \iota_i)]_{i\in I} \qquad\qquad R\alpha = [R_i(\alpha \cdot \iota_i)]_{i\in I}.$$

(ii) Let $T_1, T_2 : \dot{\Psi} \to \dot{\Sigma}$ be a pair of translations. Let $\dot{\Sigma}'$ be

$$\dot{\Sigma} \curlywedge (\Psi \vdash \tilde{T}_1 = \tilde{T}_2),$$

where $\tilde{T}_1, \tilde{T}_2 : \Sigma\text{-Alg} \to \Psi\text{-Alg}$ are obtained from T_1 and T_2 by Lemma 3.2∗11. By the assumption on $\mathscr{A}$, $\dot{\Sigma}'$ is still in $\mathrm{Eqs}_{\mathscr{A}}(\mathscr{C})$. The inclusion translation $\dot{\Sigma} \to \dot{\Sigma}'$ is the coequaliser of $T_1, T_2 : \dot{\Psi} \to \dot{\Sigma}$. □

3.3∗3 Corollary. For every lκp category $\mathscr{C}$, the freeness condition $\mathscr{A} = \mathrm{End}_{\kappa}(\mathscr{C}) \times \mathrm{End}_{\kappa}(\mathscr{C})$ satisfies the assumption of the theorem, so $\mathrm{Eqs}_{\mathscr{A}}(\mathscr{C})$ is cocomplete. For a small-complete small category $\mathscr{C}$, the freeness condition $\mathscr{A} = \mathrm{End}(\mathscr{C}) \times \mathrm{End}(\mathscr{C})$ is a freeness condition so $\mathrm{Eqs}(\mathscr{C})$ is cocomplete.

3.3∗4. Lastly, the following more description of a special case of pushouts of equational system will be convenient in the future.

3.3∗5 Lemma. Let $\dot{\Sigma} \in \mathrm{Eqs}(\mathscr{C})$ for a category $\mathscr{C}$ with finite coproducts, $\Theta_i : \mathscr{C} \to \mathscr{C}$ be an endofunctor, and $E_i = (\Theta_i \vdash L_i = R_i)$ an equation for $i \in \{1, 2\}$. Let T_1 and T_2 in the diagram below be the inclusion translations, then the following is a pushout diagram of T_1 and T_2:

$$\begin{array}{ccc} \dot{\Sigma} & \xrightarrow{\quad T_2 \quad} & \dot{\Sigma} \curlywedge \Phi_2 \curlywedge E_2 \\ {\scriptstyle T_1}\downarrow & & \downarrow \\ \dot{\Sigma} \curlywedge \Phi_1 \curlywedge E_1 & \longrightarrow & \dot{\Sigma} \curlywedge (\Phi_1 + \Phi_2) \curlywedge E \end{array}$$

where $E = (\Theta_1 + \Theta_2 \vdash [L_1 \circ \alpha_1, L_2 \circ \alpha_2] = [R_1 \circ \alpha_1, R_2 \circ \alpha_2])$ and

$$\alpha_i : (\Sigma + (\Phi_1 + \Phi_2))\text{-Alg} \to (\Sigma + \Phi_i)\text{-Alg}$$

are the evident forgetful functors.

Proof. First of all, there are evident inclusion translations P_i (which are the unlabelled arrows in the pushout diagram):

$$P_i : (\dot{\Sigma} \curlywedge (\Phi_1 + \Phi_2) \curlywedge E)\text{-Alg} \quad \to \quad (\dot{\Sigma} \curlywedge \Phi_i \curlywedge E_i)\text{-Alg}$$

such that $T_1 \circ P_1 = T_2 \circ P_2$. Now for every equational system $\dot{\Psi} \in \mathrm{Eqs}(\mathscr{C})$ with translations functors $Q_i : \dot{\Psi}\text{-Alg} \to (\dot{\Sigma} \curlywedge \Phi_i \curlywedge E_i)\text{-Alg}$ such that $T_1 \circ Q_1 = T_2 \circ Q_2$, we can define a translation functor

$$U : \dot{\Psi}\text{-Alg} \to (\dot{\Sigma} \curlywedge (\Phi_1 + \Phi_2) \curlywedge E')\text{-Alg}$$

by sending every $\dot{\Psi}$-algebra $\langle A, \alpha\rangle$ to the algebra on A with structure map:

$$[T_1(Q_1\dot{A}), \quad Q_1\alpha \cdot \iota_2, \quad Q_2\alpha \cdot \iota_2] : (\Sigma + \Phi_1 + \Phi_2)\, A \to A$$

It can be checked that such U is the unique one making $P_i \circ U = Q_i$. □

4 A Type Theory for Monoidal Categories

4∗1. When categorical constructions get complex, it is beneficial to use type theoretic *internal languages* to describe those constructions in a more intuitive manner (Crole, 1994; Jacobs, 1999; Lambek and Scott, 1986). This technique will be prevalent in this paper. In this section, we introduce *monoidal algebraic theories*, which allows us to present equational systems over monoidal categories and their algebras in a convenient syntactic manner.

4∗2. Monoidal algebraic theories are the *linear* counterpart of multi-sorted universal algebra, and the *syntactic* counterpart of *coloured PROs* (strict monoidal categories whose objects are products of a set of base objects) (MacLane, 1965). The presentation of monoidal algebraic theories below is based on the calculus of Jaskelioff and Moggi (2010), but the rules of judgemental equality is strengthened here to make terms form a monoidal category.

4.1 Monoidal Algebraic Theories

4.1∗1. A *monoidal algebraic theory* $\mathscr{L}$ is specified by three pieces of data $\langle \mathscr{B}, \mathscr{P}, \mathscr{A} \rangle$ as follows. Firstly, $\mathscr{B}$ is a set, and every element $\alpha \in \mathscr{B}$ is called a *base type*. The *types* of $\mathscr{L}$ are inductively generated by the rules

$$\frac{\alpha \in \mathscr{B}}{\vdash \alpha \text{ type}} \qquad \frac{}{\vdash I \text{ type}} \qquad \frac{\vdash A \text{ type} \qquad \vdash B \text{ type}}{\vdash A \square B \text{ type}}$$

Then $\mathscr{P}$ is a family of sets indexed by pairs of types A and B. Every element $f \in \mathscr{P}_{A,B}$ is called a *primitive operation*, and we will write $f : A \to B$ for $f \in \mathscr{P}_{A,B}$.

A *context* Γ is a finite list of variables and types $(x_1 : A_1, \ldots, x_n : A_n)$. The concatenation of two contexts is written as Γ_l, Γ_r.

The (well typed) *terms* under contexts Γ are generated by the following rules:

$$\frac{}{x : A \vdash x : A} \qquad \frac{f : A \to B \in \mathscr{P} \qquad \Gamma \vdash t : A}{\Gamma \vdash f\ t : B} \qquad \frac{}{\vdash * : I} \qquad \frac{\Gamma_1 \vdash t_1 : A \qquad \Gamma_2 \vdash t_2 : B}{\Gamma_1, \Gamma_2 \vdash (t_1, t_2) : A \square B}$$

$$\frac{\Gamma \vdash t_1 : I \qquad \Gamma_l, \Gamma_r \vdash t_2 : A}{\Gamma_l, \Gamma, \Gamma_r \vdash \mathit{let}\ * = t_1\ \mathit{in}\ t_2 : A} \qquad \frac{\Gamma \vdash t_1 : A_1 \square A_2 \qquad \Gamma_l, x_1 : A_1, x_2 : A_2, \Gamma_r \vdash t_2 : B}{\Gamma_l, \Gamma, \Gamma_r \vdash \mathit{let}\ (x_1, x_2) = t_1\ \mathit{in}\ t_2 : B}$$

Note that the type system is substructural, since the language is to be interpreted in monoidal categories rather than cartesian categories.

Lastly, $\mathscr{A}$ is a set of pairs $\langle \Gamma \vdash t_l : A,\ \Gamma \vdash t_r : A \rangle$ of terms of the same type and under the same context. Every element of $\mathscr{A}$ is called an *axiom* of $\mathscr{L}$. We will write $(\Gamma \vdash t_l = t_r : A) \in A$ when a pair $\langle \Gamma \vdash t_l : A, \Gamma \vdash t_r : A \rangle$ is in $\mathscr{A}$.

4.1∗2 Example. The monoidal algebraic theory of monoids has one base type M, two primitive operations $\mu : M \square M \to M$ and $\eta : I \to M$ and the following axioms, which correspond to the laws of monoids (2∗2):

$$\begin{aligned} x : M \vdash \mu(\eta(*), x) &= x : M & x : M \vdash \mu(x, \eta(*)) &= x : M \\ x : M, y : M, z : M \vdash \mu(\mu(x, y), z) &= \mu(x, \mu(y, z)) : M \end{aligned} \tag{4.1}$$

Although these axioms above look the same as the usual laws of monoids in Set, we will see below they can actually be interpreted in all monoidal categories, and a model of this theory will be exactly a monoid in a monoidal category.

4.1∗3 Lemma. The following *substitution rule* (or the 'cut rule') is admissible:

$$\frac{\Gamma_l, x : A, \Gamma_r \vdash t : B \qquad \Delta \vdash u : A}{\Gamma_l, \Delta, \Gamma_r \vdash t[u/x] : B}\ \textsc{Cut}$$

where $t[u/x]$ is substituting u for x in t.

Proof sketch. The typing rules of terms above all have substitution 'built-in' by having contexts Γ, Γ_l and Γ_r as general as possible. Thus the substitution rule can be shown by a straightforward induction on t. □

4.1∗4 Notation. In this paper, when we simultaneously substitute terms $u_1, \ldots, u_n$ for all the variables in the context of a term $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$, we usually just write $t[u_1, \ldots, u_n]$ instead of $t[u_1/x_1, \ldots, u_n/x_n]$.

4.1∗5. The *judgemental equality* $\Gamma \vdash t_1 \equiv t_2 : A$ of terms of a theory $\mathscr{L}$ is generated by the following rules plus the usual rules for reflexivity, symmetry, transitivity, and congruence under all term formers *and substitution*:

$$\frac{(\Gamma \vdash t_l = t_r : A) \in \mathscr{A}}{\Gamma \vdash t_l \equiv t_r : A}\ \textsc{Ax} \qquad \frac{\Gamma \vdash t : A}{\Gamma \vdash (\mathit{let} * = * \ \mathit{in}\ t : A) \equiv t : A}\ I\text{-}\beta$$

$$\frac{\Gamma \vdash t_1 : I \qquad \Gamma_l, x : I, \Gamma_r \vdash t_2 : A}{\Gamma_l, \Gamma, \Gamma_r \vdash (\mathit{let} * = t_1 \ \mathit{in}\ t_2[*/x]) \equiv t_2[t_1/x] : A}\ I\text{-}\eta$$

$$\frac{\Gamma_1 \vdash t_1 : A_1 \qquad \Gamma_2 \vdash t_2 : A_2 \qquad \Gamma_l, x_1 : A_1, x_2 : A_2, \Gamma_r \vdash t_3 : B}{\Gamma_l, \Gamma_1, \Gamma_2, \Gamma_r \vdash (\mathit{let}\ (x_1, x_2) = (t_1, t_2)\ \mathit{in}\ t_3) \equiv t_3[t_1/x_1, t_2/x_2] : B}\ \Box\text{-}\beta$$

$$\frac{\Gamma \vdash t_1 : A_1 \Box A_2 \qquad \Gamma_l, x : A_1 \Box A_2, \Gamma_r \vdash t_2 : B}{\Gamma_l, \Gamma, \Gamma_r \vdash (\mathit{let}\ (x_1, x_2) = t_1\ \mathit{in}\ t_2[(x_1, x_2)/x]) \equiv t_2[t_1/x] : B}\ \Box\text{-}\eta$$

The congruence rule under substitution is

$$\frac{\Gamma_l, x : A, \Gamma_r \vdash t_1 \equiv t_2 : B \qquad \Delta \vdash u : A}{\Gamma_l, \Delta, \Gamma_r \vdash t_1[u/x] \equiv t_2[u/x] : B}$$

4.1∗6. A *model* of a monoidal algebraic theory $\mathscr{L} = \langle \mathscr{B}, \mathscr{P}, \mathscr{A} \rangle$ in a monoidal category $\mathscr{E}$ consists of (1) an assignment of $\mathscr{E}$-objects $[\![\alpha]\!] \in \mathrm{Ob}(\mathscr{E})$ to each base type $\alpha \in \mathscr{B}$, which induces the interpretation of all types and contexts:

$$[\![I]\!] = I_{\mathscr{E}} \qquad [\![A \Box B]\!] = [\![A]\!] \Box_{\mathscr{E}} [\![B]\!]$$
$$[\![\cdot]\!] = I_{\mathscr{E}} \qquad [\![\Gamma, x : A]\!] = [\![\Gamma]\!] \Box_{\mathscr{E}} [\![A]\!]$$

and (2) an assignment of $\mathscr{E}$-morphisms $[\![f]\!] : [\![A]\!] \to [\![B]\!]$ to each primitive operation $f : A \to B \in \mathscr{P}$, which determines the interpretation of all terms:

$$[\![x]\!] = id \qquad [\![f\ t]\!] = [\![f]\!] \cdot [\![t]\!] \qquad [\![*]\!] = id \qquad [\![(t_1, t_2)]\!] = [\![t_1]\!] \Box_{\mathscr{E}} [\![t_2]\!]$$

$$[\![let * = t_1 \ in \ t_2]\!] = [\![t_2]\!] \cdot \_ \cdot ([\![\Gamma_1]\!] \square_{\mathscr{E}} [\![t_1]\!] \square_{\mathscr{E}} [\![\Gamma_2]\!])$$
$$[\![let \ (x_1, x_2) = t_1 \ in \ t_2]\!] = [\![t_2]\!] \cdot \_ \cdot ([\![\Gamma_l]\!] \square_{\mathscr{E}} [\![t_1]\!] \square_{\mathscr{E}} [\![\Gamma_r]\!])$$

where the underscores stand for the unique isomorphisms built from associators α and unitors λ, ρ to make the domain and codomain match. Moreover, the assignments $[\![-]\!]$ of a model must make $[\![t_l]\!] = [\![t_r]\!]$ for all axioms $\langle t_l, t_r \rangle \in \mathscr{A}$.

4.1∗7. Let M and N be two models of a theory $\mathscr{L} = \langle \mathscr{B}, \mathscr{P}, \mathscr{A} \rangle$ in a monoidal category $\mathscr{E}$. A family of $\mathscr{E}$-morphisms $h_\alpha : [\![\alpha]\!]_M \to [\![\alpha]\!]_N$ for all base types $\alpha \in \mathscr{B}$ extends to a family of morphisms $\tilde{h}_A : [\![A]\!]_M \to [\![A]\!]_N$ for all $\mathscr{L}$-types A:

$$\tilde{h}_I = id_I \qquad \tilde{h}_{A \square B} = \tilde{h}_A \square_{\mathscr{E}} \tilde{h}_B \qquad \tilde{h}_\alpha = h_\alpha.$$

Such a family of morphisms h is called a *homomorphism* from M to N if for every primitive operation $f : A \to B \in \mathscr{P}$, the following commutes:

$$\begin{array}{ccc} [\![A]\!]_M & \xrightarrow{[\![f]\!]_M} & [\![B]\!]_M \\ {\scriptstyle \tilde{h}_A}\downarrow & & \downarrow{\scriptstyle \tilde{h}_B} \\ [\![A]\!]_N & \xrightarrow[[\![f]\!]_N]{} & [\![B]\!]_N \end{array}$$

Models of $\mathscr{L}$ in $\mathscr{E}$ and their homomorphisms assemble to a category $\mathscr{L}$-Mod$(\mathscr{E})$.

4.1∗8 Theorem (Soundness). *Let $[\![-]\!]$ be a model of a monoidal algebraic theory $\mathscr{L}$. If two $\mathscr{L}$-terms are judgementally equal, $\Gamma \vdash t_1 \equiv t_2 : A$, then $[\![t_1]\!] = [\![t_2]\!]$.*

Proof sketch. It is straightforward verification that the rules of judgemental equalities in 4.1∗5 is validated by the axioms of a monoidal category after we show the *substitution lemma*: for all $\Gamma_l, x : A, \Gamma_r \vdash t : B$ and $\Delta \vdash u : A$, we have

$$[\![t[u/x]]\!] \ = \ [\![t]\!] \cdot (\Gamma_l \square [\![u]\!] \square \Gamma_r) : [\![\Gamma_l, \Delta, \Gamma_r]\!] \to [\![B]\!]$$

which itself can be proven by induction on t. □

4.1∗9. Given a monoidal category $\mathscr{E}$, its *internal language* $\mathscr{L}(\mathscr{E})$ is a monoidal algebraic theory defined as follows:

- The set of base types of $\mathscr{L}(\mathscr{E})$ is exactly Obj $\mathscr{E}$, so we have an interpretation of all types as objects in $\mathscr{E}$ by interpreting every base type as itself.
- The set of primitive operations between two types A and B is $\mathscr{E}([\![A]\!], [\![B]\!])$. Again, by interpreting every primitive operation as itself in the category $\mathscr{E}$, we have an interpretation of all terms in $\mathscr{E}$ as in 4.1∗6.
- The set of axioms of $\mathscr{L}(\mathscr{E})$ is the maximal one containing all pairs of terms $\langle t_1, t_2 \rangle$ such that $[\![t_1]\!] = [\![t_2]\!]$ in $\mathscr{E}$.

The canonical interpretation is a model of $\mathscr{L}(\mathscr{E})$ by construction. The internal language $\mathscr{L}(\mathscr{E})$ of a monoidal category $\mathscr{E}$ is *sound and complete* for reasoning about $\mathscr{E}$: two terms in $\mathscr{L}(\mathscr{E})$ satisfy $t_1 \equiv t_2$ if and only if they are equal under the canonical interpretation in $\mathscr{E}$. Soundness follows from Theorem 4.1∗8 and completeness is by the construction of $\mathscr{L}(\mathscr{E})$.

4.1∗10. What we have above is enough for our purpose of using a convenient syntax to describe and reason about constructions in monoidal categories, but there are certainly more

things can be said about monoidal algebraic theories. Following the standard agenda of categorical logic, we expect the following to be true but leave them as future work:

- Types and terms of $\mathscr{L}$ underlie a *classifying monoidal category* $\mathscr{M}(\mathscr{L})$ there is a model of $\mathscr{L}$ in $\mathscr{M}(\mathscr{L})$, and there is an equivalence of categories
$$\mathscr{L}\text{-}\textsc{Mod}(\mathscr{E}) \cong \textsc{MonCat}_s(\mathscr{M}(\mathscr{L}), \mathscr{E}),$$
where the right-hand side is the category of strict monoidal functors $\mathscr{M}(\mathscr{L}) \to \mathscr{E}$ and monoidal natural transformations.
- There is a 2-category $\textsc{Mat}$ of monoidal algebraic theories and a 2-equivalence
$$\textsc{Mat} \underset{\mathscr{M}}{\overset{\mathscr{L}}{\underset{\longrightarrow}{\overset{\longleftarrow}{\cong}}}} \textsc{MonCat}_s$$
between $\textsc{Mat}$ and the 2-category of monoidal categories, strict monoidal functor and monoidal natural transformations.

4.1∗11 Remark. We have defined the syntax and semantics of monoidal algebraic theories manually. A more modern approach would be directly define monoidal algebraic theories using the framework of *generalised algebraic theories* (Cartmell, 1986, 1978) (or the closely related framework of *quotient inductive-inductive types* (Altenkirch and Kaposi, 2016; Altenkirch et al., 2018; Kovács, 2023)). Using these frameworks would then directly give us a notion of models of monoidal algebraic theories and syntactic models.

4.2 More Type Formers for Monoidal Algebraic Theories

4.2∗1. Sometimes we work in monoidal categories with additional structure, and in this case we extend monoidal algebraic theories with new syntax for the additional structure. For example, when we work in left closed monoidal categories, we extend the calculus with a new type former B/A and term formers:

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A.\, t : B/A} \qquad \frac{\Gamma_1 \vdash t_1 : B/A \qquad \Gamma_2 \vdash t_2 : A}{\Gamma_1, \Gamma_2 \vdash t_1\ t_2 : B}$$

whose interpretation in a model is given by the corresponding structure of the closed monoidal category:

$$[\![\lambda x : A.\, t : B/A]\!] = abst([\![t]\!]) \qquad [\![t_1\ t_2]\!] = ev \cdot ([\![t_1]\!] \square_{\mathscr{E}} [\![t_2]\!])$$

where $abst : \mathscr{E}(C \square_{\mathscr{E}} A, B) \to \mathscr{E}(C, B/_{\mathscr{E}} A)$ is the natural isomorphism associated to the adjunction $(- \square_{\mathscr{E}} A) \dashv (-/_{\mathscr{E}} A)$ and $ev : (B/_{\mathscr{E}} A) \square_{\mathscr{E}} A \to B$ is its counit. The β and η rules characterising the universal property of B/A are added to the equational theory routinely.

4.2∗2. Note that the type former B/A is *contravariant* in the position of A, so in the presence of /-types, the way in 4.1∗7 of extending a family of morphisms $h_\alpha : [\![\alpha]\!]_M \to [\![\alpha]\!]_N$ between interpretations of base types to a family of morphisms $\tilde{h}_A$ between interpretations of all types will not work, and we will not have a category of models and *model homomorphisms* for a theory $\mathscr{L}$ with /-types. However, we can still have a category $\mathscr{L}\text{-}\textsc{Mod}_{\cong}(\mathscr{E})$ of models and *model isomorphisms* by demanding that every h_α must be an $\mathscr{E}$-isomorphism.

4.2∗3. As a comment on the notation, Jaskelioff and Moggi (2010) used B^A for the closed structure, while we use Lambek's [1958] notation B/A to avoid the confusion with exponentials (which are right adjoints to cartesian products).

4.2∗4 Example. Consider the theory of monoids from Example 4.1∗2 and the extension of the closed structure B/A as in 4.2∗1. *Cayley's theorem* from elementary algebra adapted to this setting says that the monoid $\langle M, \mu, \eta\rangle$ embeds into the monoid M/M with unit $\vdash \lambda x.\, x : M/M$ and multiplication

$$f : M/M,\ g : M/M \vdash \lambda x.\, f\ (g\ x) : M/M. \tag{4.2}$$

The embedding is given by $e = (x : M \vdash \lambda y.\, \mu(x, y) : M/M)$, which is a monoid homomorphism in the following sense:

$$e[\eta\, *] \equiv \lambda y.\, \mu(\eta\, *, y) \equiv \lambda y.\, y$$
$$e[\mu(x, y)] \equiv \lambda z.\, \mu(\mu(x, y), z) \equiv \lambda z.\, \mu(x, \mu(y, z)) \equiv \lambda z.\, e[x]\ (e[y]\ z)$$

where we write $t[-]$ for substitution when there is a unique variable in the context of t. The embedding e has a left inverse $r = (f : M/M \vdash f\ (\eta\, *) : M)$ such that

$$x : M \vdash r[e] \equiv (\lambda y.\, \mu(x, y))\ (\eta\, *) \equiv x : M$$

However, the inverse r is in general *not* a monoid homomorphism: in the context $(f : M/M, g : M/M)$, we have

$$r[\lambda x.\, f\ (g\ x)] \equiv f\ (g\ (\eta\, *)) \not\equiv \mu(f\ (\eta\, *),\ g\ (\eta\, *)) \equiv \mu(r[f], r[g])$$

This elementary result has many applications in functional programming because the multiplication (4.2) is usually a kind of function composition with $O(1)$ time complexity, regardless of the possibly expensive multiplication μ. When M is a free monoid in $\langle \textsc{Set}, \times, 1\rangle$, i.e. a list, this optimisation is known as *difference lists* (Hughes, 1986). In $\langle \textsc{Endo}_\kappa(\mathscr{C}), \circ, \mathrm{Id}\rangle$, this optimisation is known as *codensity transformation* (Hinze, 2012).

4.2∗5. Cartesian products and coproducts in monoidal categories can be internalised similarly: we add type formers $A \times B$ and $A + B$ with term formers:

$$\frac{\Gamma \vdash t_1 : A_1 \qquad \Gamma \vdash t_2 : A_2}{\Gamma \vdash \langle t_1, t_2\rangle : A_1 \times A_2}$$

$$\frac{\Gamma \vdash t_1 : A_1 \times A_2 \qquad \Gamma_l, x : A_i, \Gamma_r \vdash t_2 : B}{\Gamma_l, \Gamma, \Gamma_r \vdash t_2[\pi_i\, t_1/x] : B}\ i \in \{1, 2\}$$

$$\frac{\Gamma \vdash t : A_i}{\Gamma \vdash \iota_i\, t : A_1 + A_2}\ i \in \{1, 2\}$$

$$\frac{\Gamma \vdash t : A_1 + A_2 \qquad x_i : A_i \vdash t_i : C, i \in \{1, 2\}}{\Gamma \vdash \mathit{case}\ t\ \mathit{of}\ \{\iota_1\, x_i \mapsto t_1;\ \iota_2\, x_2 \mapsto t_2\} : C}$$

as well as their β and η rules. Sometimes we also write $[t_1, t_2]\ t$ instead of $\mathit{case}\ t\ \mathit{of}\ \{\iota_1\, x_i \mapsto t_1;\ \iota_2\, x_2 \mapsto t_2\}$ for brevity.

These rules straightforwardly generalise to the non-binary cases. Note that the first rule introduces non-linearity to the syntax as the variables in Γ may appear in both t_1 and t_2.

4.2∗6 Remark. In the presence of cartesian products, we can borrow the idea of *bunched logic* (O'Hearn and Pym, 1999) to introduce a new context former $\Gamma;\Gamma'$ with semantics $[\![\Gamma;\Gamma']\!] = [\![\Gamma]\!] \times [\![\Gamma']\!]$. This would simplify talking about linear functions $-/A$ (right adjoint to $-\,\square\,A$) and non-linear functions $-^A$ (right adjoint to $-\times A$) at the same time, but we will not need this extension in this paper.

4.3 Syntactic Presentations of Equational Systems

4.3∗1. In this subsection, we carve out a class of monoidal algebraic theories that can be defined as equational systems, therefore allowing us to apply the theorems of equational systems to those monoidal algebraic theories, or from the opposite perspective, allowing us to define equational systems syntactically use those monoidal algebraic theories.

4.3∗2 Definition. Let $\mathscr{L} = \langle \mathscr{B}, \mathscr{P}, \mathscr{A} \rangle$ be a monoidal algebraic theory. When all primitive operations $f : A \to B$ and equations $\Gamma \vdash t_l = t_r : B$ of a monoidal algebraic theory $\mathscr{L}$ satisfy that $B \in \mathscr{B}$, we say that $\mathscr{L}$ has *basic outputs*.

4.3∗3. For example, the theory of monoids in Example 4.1∗2 has basic outputs, but the internal language $\mathscr{L}(\mathscr{E})$ of a monoidal category $\mathscr{E}$ does not have basic outputs because not all axioms of $\mathscr{L}(\mathscr{E})$ have a base type B.

4.3∗4 Theorem. *For every monoidal algebraic theory $\mathscr{L} = \langle \mathscr{B}, \mathscr{P}, \mathscr{A} \rangle$ with basic outputs and every monoidal category $\mathscr{E}$ with small-coproducts, there is an equational system $\Sigma_{\mathscr{L}}$ over the product category $\mathscr{E}^{\mathscr{B}}$ such that $\mathscr{L}$-Mod$(\mathscr{E}) \cong \Sigma_{\mathscr{L}}$-Alg.*

Proof. Recall that types A of $\mathscr{L}$ are generated by I, $\square$, and $\alpha \in \mathscr{B}$. Therefore the interpretation of every type A in $\mathscr{E}$ determines a functor $[\![A]\!] : \mathscr{E}^{\mathscr{B}} \to \mathscr{E}$:

$$[\![I]\!]X = I_{\mathscr{E}} \qquad [\![A_1 \,\square\, A_2]\!]X = [\![A_1]\!]X \,\square\, [\![A_2]\!]X \qquad [\![\alpha]\!]X = X_\alpha$$

Similarly, every context Γ of $\mathscr{L}$ determines a functor $[\![\Gamma]\!] : \mathscr{E}^{\mathscr{B}} \to \mathscr{E}$.

We define an endofunctor $[\![\mathscr{P}]\!] : \mathscr{E}^{\mathscr{B}} \to \mathscr{E}^{\mathscr{B}}$ by

$$[\![\mathscr{P}]\!]X = \langle \coprod_A \coprod_{f:A\to\alpha\in\mathscr{P}} [\![A]\!]X \rangle_{\alpha\in\mathscr{B}}.$$

It can be seen that an algebra of the endofunctor $[\![\mathscr{P}]\!]$ is precisely an interpretation of base types $\mathscr{B}$ and primitive operations $\mathscr{P}$ in $\mathscr{E}$.

Let $\uparrow_\alpha : \mathscr{E} \to \mathscr{E}^{\mathscr{B}}$ be the functor mapping every $X \in \mathscr{E}$ and $\beta \in \mathscr{B}$ to X if $\alpha = \beta$, or to $0_{\mathscr{E}}$ if $\alpha \neq \beta$. We write $[\![\Gamma \vdash \alpha]\!] : \mathscr{E}^{\mathscr{B}} \to \mathscr{E}^{\mathscr{B}}$ for $\uparrow_\alpha \circ [\![\Gamma]\!]$. An algebra of the endofunctor $[\![\Gamma \vdash \alpha]\!]$ is then an object $X \in \mathscr{E}^{\mathscr{B}}$ with a morphism in $\mathscr{E}^{\mathscr{B}}$

$$\langle \uparrow_\alpha ([\![\Gamma]\!]X)\,\beta \rangle_{\beta\in\mathscr{B}} \longrightarrow \langle X_\beta \rangle_{\beta\in\mathscr{B}},$$

which amounts to just an $\mathscr{E}$-morphism $[\![\Gamma]\!]X \to X_\alpha$, i.e. $[\![\Gamma]\!]X \to [\![\alpha]\!]X$, since all other components are the unique $0_{\mathscr{E}} \to X_\beta$. Therefore the interpretation of every term $\Gamma \vdash t : \alpha$ of $\mathscr{L}$ with $\alpha \in \mathscr{B}$ then determines a mapping from a $[\![\mathscr{P}]\!]$-algebra to a $[\![\Gamma \vdash \alpha]\!]$-algebra. By induction on the term t in the style of Reynolds's [1983] *abstraction theorem*, it can be shown that this mapping extends to a functor $[\![t]\!] : [\![\mathscr{P}]\!]$-Alg $\to [\![\Gamma \vdash \alpha]\!]$-Alg satisfying $U_{[\![\Gamma\vdash\alpha]\!]} \circ [\![t]\!] = U_{[\![\mathscr{P}]\!]}$.

Let $\Sigma_{\mathscr{L}}$ be the equational system over $\mathscr{E}^{\mathscr{B}}$ with signature $[\![\mathscr{P}]\!]$ and a set of functorial equations $[\![\Gamma]\!] \vdash [\![t_l]\!] = [\![t_r]\!]$ for every axiom $(\Gamma \vdash t_l = t_r : \alpha) \in \mathscr{A}$ (c.f. Notation 3.1∗7). By construction, the category $\mathscr{L}$-Mod$(\mathscr{E})$ of $\mathscr{L}$-models in $\mathscr{E}$ and the category $\Sigma_{\mathscr{L}}$-Alg of $\Sigma_{\mathscr{L}}$-algebras (in $\mathscr{E}^{\mathscr{B}}$) are isomorphic. □

4.3∗5 Corollary. Let $\mathscr{E}$ be a cocomplete monoidal category with $\Box$ preserving colimits of α-chains for a limit ordinal α. By the theorem above and Theorem 3.1∗12, the category of models $\mathscr{L}$-Mod$(\mathscr{E})$ of a monoidal algebraic theory $\mathscr{L}$ with basic outputs is cocomplete and monadic over the category $\mathscr{E}^{\mathscr{B}}$.

4.3∗6 Example. The monoidal algebraic theory of monoids in Example 4.1∗2 has basic outputs, and the corresponding equational system obtained using the theorem above is precisely the equational system in Example 3.1∗6.

4.3∗7 Remark. The restriction of basic outputs is reminiscent of the relationship between *operads* and *PROPs* (Markl, 2006), which are two frameworks for doing algebraic theories in symmetric monoidal categories. It is also the case that the former allows multiple inputs but one output, whereas the latter allows multiple inputs and multiple outputs. However, an advantage of the restricted frameworks is that the category of algebras of operads/monoidal algebraic theories with basic outputs is monadic (under the conditions of Corollary 4.3∗5).

4.3∗8. When considering monoidal algebraic theories with extra type formers that are not covariant, such as the linear function type B/A in 4.2∗1, Theorem 4.3∗4 must additionally require a theory $\mathscr{L} = \langle \mathscr{B}, \mathscr{P}, \mathscr{A} \rangle$ with basic outputs to have *positive inputs*: every primitive operation $f : A \to \alpha$ and equation $\Gamma \vdash t_l = t_r : \alpha$ of $\mathscr{L}$ must satisfy that every base type $\beta \in \mathscr{B}$ occurs *positively* in A and Γ, so that A and Γ can be interpreted as covariant functors $\mathscr{E}^{\mathscr{B}} \to \mathscr{E}$. The rule for types in which $\beta \in \mathscr{B}$ occurs positively P and negatively N is defined inductively by the following grammar as usual:

$$P := \beta \mid \alpha \mid P \,\Box\, P \mid P/N \qquad\qquad N := \alpha \mid N \,\Box\, N \mid N/P$$

where α ranges over $\mathscr{B} \setminus \{\beta\}$. A base type β occurs positively in a context Γ if it occurs positively in every item of the context Γ. Other covariant type formers such as $\times$ and $+$ in 4.2∗5 should be treated similarly to $\Box$.

4.3∗9. As explained above, type expressions P in which base types occur positively denote functors $[\![P]\!]$. It will be convenient if we also have access in the term language to the corresponding action of $[\![P]\!]$ on morphisms. Let $\tau \in \mathscr{B}$ and $x : A \vdash f : B$. For all types P and N in which $\tau \in \mathscr{B}$ occurs positively and negatively respectively, we define terms

$$x : P[A/\tau] \vdash P_\tau f : P[B/\tau] \qquad\qquad x : N[B/\tau] \vdash N_\tau f : N[A/\tau]$$

by induction on the structure of P and N:

$$\tau_\tau f := f \qquad \alpha_\tau f := x \qquad (P/N)_\tau f := \lambda y.\, P_\tau f\ [x\ (N_\tau f\ [y])]$$
$$(P \,\Box\, P')_\tau f := \mathit{let}\ (x_l, x_r) = x\ \mathit{in}\ (P_\tau f\ [x_l], P'_\tau f\ [x_r])$$

and symmetrically for N. When the type expression P contains (at most) one base type τ, we will just write Pf for $P_\tau f$.

As a convention, we will write $P_\tau f\ t$ for $P_\tau f[t/x]$, so we have an admissible rule:

$$\frac{x : A \vdash f : B \qquad \Gamma \vdash t : P[A/\tau]}{\Gamma \vdash P_\tau f\ t : P[B/\tau]}$$

As an example, if $\mathscr{B} = \{\tau\}$, then τ occurs positively in the type expression $T = \tau \,\square\, \tau$, which denotes the functor $\square : \mathscr{E} \times \mathscr{E} \to \mathscr{E}$. Moreover we have $\Gamma \vdash T f\ t : B \,\square\, B$ for all terms $A \vdash f : B$ and $\Gamma \vdash t : A \,\square\, A$.

4.3∗10. A notable difference between a monoidal algebraic theory $\mathscr{L}$ and an equational system $\dot{\Sigma}$ over a monoidal category $\mathscr{E}$ is that the definition of the former (4.1∗1) makes no reference to any specific monoidal category, while the definition of the latter (3.1∗4) is parameterised by the underlying category $\mathscr{E}$. Therefore the operations of $\mathscr{L}$ must make sense for all monoidal categories. Such a generality can also be a limitation: sometimes we may be interested in operations that only make sense for a specific monoidal category $\mathscr{E}$. In this case, we need a slight generalisation of Theorem 4.3∗4 to use monoidal algebraic theories to denote equational systems over $\mathscr{E}$ syntactically.

4.3∗11. Let $\mathscr{L} = \langle \mathscr{B}, \mathscr{P}, \mathscr{A} \rangle$ be a monoidal algebraic theory and M be a model of it in some monoidal category $\mathscr{E}$. We say that another theory $\mathscr{L}' = \langle \mathscr{B}', \mathscr{P}', \mathscr{A}' \rangle$ is an *extension* of $\mathscr{L}$ if $\mathscr{B} \subseteq \mathscr{B}'$, $\mathscr{P} \subseteq \mathscr{P}'$, and $\mathscr{A} \subseteq \mathscr{A}'$, where we implicitly treat types/terms generated by $\mathscr{B}$ and $\mathscr{P}$ as types/terms generated generated by the superset $\mathscr{B}'$ and $\mathscr{P}'$. Moreover, a model M' of $\mathscr{L}'$ in $\mathscr{E}$ is said to be *over* the model M if $[\![A]\!]_{M'} = [\![A]\!]_M$ and $[\![t]\!]_{M'} = [\![t]\!]_M$ for all types A and terms t of $\mathscr{L}$.

For example, let $\mathscr{E}$ be a monoidal category and $A \in \mathscr{E}$ be an object of $\mathscr{E}$. Let $\mathscr{L}'$ be the extension of the internal language $\mathscr{L}(\mathscr{E})$ with a new base type τ and a new operation $f : A \to \tau$. A model of $\mathscr{L}'$ over the canonical model of $\mathscr{L}(\mathscr{E})$ in $\mathscr{E}$ is precisely an object $X \in \mathscr{E}$ with a morphism $f : A \to X$.

4.3∗12. We say that an extension $\mathscr{L}'$ of $\mathscr{L}$ has basic outputs if every new operation $(f : A \to B) \in (\mathscr{P}' \setminus \mathscr{P})$ and every equation $(\Gamma \vdash t_l = t_r : B) \in (\mathscr{A}' \setminus \mathscr{A})$ satisfies that $B \in \mathscr{B}' \setminus \mathscr{B}$. Similarly, we say that the extension $\mathscr{L}'$ has positive inputs if every $\alpha \in \mathscr{B}' \setminus \mathscr{B}$ occurs positively in A and Γ.

4.3∗13 Theorem. *Let $\mathscr{L}$ be a monoidal algebraic theory (with possibly extra type formers such as $/$, $+$, $\times$-types in Section 4.2), and let M be a model of $\mathscr{L}$ in a monoidal category $\mathscr{E}$ with small-coproducts. Every extension $\mathscr{L}'$ of $\mathscr{L}$ that has basic outputs (and positive inputs) determines an equational system $\Sigma_{\mathscr{L}',M}$ over $\mathscr{E}^{\mathscr{B}\setminus\mathscr{B}'}$ such that algebras of $\Sigma_{\mathscr{L}',M}$ are in bijection with models of $\mathscr{L}'$ in $\mathscr{E}$ over M.*

Proof. The proof goes almost the same as Theorem 4.3∗4, except that the base types and primitive operations of $\mathscr{L}$ are fixed by the model M. □

4.3∗14. Our main use case of Theorem 4.3∗13 is when $\mathscr{L}$ is the internal language $\mathscr{L}(\mathscr{E})$ of a monoidal category with coproducts $\mathscr{E}$ (4.1∗9), and M is the canonical model, and there is exactly one new base type: $\mathscr{B}' \setminus \mathscr{B} = \{\tau\}$ of $\mathscr{L}(\mathscr{E})$ in $\mathscr{E}$. This allows us to present an equational system over $\mathscr{E}$ syntactically by a set of operations $f : A \to \tau$ and axioms $\Gamma \vdash t_l = t_r : \tau$ where A, Γ, t_l and t_r can refer to the existing objects and morphisms in $\mathscr{E}$.

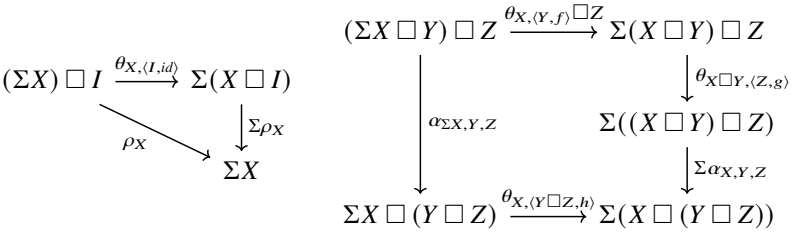
5 Equational Systems for Monoids with Operations

5∗1. After the detour to monoidal algebraic theories, now we come back to monoids with operations, by which we mean the equational system Mon extended with a new operation $op : \Sigma M \to M$ for some endofunctor $\Sigma : \mathscr{E} \to \mathscr{E}$ and possibly some equations. In this section, we will discuss a general notion known as Σ-*monoids* in the literature (Fiore et al., 1999), which demands that the operation op is compatible with the monoid multiplication in a sense. After this, we will see some concrete examples.

5∗2. Let $\mathscr{E}$ be a monoidal category and $\Sigma : \mathscr{E} \to \mathscr{E}$ be an endofunctor. A *pointed strength* θ for Σ is a natural transformation

$$\theta_{X,\langle Y,f\rangle} : (\Sigma X) \square Y \to \Sigma(X \square Y)$$

for all X in $\mathscr{E}$ and $\langle Y \in \mathscr{E}, f : I \to Y\rangle$ in the coslice category $I/\mathscr{E}$, satisfying coherence conditions analogous to those of strengths (2.4, 2.5):



for all $X, Y, Z \in \mathscr{E}$, $f : I \to Y$, $g : I \to Z$, and $h := (f \square g) \cdot \rho_I^{-1} : I \to Y \square Z$.

5∗3. To denote Σ and θ syntactically, we extend the internal language $\mathscr{L}(\mathscr{E})$ (4.1∗9) with a new type constructor Σ and the following typing rules

$$\frac{x : A \vdash f : B \qquad \Gamma \vdash t : \Sigma A}{\Gamma \vdash \Sigma f\ t : \Sigma B} \qquad\qquad \frac{\cdot \vdash f : Y \qquad \Gamma \vdash t : (\Sigma X) \square Y}{\Gamma \vdash \theta_{X,\langle Y,f\rangle}\ t : \Sigma(X \square Y)}$$

which of course are interpreted in $\mathscr{E}$ by the functor Σ and the strength θ.

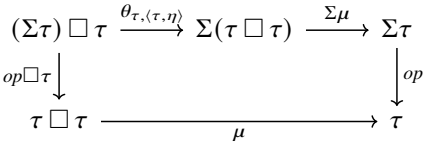
5∗4. The equational system Σ-Mon of Σ-*monoids* (Fiore et al., 1999; Fiore and Hur, 2009) extends the theory Mon of monoids (3.1∗6 and 4.1∗2) with a new operation $op : \Sigma\tau \to \tau$ and a new equation $L_{\Sigma\text{-Mon}} = R_{\Sigma\text{-Mon}}$:

$$\Sigma\text{-Mon} = \text{Mon} \curvearrowleft \Sigma \curvearrowleft \left(\Sigma - \square - \vdash L_{\Sigma\text{-Mon}} = R_{\Sigma\text{-Mon}}\right) \tag{5.1}$$

where the new equation $L_{\Sigma\text{-Mon}} = R_{\Sigma\text{-Mon}}$ is given in terms of an extension of the internal language of $\mathscr{E}$ by the following pair of terms:

$$x : \Sigma\tau, y : \tau \vdash \mu\ (op\ x, y) = op\ (\Sigma\mu\ (\theta_{\tau,\langle\tau,\eta\rangle}\ (x, y))) : \tau \tag{5.2}$$

which encodes the following commutative diagram:



Note that (5.2) refers to θ, so technically the equational system should be denoted by $\langle \Sigma, \theta \rangle$-Mon, but writing Σ-Mon is unlikely to cause confusion.

5∗5. Equation (5.2) expresses that the operation *op* commutes with monoid multiplication. When using $\langle \textsc{Endo}_f(\textsc{Set}), \bullet, V \rangle$ for modelling higher-order abstract syntax, which was the original context where Fiore et al. (1999) introduced Σ-monoids, this equation expresses the sensible condition that syntactic operations must commute with substitution.

However, this equation might not be desirable in other contexts. For example, in the study of *modal type theories*, there are plenty of operations that do not commute with substitution (Gratzer, 2023). Also, when using $\langle \textsc{Endo}_\kappa(\mathscr{C}), \circ, \mathrm{Id} \rangle$ to model computational effects, this equation expresses that effectful operations must commute with sequential composition, which is not true in general. Those effectful operations that do satisfy this condition and have signature $\Sigma = A \square -$ for some $A \in \mathscr{E}$ are called *algebraic operations* (Plotkin and Power, 2001; Jaskelioff and Moggi, 2010). We will say more about equation (5.2) shortly in Lemma 5∗13 and see that imposing it on Σ-monoids actually does not lose generality.

5∗6. When the monoidal category $\mathscr{E}$ is cocomplete and functors Σ, $\square$ both preserve colimits of α-chains for some limit ordinal α, Theorem 3.1∗12 ensures the existence of free Σ-monoids. When $\mathscr{E}$ is additionally closed, such as $\langle \textsc{Endo}_\kappa(\mathscr{C}), \circ, \mathrm{Id} \rangle$ in Section 2, there is a simple description of the free Σ-monoid (Fiore and Hur, 2007): it is carried by the initial algebra $\mu X.\, I + A \square X + \Sigma X$. This formula has many applications in modelling abstract syntax: variable binding (Fiore and Szamozvancev, 2022), explicit substitution (Ghani et al., 2006), and scoped operations (Piróg et al., 2018).

5∗7 Remark. Given a strong monad T over $\mathscr{E}$, Piróg (2016), Fiore and Saville (2017) studied a notion akin to Σ-monoids in 5∗4 (with $\Sigma = T$) but additionally asking the operation $op : T\tau \to \tau$ to be an Eilenberg-Moore algebra of T. This concept is called *Eilenberg-Moore monoids* by Piróg (2016) and *T-monoids* by Fiore and Saville (2017). When $\mathscr{E}$ is closed, there is again a simple formula $\mu X.\, T(I + A \square X)$ for the free T-monoid/Eilenberg-Moore monoid over $A \in \mathscr{E}$. Thinking of a monad T as an algebraic theory, using T-monoids instead of Σ-monoids allows us to impose equational laws on the operation *op*. In this paper we have opted in to use equational systems to present equations, so in the following we will continue with using Σ-monoids (with possibly additional equations). This has an additional advantage that we can have equations that involve both the monoid operations η, μ and the operation *op* at the same time, rather than just equations involving *op*.

5∗8. Now let us look at some concrete examples. In all the following examples, the monoidal category $\langle \mathscr{E}, \square, I \rangle$ is assumed to have small-coproducts $\coprod_{i \in S} A_i$ and finite products $\prod_{i \in F} A_i$. Additionally, we assume the monoidal product distributes over coproducts from the right:

$$\left(\coprod_{i \in S} A_i\right) \square B \;\cong\; \coprod_{i \in S} (A_i \square B).$$

5∗9 Example. Let S be a set. The theory St_S of *monads with global S-state* (Plotkin and Power, 2002) can be generally defined for monoids as follows. The theory St_S is Σ_{St_S}-Mon with signature Σ_{St_S} denoted by the type expression:

$$\left(\left(\prod_S I\right) \square \tau\right) + \left(\left(\coprod_S I\right) \square \tau\right),$$

whose first component represents an operation $g : (\prod_S I) \square \tau \to \tau$ reading the state, and the second component is an operation $p : (\coprod_S I) \square \tau \to \tau$ writing an S-value into the state.

Plotkin and Power's [2002] equations of these two operations can also be specified at this level of generality. For example, the law saying that writing $s \in S$ to the state and reading it immediately gives back s is

$$k : \prod_S I \vdash p_s(g(k, \eta^\tau)) = p_s(let * = \pi_s k \ in \ \eta^\tau) : \tau$$

where $p_s(x)$ abbreviates $p(\iota_s *, x)$.

5∗10 Example (Exception throwing). Letting E be a set, the theory ET_E of *exception throwing* is the theory Σ_{ET_E}-MON where $\Sigma_{\mathrm{ET}_E} = (\coprod_E 1) \square - : \mathscr{E} \to \mathscr{E}$ equipped the associator $\alpha : ((\coprod_E 1) \square X) \square Y \to \coprod_E 1 \square (X \square Y)$ as the strength, where $\coprod_E 1$ is the E-fold coproduct of the terminal object in $\mathscr{E}$ (which may be different from the monoidal unit I).

For the case $\mathscr{E} = \langle \mathrm{ENDO}_\kappa(\mathscr{C}), \circ, \mathrm{Id} \rangle$, the equational system ET_E describes (κ-accessible) monads $M : \mathscr{C} \to \mathscr{C}$ equipped with a natural transformation

$$throw : (\coprod_E 1) \circ M = \coprod_E (1 \circ M) = \coprod_E 1 \quad \longrightarrow \quad M \tag{5.3}$$

whose component $1 \to M$ for each $e \in E$ represents a computation throwing an exception e. Working in the generality of monoids allows us to generalise exceptions to more settings: taking $\mathscr{E} = \langle \mathrm{ENDO}_\kappa(\mathrm{SET}), *, \mathrm{Id} \rangle$, the theory describes applicatives F with exception throwing:

$$(\coprod_E 1) * F \cong \coprod_E (1 * F) = \coprod_E (\textstyle\int^{a,b} 1a \times Fb \times -^{a \times b}) \cong \coprod_E (\int^b Fb) \quad \to \quad F \tag{5.4}$$

Note that exception throwing for monads (5.3) and for applicatives (5.4) differ by the domain: 1 vs $\int^b Fb$. This reflects the nature of applicative functors that computations are independent, so the computation after exception throwing is not necessarily discarded.

5∗11. Although exception *throwing* is an algebraic operation, exception *catching* is not: if we were to model it as an operation $catch : M \times M \to M$ on a monad M such that $catch \ \langle p, h \rangle$ means catching exceptions possibly thrown by p and handling exceptions using h, then equation (5.2) for $\Sigma M = M \times M$ with the isomorphism $(M \times M) \circ Y \cong (M \circ Y) \times (M \circ Y)$ as the strength implies that

$$ph : M \times M, k : M \vdash \mu(catch \, ph, k) = catch \ \langle \mu(\pi_1 \, ph, \ k), \mu(\pi_2 \, ph, \ k) \rangle : M \tag{5.5}$$

But this is undesirable because the scopes of catching on the two sides are different: the left-hand side does not catch exceptions in k but the right-hand side catches exceptions in k.

5∗12. Plotkin and Pretnar's [2009; 2013] take on this problem is that catching is inherently different from throwing: throwing is the only *operation* of the algebraic theory of computation with exceptions, but catching is a *model* of the theory. This view leads to the fruitful line of research on *handlers of algebraic effects*.

Wu et al. (2014) proposed an alternative perspective: *catch* is still an operation in the theory of monads supporting the effect of exceptions, but its signature should be $(M \times M) \circ M \to M$ rather than $M \times M \to M$. Their perspective can be justified by the following observation.

5∗13 Lemma. Let $\langle \mathscr{E}, \square, I \rangle$ be a monoidal category. For all functors $\Phi : \mathscr{E} \to \mathscr{E}$ and monoids $\langle M, \mu, \eta \rangle$ in $\mathscr{E}$, define $\Sigma = (\Phi -) \square -$ and a pointed strength:

$$(\Sigma X) \square Y \xrightarrow{\cong} \Phi(X \square I) \square (X \square Y) \xrightarrow{\Phi(X \square \eta^Y) \square id} \Phi(X \square Y) \square (X \square Y) = \Sigma(X \square Y).$$

Then morphisms $f : \Phi M \to M$ *without any condition* are in bijection with morphisms $g : \Sigma M \to M$ satisfying the compatibility equation (5.2) of Σ-monoids instantiated with Σ.

Proof. The bijection between f and g is given by

$$f \mapsto (\Phi M \square M \xrightarrow{f \square M} M \square M \xrightarrow{\mu} M) \qquad g \mapsto (\Phi M \xrightarrow{\Phi M \square \eta} \Phi M \square M \xrightarrow{g} M).$$

The reader is encouraged to use the internal language of $\mathscr{E}$ to check the round-trip property of this bijection, and below is a proof using the traditional notation. If we start with $f : \Phi M \to M$, after a round-trip we get the morphism

$$\Phi M \xrightarrow{\Phi M \square \eta} \Phi M \square M \xrightarrow{f \square M} M \square M \xrightarrow{\mu} M,$$

which, by the naturality of η, is equal to $(\mu \cdot \eta) \cdot f = f$. If we start with a morphism $g : \Sigma M \to M$ satisfying (5.2), after a round-trip we get the morphism

$$\Phi M \square M \xrightarrow{\Phi M \square \eta \square M} \Phi M \square M \square M \xrightarrow{g \square M} M \square M \xrightarrow{\mu} M. \tag{5.6}$$

By the definition of the pointed strength of Σ above, equation (5.2) instantiates to the equation of the following two morphisms $\Phi M \square M \square M \to M$:

$$\mu \cdot (g \square M) = g \cdot (\Phi \mu \square \mu) \cdot (\Phi(M \square \eta) \square M \square M)$$

The right-hand side is equal to $g \cdot (\Phi M \square \mu)$, so (5.6) is equal to $g \cdot (\Phi M \square \mu) \cdot (\Phi M \square \eta \square M) = g \cdot (\Phi M \square (\mu \cdot (\eta \square M))) = g \cdot (\Phi M \square id) = g$. □

5∗14. Therefore, using Lemma 5∗13, exception catching on a monad can still be formulated as a Σ-monoid with $\Sigma M := (M \times M) \circ M = (\mathrm{Id} \times \mathrm{Id}) \circ M \circ M$. More generally, operations $s : A \circ M \circ M \to M$ satisfying (5.2) for an endofunctor A, or equivalently $A \circ M \to M$ without (5.2), are called *scoped operations* by Wu et al. (2014) and Piróg et al. (2018).

The name 'scoped operations' refers to the intuition that the operation s genuinely take computations as its argument, so there is a meaningful boundary between its argument and the future continuation, so its arguments are 'in the scope of s'. This is in contrast with algebraic operations $a : B \circ M \to M$ satisfying (5.2), which are equivalently $B \to M$ by 5∗13, so they do not genuinely take computations as input although they are usually presented in the form of $B \circ M \to M$.

5∗15 Example (Exception catching). Now let us come back the example of exception catching. Let $\mathscr{C}$ be a category with finite products and coproducts. Exception throwing and catching can be modelled as the theory of Σ_{Ec}-monoids in $\langle \textsc{Endo}(\mathscr{C}), \circ, \mathrm{Id} \rangle$, where the signature functor $\Sigma_{\mathrm{Ec}} : \mathscr{E} \to \mathscr{E}$ is

$$\Sigma_{\mathrm{Ec}} = ((\mathrm{Id} \times \mathrm{Id}) \circ - \circ -) + (1 \circ -)$$

equipped with the following pointed strength for all $X \in \mathscr{E}$ and $\langle Y, f \rangle : I/\mathscr{E}$:

$$\begin{aligned}(\Sigma_{\mathrm{Ec}} X) \circ Y &= \big(((\mathrm{Id} \times \mathrm{Id}) \circ X \circ X) + (1 \circ X)\big) \circ Y \\ &\cong ((\mathrm{Id} \times \mathrm{Id}) \circ X \circ X \circ Y) + (1 \circ X \circ Y) \\ &\to ((\mathrm{Id} \times \mathrm{Id}) \circ X \circ \boxed{Y} \circ X \circ Y) + (1 \circ X \circ Y) \cong \Sigma_{\mathrm{Ec}}(X \circ Y)\end{aligned}$$

where the boxed Y is inserted using $f : I \to Y$.

The intuition for the signature Σ_{Ec} is that the first operation $1 \circ M \to M$ is *throw*ing an exception as in Example 5∗10, and the second operation

$$\mathit{catch} : (\mathrm{Id} \times \mathrm{Id}) \circ M \circ M \cong (M \times M) \circ M \quad \longrightarrow \quad M \tag{5.7}$$

is catching. As we explained above, the trick here to avoid the undesirable equation (5.5) is that *catch* has after $M \times M$ an additional $- \circ M$ that represents an *explicit continuation* after the scoped operation *catch* (Piróg et al., 2018): $\mathit{catch}\ (\langle p, h\rangle, k)$ is understood as handling the exception in p with h and then continuing as k. Then equation (5.2) instantiates to

$$ph : M \times M, k : M, k' : M \vdash \mu(\mathit{catch}\ (ph, k), k') = \mathit{catch}\ (ph, \mu(k, k')) : M. \tag{5.8}$$

Unlike (5.5), this equation is reasonable for the theory of exceptions: catching ph and then doing k and then k' should be the same as catching ph and then continuing as $\mu(k, k')$. The scope of *catch* is not confused.

5∗16. Moreover, we can add equations to the theory Σ_{Ec}-Mon to characterise the interaction of *throw* and *catch*. The theory Ec is Σ_{Ec}-Mon extended with the following equations:

$$k : \tau \vdash \mathit{catch}(\langle \mathit{throw}, \eta\rangle, k) = k : \tau \qquad k : \tau \vdash \mathit{catch}(\langle \mathit{throw}, \mathit{throw}\rangle, k) = \mathit{throw} : \tau$$
$$k : \tau \vdash \mathit{catch}(\langle \eta, \mathit{throw}\rangle, k) = k : \tau \qquad k : \tau \vdash \mathit{catch}(\langle \eta, \eta\rangle, k) = k : \tau$$

where $\eta : I \to \tau$, $\mathit{throw} : 1 \to \tau$, and $\mathit{catch} : (\tau \times \tau) \,\square\, \tau \to \tau$. These equations can be alternatively presented with an empty context by replacing all the k's with η as in $\cdot \vdash \mathit{catch}(\langle \mathit{throw}, \eta\rangle, \eta) = \eta : \tau$, which is equivalent to the first equation above, since by (5.8), $\mathit{catch}(\langle x, y\rangle, \eta); k = \mathit{catch}(\langle x, y\rangle, k)$.

5∗17 Remark. By modelling scoped operations like *catch* as genuine operations rather than handlers, they enjoy the usual benefits of algebraic effects: we can impose equational axioms on them, consider the free models, combine their theories with effects, etc. But the reader may still wonder – from a purely practical programming perspective, what do we gain by modelling them as scoped operations rather than as effect handlers? Towards this question, Wu et al. (2014) argued that there is a problem of *compositionality* if we model operations like *catch* as effect handlers. Wu et al. used a concrete example of non-deterministic parsing with effect handlers, and Yang et al. (2022) clarified this problem using exception catching. Here, we explain this problem again using a small example about parallel composition as either a handler or an operation in its own right; c.f. Castellano et al. (1987).

Implementing parallel composition as a handler is similar to *interleaving concurrency* in concurrency theory: the parallel composition $P \,||_i\, Q$ of two processes P and Q is handled/reduced to the nondeterministic choice of all the ways of interleaving the actions of P and Q. For example, if $P := a.0$ is the process that performs an action a and stops, and $Q := b.0$ is the process that performs an action b and stops, then we have

$$P \,||_i\, Q = (a.b.0) + (b.a.0).$$

A problem arises if we want to lower the level of abstraction by refining the action a into two actions a_1 and a_2 (e.g. we may want to refine the action of writing a memory location into several actions when we move down the level of abstractions from an abstract memory model to a more realistic memory model). We may define this refinement as a meta-operation r on programs that replaces every action a with two actions a_1 and a_2. The

expected outcome of 'parallel composition of P and Q' with actions refined is

$$r(P) \,||_i\, r(Q) = (a_1.a_2.0) \,||_i\, (b.0) = (a_1.a_2.b.0) + (a_1.b.a_2.0) + (b.a_1.a_2.0),$$

but refining $P \,||_i\, Q$ above gives us

$$r(P \,||_i\, Q) = r((a.b.0) + (b.a.0)) = (a_1.a_2.b.0) + (b.a_1.a_2.0),$$

which is not the expected outcome in this scenario.

The problem here is precisely that if we treat parallel composition as a handler, then the scope of parallel composition is lost after handling. On the other hand, if we treat parallel composition as an operation in its own right, parallel composition of P and Q is *just* $P \,||\, Q$ (with perhaps some axioms), so

$$r(P \,||\, Q) = (a_1.a_2.0) \,||\, b.0 = r(P) \,||\, r(Q).$$

Therefore, if we model parallel composition as a handler $||_i$, we must carefully ensure that the refinement operation r is applied to the processes P and Q before applying $||_i$ to them, whereas if we model parallel composition a scoped operation $||$, we can apply the refinement operation r to processes built with $||$, which is a more compositional approach.

5∗18. There are many more examples of Σ-monoids that we cannot expand on here. Some interesting ones are lambda abstraction (Fiore et al., 1999), the algebraic operations of π-calculus (Stark, 2008), and the non-algebraic operation of parallel composition (Piróg et al., 2018). A collection of interesting programming examples can be found in (van den Berg and Schrijvers, 2024).

6 Families of Operations

6∗1. Given a monoidal category $\mathscr{E}$, the coslice category $\textsc{Mon}/\textsc{Eqs}(\mathscr{E})$ contains all equational systems that extend the theory of monoids with new operations/equations. However, this category is sometimes too general – we will see later that there are many constructions that only work for a certain kind of operations, such as algebraic operations or scoped operations. Therefore, we will need to consider subcategories of $\textsc{Mon}/\textsc{Eqs}(\mathscr{E})$ that contain equational systems that extend $\textsc{Mon}$ with a certain family of operations. In this section, let us have a look at some important operation families, and it turns out that there are quite some interesting things to be said about them.

6∗2 Definition. An *operation family* on monoids in a monoidal category $\mathscr{E}$ is a subcategory $\mathcal{F} \subseteq \textsc{Mon}/\textsc{Eqs}(\mathscr{E})$ of the coslice category under the theory of monoids.

6∗3. An object $\ddot{\Sigma}$ in an operation family $\mathcal{F}$ is a pair $\langle \dot{\Sigma}, T_\Sigma : \textsc{Mon} \to \dot{\Sigma} \rangle$, but we will colloquially say *something* of $\ddot{\Sigma}$ to mean that thing of $\dot{\Sigma}$. For example, when we say $\ddot{\Sigma}$ has the free-forgetful adjunction, we mean that $\dot{\Sigma}$ has it.

6∗4 (Algebraic operations). The simplest example is the family $\textsc{Alg}(\mathscr{E})$ of *algebraic operations* on a monoidal category $\langle \mathscr{E}, \square, I \rangle$ with binary coproducts. The full subcategory $\textsc{Alg}(\mathscr{E}) \subseteq \textsc{Mon}/\textsc{Eqs}(\mathscr{E})$ contains objects of the following form

$$\langle\, (A \,\square\, -)\text{-}\textsc{Mon}^{\,\lnot}\, (\mathrm{K}_B \vdash L = R), \quad T \,\rangle \tag{6.1}$$

where $A, B \in \mathscr{E}$; the functor $A \square -$ is equipped with the pointed strength

$$\alpha_{A,X,Y} : (A \square X) \square Y \cong A \square (X \square Y);$$

T is the inclusion translation from Mon; $\mathrm{K}_B : \mathscr{E} \to \mathscr{E}$ is the constant functor mapping to B. In other words, $\textsc{Alg}(\mathscr{E})$ contains all equational systems Σ-Mon in 5∗4 extended with an equation for some $\Sigma = A \square -$.

6∗5. In particular, the theory of exceptions (Example 5∗10) and state (Example 5∗9) are in $\textsc{Alg}(\mathscr{E})$. When $\mathscr{E} = \langle \textsc{Endo}_\kappa(\mathscr{C}), \circ, \mathrm{Id}\rangle$ for an lκp category $\mathscr{C}$, $\textsc{Alg}(\mathscr{E})$ consists of theories of algebraic operations $A \circ M \to M$ for $A \in \textsc{Endo}_\kappa(\mathscr{C})$ on κ-accessible monads M. When $\mathscr{E}$ is $\langle \textsc{Endo}_\kappa(\textsc{Set}), *, \mathrm{Id}\rangle$, it then contains theories of applicatives F with 'applicative-algebraic' operations $A * F \to F$.

6∗6. When the monoidal category $\mathscr{E}$ is *right distributive* for binary coproducts, which means that the canonical morphism

$$[\iota_1 \square C, \iota_2 \square C] : (A \square C + B \square C) \to (A + B) \square C$$

is an isomorphism, the category $\textsc{Alg}(\mathscr{E})$ has binary coproducts as well: binary coproducts in $\textsc{Alg}(\mathscr{E})$ are equivalently pushouts in $\textsc{Eqs}(\mathscr{E})$, and by Lemma 3.3∗5, such a pushout still has a constant context $\mathrm{K}_B + \mathrm{K}_{B'} = \mathrm{K}_{B+B'}$ and a signature $(A \square -) + (A' \square -) \cong (A + A') \square -$, so it is still in $\textsc{Alg}(\mathscr{E})$.

6∗7. Many of the monoidal categories in Section 2 are right distributive:

- $\langle \textsc{Endo}(\mathscr{C}), \circ, \mathrm{Id}\rangle$ for an small-complete small $\mathscr{C}$;
- $\langle \textsc{Endo}_\kappa(\mathscr{C}), \circ, \mathrm{Id}\rangle$ for an lκp $\mathscr{C}$;
- $\langle \mathscr{C}, \times, 1\rangle$ for a cocomplete cartesian closed $\mathscr{C}$;
- $\langle \textsc{Endo}_\kappa(\textsc{Set}), *, \mathrm{Id}\rangle$ underlying applicative functors;
- $\langle \textsc{Endo}_{s\kappa}(\mathscr{C}), \circ_s, \mathrm{Id}_s\rangle$ for an lκp as a cartesian closed category $\mathscr{C}$;
- $\langle \textsc{Endo}_f(\textsc{Set})^{\mathscr{G}}, *, I\rangle$ for a small strict monoidal category $\mathscr{G}$.

Moreover, for these choices of $\mathscr{E}$, every object of $\textsc{Alg}(\mathscr{E})$ has the free-forgetful adjunction using the freeness conditions in Section 3.1.

6∗8. The restriction in (6.1) that the context of the equation must be a constant functor K_B deserves some explanation. Let $\Sigma : \mathscr{C} \to \mathscr{C}$ be an endofunctor on a category $\mathscr{C}$. We call a functorial equation $\mathrm{K}_B \vdash L = R$ over the signature Σ with a constant functor as its context a *constant equation*. Assume that $\mathscr{C}$ is locally small and small-complete, and that the functor Σ has the free-forgetful adjunction $\mathrm{F}_\Sigma \dashv \mathrm{U}_\Sigma$. By Corollary 3.2∗7, the equation $\mathrm{K}_B \vdash L = R$ is equivalently a pair of natural transformations $\mathrm{K}_B \rightrightarrows \mathrm{F}_\Sigma\mathrm{U}_\Sigma$. Moreover, when $\mathscr{C}$ has an initial object 0, such a natural transformation is uniquely determined by its component at 0:

$$\begin{array}{ccc} \mathrm{K}_B 0 = B & \xlongequal{id_B} & \mathrm{K}_B X = B \\ {\scriptstyle L_0}\downarrow & & \downarrow{\scriptstyle L_X} \\ \mathrm{U}_\Sigma\mathrm{F}_\Sigma 0 & \xrightarrow[\mathrm{U}_\Sigma\mathrm{F}_\Sigma f]{} & \mathrm{U}_\Sigma\mathrm{F}_\Sigma X \end{array}$$

Therefore constant equations $\mathrm{K}_B \vdash L = R$ over the signature $\Sigma : \mathscr{C} \to \mathscr{C}$ are in bijection with pairs of $\mathscr{C}$-morphisms $L', R' : B \to \mathrm{U}_\Sigma \mathrm{F}_\Sigma 0$, whose codomain is precisely (the carrier of) the initial algebra $\mu\Sigma$ of Σ.

6∗9. More specially, assume that a monoidal category $\mathscr{E}$ satisfies the conditions on $\mathscr{C}$ in the last paragraph, and that $\Box : \mathscr{E} \times \mathscr{E} \to \mathscr{E}$ preserves colimits of α-chains for some limit ordinal α. In this case, let Σ be the signature functor of the equational system $(A \Box -)$-Mon, i.e. $\Sigma = I + - \Box - + A \Box -$. Then both Σ and the equational system $(A \Box -)$-Mon have initial algebras, whose carriers we denote by X and Y respectively. For every algebra $\langle Z, \alpha \rangle$ of $(A \Box -)$-Mon, it satisfies $\mathrm{K}_B \vdash L = R$ if and only if the following diagram commutes:

$$B \underset{r}{\overset{l}{\rightrightarrows}} X \xrightarrow{!_\Sigma} Z$$

where the morphism $!_\Sigma : X \to Z$ is the unique Σ-homomorphism from X to Z. Since the algebras of $(A \Box -)$-Mon is a full subcategory of Σ-algebras, the morphism $!_\Sigma$ factors via $!_{(A\Box-)\text{-Mon}} : Y \to Z$, the unique homomorphism from the initial algebra of $(A \Box -)$-Mon:

$$B \underset{r}{\overset{l}{\rightrightarrows}} X \xrightarrow{!_\Sigma} Y \xrightarrow{!_{(A\Box-)\text{-Mon}}} Z$$

Therefore, equations $\mathrm{K}_B \vdash L = R$ can be given as a pair of morphisms $B \rightrightarrows Y$ without loss of expressivity. In particular, if $\mathscr{E}$ is $\langle \mathrm{End}_f(\mathrm{Set}), \circ, \mathrm{Id} \rangle$, Y is precisely the free monad A^* over A, a pair of morphisms $B \rightrightarrows A^*$ for $A, B \in \mathrm{End}_f(\mathrm{Set})$ is indeed the traditional way of presenting a finitary algebraic theory. In fact, we can show that the category $\mathrm{Alg}(\mathscr{E})$ is equivalent to the category (as defined by e.g. Fiore and Mahmoud (2014)) of presentations finitary algebraic theories and translations. Therefore, although constant equations seem very restrictive, they are still useful enough when the monoidal category $\mathscr{E}$ itself is informative.

6∗10 Theorem. *Let $\mathscr{E}$ be a monoidal category with binary coproducts such that every object in* $\mathrm{Alg}(\mathscr{E})$ *has the free-forgetful adjunction. There is an equivalence* $\mathrm{Alg}(\mathscr{E}) \cong \mathrm{Mon}(\mathscr{E})$ *between* $\mathrm{Alg}(\mathscr{E})$ *and the category* $\mathrm{Mon}(\mathscr{E})$ *of monoids in $\mathscr{E}$.*

Proof. In sketch, every $\ddot{\Sigma} \in \mathrm{Alg}(\mathscr{E})$ is mapped to its initial algebra treated as a monoid, forgetting the operation. On the other hand, every monoid M is mapped to the theory of *M-actions on monoids*.

In more detail, the direction $\mathrm{Alg}(\mathscr{E}) \to \mathrm{Mon}(\mathscr{E})$ of the equivalence sends every object $\ddot{\Sigma} = \langle \dot{\Sigma}, T_\Sigma \rangle$ in $\mathrm{Alg}(\mathscr{E})$ to the initial algebra $\langle \mu\dot{\Sigma}, \alpha^\Sigma \rangle$ regarded as a monoid $T_\Sigma \langle \mu\dot{\Sigma}, \alpha^\Sigma \rangle$. On morphisms, every translation $T : \ddot{\Sigma} \to \ddot{\Psi} \in \mathrm{Alg}(\mathscr{E})$ induces a unique $\ddot{\Sigma}$-homomorphism out of the initial algebra:

$$h : \langle \mu\dot{\Sigma}, \alpha^\Sigma \rangle \to T \langle \mu\dot{\Psi}, \alpha^\Psi \rangle.$$

Then the arrow mapping is $T \mapsto T_\Sigma h$ where

$$T_\Sigma h : T_\Sigma \langle \mu\dot{\Sigma}, \alpha^\Sigma \rangle \to T_\Sigma (T \langle \mu\dot{\Psi}, \alpha^\Psi \rangle) = T_\Psi \langle \mu\dot{\Psi}, \alpha^\Psi \rangle.$$

The equality $T_\Sigma \circ T = T_\Psi$ is by the definition of morphisms in $\mathrm{Alg}(\mathscr{E})$.

For the other direction, every monoid $\dot{M} = \langle M, \mu^M, \eta^M \rangle$ in $\mathscr{E}$ is sent to the theory $\dot{M}$-Act of *$\dot{M}$-actions on monoids*, which is the theory of $(M \Box -)$-Mon extended with the following

two equations (expressed as an extension of the internal language of $\mathscr{E}$ as in 4.3∗14):

$$\vdash op(\eta^M, \eta^\tau) = \eta^\tau : \tau$$
$$x : M, y : M \vdash op(\mu^M(x, y), \eta^\tau) = op(x, op(y, \eta^\tau)) : \tau$$

saying that $op : M \,\square\, \tau \to \tau$ is a monoid action on τ. Every monoid morphism $f : \dot{M} \to \dot{N}$ is mapped to the translation $\dot{M}$-Act $\to \dot{N}$-Act sending $\dot{N}$-actions

$$\langle A \in \mathscr{E}, \alpha : (N \,\square\, A) + \Sigma_{\text{Mon}} A \to A \rangle$$

to $\dot{M}$-actions $\langle A, [\alpha \cdot \iota_1 \cdot (f \,\square\, A), \alpha \cdot \iota_2] : (M \,\square\, A) + \Sigma_{\text{Mon}} A \to A \rangle$.

It remains to show that the mappings above are a pair of equivalence. Starting from a monoid $\dot{M}$, it can be shown that the category ($\dot{M}$-Act)-Alg is equivalent to the coslice category $\dot{M}/\text{Mon}(\mathscr{E})$ (see e.g. Fiore and Saville (2017, Proposition 5.5)). Thus the initial algebra of $\dot{M}$-Act is $\dot{M}$ as required.

Starting from a theory $\langle \dot{\Sigma}, T_\Sigma \rangle \in \text{Alg}(\mathscr{E})$ where

$$\dot{\Sigma} = (S \,\square\, -)\text{-Mon} \upharpoonleft (\text{K}_B \vdash L = R),$$

it is mapped to the monoid $T_\Sigma \langle \mu\dot{\Sigma}, \alpha^\Sigma \rangle$, which is then mapped back to the theory $\mu\dot{\Sigma}$-Act with the inclusion translation. We need to construct an isomorphism translation $T : \dot{\Sigma} \to \mu\dot{\Sigma}$-Act that preserves monoid operations. Given a monoid A with $\alpha : \mu\dot{\Sigma} \,\square\, A \to A$ satisfying the laws of $\mu\dot{\Sigma}$-Act, T maps it to the $\dot{\Sigma}$-algebra on A with the following operation:

$$S \,\square\, A \xrightarrow{S \square \eta^{\mu\dot{\Sigma}} \square A} S \,\square\, \mu\dot{\Sigma} \,\square\, A \xrightarrow{\alpha^{\mu\dot{\Sigma}} \square A} \mu\dot{\Sigma} \,\square\, A \xrightarrow{\alpha} A$$

where $\alpha^{\mu\dot{\Sigma}} : S \,\square\, \mu\dot{\Sigma} \to \mu\dot{\Sigma}$ is the structure map of the initial algebra.

For the inverse of T, every tuple $\langle A, \beta : S \,\square\, A \to A \rangle \in \dot{\Sigma}$-Alg is mapped to the following $\mu\dot{\Sigma}$-Act-algebra on A:

$$\mu\dot{\Sigma} \,\square\, A \xrightarrow{(\!|\beta|\!) \square A} A \,\square\, A \xrightarrow{\mu^A} A$$

where $(\!|\beta|\!)$ is the unique homomorphism from the initial $\dot{\Sigma}$-algebra $\mu\dot{\Sigma}$ to the $\dot{\Sigma}$-algebra $\langle A, \beta \rangle$. This completes the proof. □

6∗11. Instantiating $\mathscr{E}$ with $\langle \text{Endo}_f(\mathscr{C}), \circ, \text{Id} \rangle$ for an lfp $\mathscr{C}$, we obtain an equivalence between finitary monads over $\mathscr{C}$ and theories of algebraic operations on finitary monads. This is reminiscent of the classical *theory-monad correspondence* between finitary monads and (presentations of) first-order algebraic theories. What is new is that Theorem 6∗14 is applicable to other monoidal categories, such as those in Section 2, giving us equivalences of cartesian monoids/applicative functors/graded monads and the corresponding categories of theories of algebraic operations.

6∗12. Another interesting property of Alg($\mathscr{E}$) is the following saying that (almost) all equational systems of operations on monoids can be turned into one in Alg($\mathscr{E}$) by a coreflection, and the coreflection preserves initial algebras, i.e. the abstract syntax of terms of operations. Hence in principle, theories of algebraic operations alone are sufficient for the purpose of modelling syntax.

6∗13 Lemma. Let $\mathscr{E}$ be a monoidal category with binary coproducts. For every $\ddot{\Psi} \in \text{Alg}(\mathscr{E})$ and $\ddot{\Sigma} \in \text{Mon}/\text{Eqs}(\mathscr{E})$ such that both of them have initial algebras, the set $\text{Mon}/\text{Eqs}(\ddot{\Psi}, \ddot{\Sigma})$

is in natural bijection to monoid morphisms $\mu\ddot{\Psi} \to \mu\ddot{\Sigma}$ between the initial algebras of $\ddot{\Psi}$ and $\ddot{\Sigma}$ viewed as monoids.

Proof sketch. For one direction of the bijection ϕ, every translation $T : \ddot{\Psi} \to \ddot{\Sigma}$ sends the initial algebra $\mu\ddot{\Sigma}$ of $\ddot{\Sigma}$ to a $\ddot{\Psi}$-algebra carried by $\mu\ddot{\Sigma}$. Then by the initiality of $\mu\ddot{\Psi}$, there is a $\ddot{\Psi}$-homomorphism, which is also a monoid morphism, $u : \mu\ddot{\Psi} \to \mu\ddot{\Sigma}$. We set $\phi(T) = u$. For the backward direction of the bijection ϕ, given a monoid morphism $h : \mu\ddot{\Psi} \to \mu\ddot{\Sigma}$, we define a translation $T : \ddot{\Psi} \to \ddot{\Sigma}$, i.e. a functor $T : \ddot{\Sigma}\text{-}\textsc{Alg} \to \ddot{\Psi}\text{-}\textsc{Alg}$ as follows. Recall that $\ddot{\Psi} \in \textsc{Alg}(\mathscr{E})$ must be of the form $\ddot{\Psi} = (G \,\square\, -)\text{-}\textsc{Mon} \upharpoonleft (\mathrm{K}_B \vdash L = R)$, for some $G \in \mathscr{E}$. The functor T maps every $\ddot{\Sigma}$-algebra $\langle A, \alpha : \Sigma A \to A\rangle$ to the $\ddot{\Psi}$-algebra carried by A with

$$G \,\square\, A \xrightarrow{G\square\eta^{\mu\ddot{\Psi}}} G \,\square\, \mu\ddot{\Psi} \,\square\, A \xrightarrow{\alpha^{\mu\ddot{\Psi}}} \mu\ddot{\Psi} \,\square\, A \xrightarrow{h} \mu\ddot{\Sigma} \,\square\, A \xrightarrow{(\!|\alpha|\!)} A \,\square\, A \xrightarrow{\mu^A} A$$

where $\alpha^{\mu\ddot{\Psi}} : G \,\square\, \mu\ddot{\Psi} \to \mu\ddot{\Psi}$ is the structure map of the initial $\ddot{\Psi}$-algebra, $(\!|\alpha|\!) : \mu\ddot{\Sigma} \to A$ is the unique $\ddot{\Sigma}$-homomorphism from the initial algebra $\mu\ddot{\Sigma}$ to $\langle A, \alpha\rangle$. It can be checked that ϕ is a natural bijection. □

6∗14 Theorem. *Let $\mathscr{E}$ be a monoidal category with binary coproducts such that every object of $\textsc{Alg}(\mathscr{E})$ has the free-forgetful adjunction. The category $\textsc{Alg}(\mathscr{E})$ is a coreflective subcategory of $\textsc{Mon}/\textsc{Eqs}_f(\mathscr{E})$, where $\textsc{Eqs}_f(\mathscr{C}) \subseteq \textsc{Eqs}(\mathscr{C})$ is the full subcategory containing equational systems with the free-forgetful adjunction*:

$$\textsc{Alg}(\mathscr{E}) \;\underset{\tau}{\overset{\lfloor - \rfloor}{\leftrightarrows}}\; \textsc{Mon}/\textsc{Eqs}_f(\mathscr{E}).$$

Moreover, the coreflector $\lfloor - \rfloor$ preserves initial algebras: for every $\langle \dot{\Sigma}, T\rangle$ in the category $\textsc{Mon}/\textsc{Eqs}_f(\mathscr{E})$, the initial $\dot{\Sigma}$-algebra (viewed as a monoid via T) is isomorphic to the initial algebra of $\lfloor \langle \dot{\Sigma}, T\rangle \rfloor$ also viewed as a monoid.

Proof sketch. Every theory $\langle \dot{\Sigma}, T_\Sigma\rangle \in \textsc{Mon}/\textsc{Eqs}_f(\mathscr{E})$ has an initial algebra $\mu\dot{\Sigma} \in \mathscr{E}$ by assumption, and $\mu\dot{\Sigma}$ carries a monoid structure by $T_\Sigma : \textsc{Mon} \to \dot{\Sigma}$. We define the coreflector $\lfloor - \rfloor$ to map every $\langle \dot{\Sigma}, T_\Sigma\rangle$ to $\mu\dot{\Sigma}\text{-}\textsc{Act} \in \textsc{Alg}(\mathscr{E})$ as in the proof of Theorem 6∗10. For every theory $\langle \dot{\Psi}, T_\Psi\rangle \in \textsc{Alg}(\mathscr{E})$, by Lemma 6∗13, each translation in the hom-set $\textsc{Mon}/\textsc{Eqs}_f(\langle \dot{\Psi}, T_\Psi\rangle, \langle \dot{\Sigma}, T_\Sigma\rangle)$ is equivalently a monoid morphism $\mu\dot{\Psi} \to \mu\dot{\Sigma}$, which is also equivalently a translation in $\textsc{Alg}(\langle \dot{\Psi}, T_\Psi\rangle, \lfloor \langle \dot{\Sigma}, T_\Sigma\rangle \rfloor)$ by Theorem 6∗10.

The coreflector maps each $\langle \dot{\Sigma}, T\rangle \in \textsc{Mon}/\textsc{Eqs}_f(\mathscr{E})$ to the theory $\mu\dot{\Sigma}\text{-}\textsc{Act}$. It can be shown that the category of algebras of $\mu\dot{\Sigma}\text{-}\textsc{Act}$ is equivalent to the coslice category $\mu\dot{\Sigma}/\textsc{Mon}(\mathscr{E})$ of monoids under $T\langle \mu\dot{\Sigma}, \alpha^{\Sigma}\rangle$, so the initial algebra of $\mu\dot{\Sigma}\text{-}\textsc{Act}$ is still the monoid $\mu\dot{\Sigma}$. □

6∗15. Although $\textsc{Alg}(\mathscr{E})$ is sufficient for modelling syntax, it is *not* enough when we also consider models. The counit of the coreflection gives us a translation $\lfloor \langle \dot{\Sigma}, T\rangle \rfloor \to \langle \dot{\Sigma}, T\rangle$, i.e. a functor $\dot{\Sigma}\text{-}\textsc{Alg} \to \lfloor \langle \dot{\Sigma}, T\rangle \rfloor\text{-}\textsc{Alg}$, but this does not have to be an equivalence of categories.

6∗16 (Scoped operations). Our next example of operation families is the family $\textsc{Scp}(\mathscr{E})$ of *scoped (and algebraic) operations*, such as exception catching (Example 5∗15). Let $\mathscr{E}$ be a monoidal category with right distributive binary coproducts (6∗6). The family $\textsc{Scp}(\mathscr{E})$ is the full subcategory of $\textsc{Mon}/\textsc{Eqs}(\mathscr{E})$ containing objects

$$\langle\, ((A \,\square\, - \,\square\, -) + (B \,\square\, -))\text{-}\textsc{Mon} \upharpoonleft (\mathrm{K}_C \vdash L = R), \quad T \,\rangle \tag{6.2}$$

where $A, B, C \in \mathscr{E}$ and T is the inclusion translation. Letting $\Sigma := (A \square - \square -) + (B \square -)$, the pointed strength $\theta_{X,\langle Y,f\rangle}$ for Σ is the following composite:

$$\begin{aligned}(\Sigma X) \square Y &= (A \square X \square X + B \square X) \square Y \\ &\cong (A \square X \square X \square Y) + (B \square X \square Y) \\ &\to (A \square X \square \boxed{Y} \square X \square Y) + (B \square X \square Y) \\ &\cong \Sigma(X \square Y)\end{aligned}$$

where the boxed Y is inserted using $f : I \to Y$.

In the example of exceptions in Example 5∗15, the monoidal category $\mathscr{E}$ is $\langle \textsc{Endo}(\mathscr{C}), \circ, \mathrm{Id}\rangle$, and A is $\mathrm{Id} \times \mathrm{Id}$, encoding the scoped operation *catch* that takes in two operands, and B is 1, encoding the algebraic operation *throw* that takes in no operands.

6∗17. When every object of $\textsc{Scp}(\mathscr{E})$ has the free-forgetful adjunction, for example when $\mathscr{E}$ is cocomplete and $\square : \mathscr{E} \times \mathscr{E} \to \mathscr{E}$ preserves colimits of α-chains for some limit ordinal α, a corollary of Theorem 6∗14 is that the initial-algebra preserving coreflection there restricts to

$$\textsc{Alg}(\mathscr{E}) \quad \underset{T}{\overset{\longleftarrow}{\longrightarrow}} \quad \textsc{Scp}(\mathscr{E}).$$

Thus the abstract syntax of programs with scoped operations can be alternatively expressed with only algebraic ones, but as argued by Piróg et al. (2018) and Yang et al. (2022), the models of scoped operations are different from those of the coreflected algebraic operations.

6∗18. The operation family $\textsc{Scp}(\mathscr{E})$ should be more accurately called the family of *scoped and algebraic operations with constant equations*. We can certainly relax the restriction of constant equations to obtain bigger operation families. For example, we may have a family $\textsc{Scp}_1(\mathscr{E})$ that is similar to (6.2) but permits *first-order equations*, meaning that the functorial context can be either a constant functor K_C or a functor $C \square -$ for some $C \in \mathscr{E}$.

For computational effects in practice, it seems constant equations in $\textsc{Endo}_\kappa(\mathscr{C})$ are enough for algebraic operations, evidenced by the examples in (Plotkin and Power, 2002), whereas scoped operations sometimes need first-order equations. For example, a reasonable equation for exception catching $catch : M \times M \to M$ is that it is associative for $\times$:

$$\begin{array}{ccc} M \times M \times M & \xrightarrow{catch \times M} & M \times M \\ {\scriptstyle M \times catch}\downarrow & & \downarrow{\scriptstyle catch} \\ M & \xrightarrow[catch]{} & M \end{array}$$

As an equation in the monoidal category $\langle \textsc{Endo}_\kappa(\mathscr{C}), \circ, \mathrm{Id}\rangle$, this equation is first-order, since the context is $(\mathrm{Id}_{\mathscr{C}} \times \mathrm{Id}_{\mathscr{C}} \times \mathrm{Id}_{\mathscr{C}}) \circ - : \textsc{Endo}(\mathscr{C}) \to \textsc{Endo}(\mathscr{C})$.

6∗19. We did *not* require operation families $\mathcal{F} \subseteq \textsc{Mon}/\textsc{Eqs}(\mathscr{E})$ to be full, so we may also restrict the translations. For example, we can consider only translations that map operations to *operations* (rather than terms in general). Such translations are sometimes called *transliterations* (Arkor, 2022). In particular, we define $\textsc{Scp}_l(\mathscr{E}) \subseteq \textsc{Scp}(\mathscr{E})$ to be the subcategory containing translations

$$T : ((A \square - \square -) + (B \square -))\text{-}\textsc{Mon} \to ((A' \square - \square -) + (B' \square -))\text{-}\textsc{Mon}$$

that sends $\langle M, s : A' \square M \square M, a : B' \square M \to M\rangle$ to $\langle M, s \cdot (f \square M \square M), a \cdot (g \square M)\rangle$ for some morphisms $f : A \to A'$ and $g : B \to B'$ in $\mathscr{E}$.

For example, let $A = \mathrm{Id} \times \mathrm{Id} \times \mathrm{Id}$ and $A' = \mathrm{Id} \times \mathrm{Id}$, corresponding to a ternary operation t and a binary operation b respectively. Then the transliteration given by $f\langle x, y, z\rangle = \langle x, y\rangle$ translates the operation $t(x, y, z)$ to $b(x, y)$, but a transliteration cannot translate $t(x, y, z)$ to $b(b(x, y), z)$ that uses the target operation more than once, nor can it use the monoid structure μ and η in the translation.

6∗20 (Variable-binding operations). Algebraic theories of operations with *variable-bindings* are called *second-order algebraic theories* (Fiore and Hur, 2010; Fiore and Mahmoud, 2010, 2014; Fiore and Szamozvancev, 2022), and they can be formulated as an operation family as follows. For simplicity, we work specially in the monoidal category $\langle \mathrm{Set}^{\mathrm{Fin}}, \bullet, V\rangle$ in 2.2∗2, but it is possible to replace $\mathrm{Set}^{\mathrm{Fin}}$ with $\mathrm{Endo}_\kappa(\mathrm{Set})$ for infinitary syntax or with $\mathrm{Set}^{\mathrm{Ctx}}$ for simply typed syntax given a category Ctx of contexts and renamings.

6∗21. A *binding signature* $\langle O, a\rangle$ consists of a set O of operations and an arity assignment $a : O \to \mathbb{N}^*$ of a sequence of natural numbers to each operation. Each $o \in O$ with $a(o) = \langle n_i\rangle_{1\leqslant i\leqslant k}$ stands for an operation taking k arguments, each binding n_i variables:

$$o((x_{1,1}x_{1,2}\cdots x_{1,n_1}).\, e_1,\ \cdots,\ (x_{k,1}x_{k,2}\cdots, x_{k,n_k}).\, e_k)$$

For example, the binding signature for λ-calculus has two operations $\{app, abs\}$: function application $a(app) = \langle 0, 0\rangle$ has two arguments binding no variables; λ-abstraction $a(abs) = \langle 1\rangle$ has one argument that binds one variable.

A binding signature $\langle O, a\rangle$ determines an endofunctor $[\![O, a]\!]$ on $\mathrm{Set}^{\mathrm{Fin}}$:

$$[\![O, a]\!] = \coprod_{o\in O,\ a(o)=\langle n_i\rangle_{1\leqslant i\leqslant k}} \prod_{1\leqslant i\leqslant k} (-)^{V^{n_i}}$$

where $(-)^{V^{n_i}}$ is the exponential by n_i-fold product of the monoidal unit V. This functor has a pointed strength $\theta_{X,\langle Y,\eta^Y\rangle}$:

$$(\coprod_o \prod_i X^{V^{n_i}}) \bullet Y \cong \coprod_o \prod_i (X^{V^{n_i}} \bullet Y) \xrightarrow{\coprod_o \prod_i t_{o,i}} \coprod_o \prod_i (X \bullet Y)^{V^{n_i}}$$

where $t_{o,i}$ is the adjoint transpose of

$$(X^{V^{n_i}} \bullet Y) \times V^{n_i} \xrightarrow{id\times \eta^Y} (X^{V^{n_i}} \bullet Y) \times (V^{n_i} \bullet Y) \cong (X^{V^{n_i}} \times V^{n_i}) \bullet Y \to X \bullet Y.$$

6∗22. The operation family $\mathrm{Var}(\mathrm{Set}^{\mathrm{Fin}}) \subseteq \mathrm{Mon}/\mathrm{Eqs}(\mathrm{Set}^{\mathrm{Fin}})$ then contains all objects of the following form:

$$\langle [\![O, a]\!]\text{-Mon} \upharpoonleft ([\![P, b]\!] \vdash L = R),\ T\rangle$$

where $\langle O, a\rangle$ and $\langle P, b\rangle$ are two binding signatures and T is still the inclusion translation. The category $\mathrm{Var}(\mathrm{Set}^{\mathrm{Fin}})$ is closed under coproducts by Lemma 3.3∗5 and every object of it has the free-forgetful adjunction because $[\![O, a]\!]$ and $[\![P, b]\!]$ are finitary, which is a consequence of $(-)^V$ being a left adjoint to the right Kan extension Ran_{V+1}, so $(-)^V$ preserves all colimits.

6∗23. Again, the coreflector in Theorem 6∗14 allows us to turn every theory in Var into one with only algebraic operations but isomorphic initial algebras. For example, under the coreflection, the theory Λ of untyped λ-calculus is turned into a theory $\lfloor\Lambda\rfloor$ which has an

ordinary n-ary operation t for *every* λ-term t with n free variables, together with suitable equations. The equational systems Λ and $\lfloor\Lambda\rfloor$ have isomorphic initial algebras (as monoids).

* * *

6∗24. To reflect what we have done so far in this paper:

- We have seen how to present theories of monoids with operations using equational systems and monoidal algebraic theories (Sections 3 to 5);
- We can construct (relative) free algebras on cocomplete categories: Theorem 3.1∗12, Theorem 3.1∗19, Theorem 3.2∗14;
- We can combine such theories modularly using colimits (Section 3.3);
- Theories of monoids with operations can be classified into operation families (Section 6), with the family of algebraic operations playing a special role: Theorem 6∗10 and 6∗14.

These results achieve syntactic modularity for computational effects. In the rest of the paper, we will develop a framework of *modular models*, which will allow us to combine models of existing theories into a model of a combined theory, thus achieving modularity for both syntax and semantics.

7 Modular Constructions of Algebraic Structures

7∗1. In many frameworks of algebraic theories, we can combine smaller theories into bigger ones by taking colimits. This gives us a modular way to design programming languages: language features are defined individually as algebraic theories, which are then combined to form algebraic theories of full-fledged languages. Programming language theory is not only about syntax/theories of languages though. What is usually more interesting is the implementations/models, and it turns out that modularly combining models is significantly harder than modularly combining theories. In the rest of this paper, we propose a formal theory of modularity in algebraic structures and show some concrete examples.

7∗2. Let us begin with some high-level description of the concepts that we will define. Let $\mathscr{T}$ be a category of some notion of algebraic theories, together with a functor $(-)\text{-}\textsc{Alg} : \mathscr{T}^{\mathrm{op}} \to \mathrm{CAT}$ that associates a category of models to every theory in $\mathscr{T}$, such that $\mathscr{T}$ has finite coproducts and each category of models $\Gamma\text{-}\textsc{Alg}$ has an initial object $\mu\Gamma$. For example, $\mathscr{T}$ may be the category $\textsc{Alg}(\mathscr{E})$ or $\textsc{Scp}(\mathscr{E})$ of theories of monoids with algebraic or scoped operations that we saw in Section 6 or the category of (ordinary) algebraic theories.

The motivation for a *modular model* of $\Gamma \in \mathscr{T}$ is that we treat $\Gamma \in \mathscr{T}$ as a *language feature* that can be added into existing programming languages, rather than a complete language on its own. A modular model for Γ is then going to be a family of functors $M_\Sigma : \Sigma\text{-}\textsc{Alg} \to (\Sigma + \Gamma)\text{-}\textsc{Alg}$, (oplax-) natural in $\Sigma \in \mathscr{T}$. In this way, for an arbitrary language $\Sigma \in \mathscr{T}$ and a model $A \in \Sigma\text{-}\textsc{Alg}$ of it, we can add the language feature Γ to the language Σ, giving us the combined language $\Sigma + \Gamma$, and a new model $M_\Sigma A \in (\Sigma + \Gamma)\text{-}\textsc{Alg}$.

7∗3. In the situation above, we may also want to relate the old model $A \in \Sigma\text{-}\textsc{Alg}$ and the new model $M_\Sigma A \in (\Sigma + \Gamma)\text{-}\textsc{Alg}$. For this, we will define an *updater* u for the modular model M to be a family of Σ-homomorphisms $u_{\Sigma,A} : A \to M_\Sigma A$, (oplax-) natural in Σ and A. The

name 'updater' comes from the intuition that a model $A \in \Sigma\text{-}\textsc{Alg}$ is like a compiler for the language Σ; an element a fo A is like a compiled binary file; the morphism $u_{\Sigma,A}$ updates every compiled binary a for the old compiler A to a binary for the new compiler $M_\Sigma A$.

7∗4. Moreover, we notice that coproducts $\Sigma + \Gamma$ are just one particular way to combine algebraic theories, and we could also consider other ways of combining theories, such as the *commutative combination* $\Sigma \otimes \Gamma$, also known as the *tensor* (Hyland et al., 2006), which is the theory $\Sigma + \Gamma$ with additionally equations stating that operations from Σ and operations from Γ are commutative. Motivated by this, we will generalise from $(- + \Gamma) : \mathscr{T} \to \mathscr{T}$ in the definition of modular models to an arbitrary functor $\mathscr{T} \to \mathscr{T}$, better still, a functor $T : \mathscr{T} \to \mathscr{T}'$ where the domain $\mathscr{T}$ and codomain $\mathscr{T}'$ can be different ($\mathscr{T}'$ is also equipped with a functor $(-)\text{-}\textsc{Alg} : \mathscr{T}'^{\mathrm{op}} \to \mathrm{CAT}$). We will then call an (oplax-) natural family of functors $M_\Sigma : \Sigma\text{-}\textsc{Alg} \to (T\Sigma)\text{-}\textsc{Alg}$ a *model transformer* for T.

Model transformers will be shown to be equivalent to *liftings of functors* along fibrations, i.e. functors M making the diagram commute strictly,

$$\begin{array}{ccc} \mathscr{A} & \xrightarrow{M} & \mathscr{A}' \\ {\scriptstyle p}\downarrow & & \downarrow{\scriptstyle p'} \\ \mathscr{T} & \xrightarrow[T]{} & \mathscr{T}' \end{array}$$

where p and p' are the fibrations corresponding to $(-)\text{-}\textsc{Alg} : \mathscr{T}^{\mathrm{op}} \to \mathrm{CAT}$ and $(-)\text{-}\textsc{Alg} : \mathscr{T}'^{\mathrm{op}} \to \mathrm{CAT}$ respectively via the Grothendieck construction. A toolbox of constructions of such model transformers will be developed in 8∗1.

7∗5. The structure of this section is as follows: In Section 7.1, we define modular models of monoids with operations, and establish the correspondence between two formulations based on *indexed categories* and *fibrations* respectively. In Section 7.2, motivated by more scenarios of algebraic structures, we generalise modular models to *model transformers*, which are just a 'name with an attitude' (nLab, 2024) for liftings of functors along fibrations.

7.1 Modular Models of Monoids

7.1∗1 Notation. We will work with indexed categories $\mathscr{C} \to \mathrm{CAT}$ a lot in this subsection, where CAT is the category of large categories, so for convenient when we say 'a category' in this subsection, by default we mean a large category unless otherwise specified. We fix a monoidal category $\mathscr{E}$ and an operation family $\mathcal{F} \subseteq \textsc{Mon}/\textsc{Eqs}(\mathscr{E})$ such that $\mathcal{F}$ is closed under finite coproducts in $\textsc{Mon}/\textsc{Eqs}(\mathscr{E})$. Every object $\ddot{\Sigma} \in \mathcal{F}$ is a pair $\langle \dot{\Sigma} \in \textsc{Eqs}(\mathscr{E}), T : \textsc{Mon} \to \dot{\Sigma} \rangle$. We will write $\ddot{\Sigma}\text{-}\textsc{Alg}$ to mean the category $\dot{\Sigma}\text{-}\textsc{Alg}$ of algebras for $\dot{\Sigma}$.

7.1∗2. In this subsection, we will make precise the idea of *modular models*, as motivated in 7∗2, in two equivalent formulations, one based on *indexed categories* (Definition 7.1∗5), and another based on *fibrations* (Theorem 7.1∗18). The former is more explicit while the latter is more convenient to work with.

7.1∗3. Recall that a morphism $T : \dot{\Sigma} \to \dot{\Psi}$ in $\textsc{Eqs}(\mathscr{E})$ is a functor $\dot{\Psi}\text{-}\textsc{Alg} \to \dot{\Sigma}\text{-}\textsc{Alg}$ such that $\mathrm{U}_{\dot{\Sigma}} \circ T = \mathrm{U}_{\dot{\Psi}} : \dot{\Psi}\text{-}\textsc{Alg} \to \mathscr{E}$, so we can treat $(-)\text{-}\textsc{Alg}$ as a functor $\textsc{Eqs}(\mathscr{E})^{\mathrm{op}} \to \mathrm{CAT}$. Similarly, we have a functor $(-)\text{-}\textsc{Alg} : \mathcal{F}^{\mathrm{op}} \to \mathrm{CAT}$.

7.1∗4 Definition (Johnson and Yau (2020)). For a category $\mathscr{C}$ and two functors $F, G : \mathscr{C} \to$ CAT, a *lax transformation* $\alpha : F \to G$ consists of a family of functors $\alpha_X : FX \to GX$ for all $X \in \mathscr{C}$ and a family of natural transformations $\alpha_f : Gf \circ \alpha_X \to \alpha_Y \circ Ff$ for all $f : X \to Y$:

$$\begin{array}{ccc} FX & \xrightarrow{Ff} & FY \\ {\scriptstyle \alpha_X}\downarrow & \swarrow^{\alpha_f} & \downarrow{\scriptstyle \alpha_Y} \\ GX & \xrightarrow[Gf]{} & GY \end{array}$$

Additionally, α must satisfy that $\alpha_{id_X} = id : \alpha_X \to \alpha_X$ for all $X \in \mathscr{C}$, and for all $f : X \to Y$ and $g : Y \to Z$ in $\mathscr{C}$, $\alpha_{g \cdot f}$ is exactly the pasting of α_f and α_g:

$$\begin{array}{ccccc} FX & \xrightarrow{Ff} & FY & \xrightarrow{Gg} & FZ \\ {\scriptstyle \alpha_X}\downarrow & \swarrow^{\alpha_f} & \downarrow{\scriptstyle \alpha_Y} & \swarrow^{\alpha_g} & \downarrow{\scriptstyle \alpha_Z} \\ GX & \xrightarrow[Gf]{} & GY & \xrightarrow[Gg]{} & GZ \end{array} \quad = \quad \begin{array}{ccccc} FX & \xrightarrow{Ff} & FY & \xrightarrow{Gg} & FZ \\ {\scriptstyle \alpha_X}\downarrow & \swarrow^{\alpha_{g\cdot f}} & & & \downarrow{\scriptstyle \alpha_Z} \\ GX & \xrightarrow[Gf]{} & GY & \xrightarrow[Gg]{} & GZ \end{array}$$

An *oplax* transformation from F to G is similar to a lax transformation except that the direction of the 2-cells α_f becomes $\alpha_Y \circ Ff \to Gf \circ \alpha_X$. An (op)lax transformation α is called *strong* if α_f is a natural isomorphism for every $f : X \to Y$ in $\mathscr{C}$, and it is called *strict* if α_f is exactly the identity for every f.

7.1∗5 Definition. Given $\ddot{\Psi} \in \mathcal{F}$, a (strong/strict) *modular model* M of $\ddot{\Psi}$ is a (strong/strict) oplax transformation from $(-)$-ALG to $(- + \ddot{\Psi})$-ALG : $\mathcal{F}^{\mathrm{op}} \to$ CAT.

7.1∗6. Unpacking the definition, a modular model M of $\ddot{\Psi} \in \mathcal{F}$ consists of a family of functors $M_{\ddot{\Sigma}} : \ddot{\Sigma}$-ALG $\to (\ddot{\Sigma} + \ddot{\Psi})$-ALG for all $\ddot{\Sigma} \in \mathcal{F}$ and a family of natural transformations $M_T : M_{\ddot{\Sigma}} \circ T \to (T + \ddot{\Psi}) \circ M_{\ddot{\Sigma}'}$ for all $T : \ddot{\Sigma} \to \ddot{\Sigma}'$ in $\mathcal{F}$:

$$\begin{array}{ccc} \ddot{\Sigma}\text{-ALG} & \xleftarrow{T} & \ddot{\Sigma}'\text{-ALG} \\ {\scriptstyle M_{\ddot{\Sigma}}}\downarrow & \swarrow^{M_T} & \downarrow{\scriptstyle M_{\ddot{\Sigma}'}} \\ (\ddot{\Sigma} + \ddot{\Psi})\text{-ALG} & \xleftarrow[T+\ddot{\Psi}]{} & (\ddot{\Sigma}' + \ddot{\Psi})\text{-ALG} \end{array}$$

such that M_{id} is the identity transformation, and for a pair of morphisms $T : \ddot{\Sigma} \to \ddot{\Sigma}'$ and $T' : \ddot{\Sigma}' \to \ddot{\Sigma}''$, $M_{T' \cdot T}$ is exactly the pasting of M_T and $M_{T'}$:

$$\begin{array}{ccccc} \ddot{\Sigma}\text{-ALG} & \xleftarrow{T} & \ddot{\Sigma}'\text{-ALG} & \xleftarrow{T'} & \ddot{\Sigma}''\text{-ALG} \\ {\scriptstyle M_{\ddot{\Sigma}}}\downarrow & \swarrow^{M_T} & \downarrow{\scriptstyle M_{\ddot{\Sigma}'}} & \swarrow^{M_{T'}} & \downarrow{\scriptstyle M_{\ddot{\Sigma}''}} \\ (\ddot{\Sigma} + \ddot{\Psi})\text{-ALG} & \xleftarrow[T+\ddot{\Psi}]{} & (\ddot{\Sigma}' + \ddot{\Psi})\text{-ALG} & \xleftarrow[T'+\ddot{\Psi}]{} & (\ddot{\Sigma}'' + \ddot{\Psi})\text{-ALG} \end{array}$$

Specially, the data of a *strict* modular model M of $\ddot{\Psi} \in \mathcal{F}$ is only a family of functors $M_{\ddot{\Sigma}}$ such that the following diagram in CAT commutes:

$$\begin{array}{ccc} \ddot{\Sigma}\text{-ALG} & \xleftarrow{T} & \ddot{\Sigma}'\text{-ALG} \\ {\scriptstyle M_{\ddot{\Sigma}}}\downarrow & & \downarrow{\scriptstyle M_{\ddot{\Sigma}'}} \\ (\ddot{\Sigma} + \ddot{\Psi})\text{-ALG} & \xleftarrow[T+\ddot{\Psi}]{} & (\ddot{\Sigma}' + \ddot{\Psi})\text{-ALG} \end{array} \tag{7.1}$$

Therefore a strict modular model for $\ddot{\Psi}$ is exactly an ordinary natural transformation between the functors $(-)$-Alg and $(- + \ddot{\Psi})$-Alg : $\mathcal{F}^{\mathrm{op}} \to \mathrm{CAT}$.

7.1∗7 Example. For a trivial example, let $\mathcal{F}$ be the operation family containing only the initial object $\langle \mathrm{Mon}, \mathrm{Id} : \mathrm{Mon} \to \mathrm{Mon} \rangle$ of $\mathrm{Mon}/\mathrm{Eqs}(\mathscr{E})$. In this case, $\langle \mathrm{Mon}, \mathrm{Id} \rangle + \langle \mathrm{Mon}, \mathrm{Id} \rangle$ is still $\langle \mathrm{Mon}, \mathrm{Id} \rangle$, so a modular model of $\langle \mathrm{Mon}, \mathrm{Id} \rangle$ is exactly an endofunctor $\mathrm{Mon}(\mathscr{E}) \to \mathrm{Mon}(\mathscr{E})$ over the category of monoids in $\mathscr{E}$.

7.1∗8 Example. Let $\mathscr{E}$ be $\langle \mathrm{Endo}_\kappa(\mathscr{C}), \circ, \mathrm{Id} \rangle$ for lκp $\mathscr{C}$. A strict modular model M for the theory Et_E of *exception throwing* (Example 5∗10) in the family $\mathrm{Alg}(\mathscr{E})$ of algebraic operations is given by a family of functors

$$M_{\dot{\Sigma}} : \ddot{\Sigma}\text{-Alg} \to (\ddot{\Sigma} + \mathrm{Et}_E)\text{-Alg}$$

natural in $\ddot{\Sigma} \in \mathrm{Alg}(\mathscr{E})$. Recall that each $\langle \dot{\Sigma}, T_\Sigma \rangle \in \mathrm{Alg}(\mathscr{E})$ is of the form

$$(S \,\square\, -)\text{-Mon} \upharpoonleft (\mathrm{K}_G \vdash L = R)$$

for some S and $G \in \mathrm{Endo}_\kappa(\mathscr{C})$, so objects of $\ddot{\Sigma}$-Alg are tuples

$$\langle A \in \mathrm{Endo}_\kappa(\mathscr{C}),\ \ \alpha : S \circ A \to A,\ \ \eta^A : \mathrm{Id} \to A,\ \ \mu^A : A \circ A \to A \rangle$$

that are mapped by L and R to the same algebra. We define $M_{\ddot{\Sigma}}$ to send each of them to a $(\ddot{\Sigma} + \mathrm{Et}_E)$-algebra carried by $C_A := A \circ (\bar{E} + \mathrm{Id})$, i.e. the *exception monad transformer* applied to A, where $\bar{E}$ is the E-fold coproduct of the terminal object 1 in $\mathrm{Endo}_\kappa(\mathscr{C})$. Using the internal language of $\mathrm{Endo}_\kappa(\mathscr{C})$, the operations $[\alpha^\sharp, \beta] :\ (S \circ C_A) + \bar{E} \to C_A$ are

$$\begin{aligned} \alpha^\sharp &= [\![s : S, a : A, e : \bar{E} + \mathrm{Id} \vdash (\alpha(s, a),\ e) : C_A]\!] \\ \beta &= [\![e : \bar{E} \vdash (\eta^A,\ \iota_1\, e) : C_A]\!] \end{aligned}$$

and C_A has the following monad structure:

$$\begin{aligned} \eta^C &= [\![\vdash (\eta^A, \iota_2(*)) : C_A]\!] \\ \mu^C &= [\![a : A,\ e : \bar{E} + \mathrm{Id},\ a' : A,\ e' : \bar{E} + \mathrm{Id} \vdash \\ &\qquad\qquad \mathit{let}\ (a'', e'') = d(e, a')\ \mathit{in}\ (\mu^A(a, a''), \mu^{\bar{E}+\mathrm{Id}}(e'', e'))]\!] \end{aligned}$$

where $d : (\bar{E} + \mathrm{Id}) \circ A \to A \circ (\bar{E} + \mathrm{Id})$ is the following distributive law:

$$e : \bar{E} + \mathrm{Id},\ a : A \ \vdash \mathit{case}\ e\ \mathit{of}\ \{\ \iota_1\, e' \mapsto (\eta^A, \iota_1 e');\ \ \iota_2 * \mapsto (a, \iota_2 *) : C_A \},$$

and $\mu^{\bar{E}+\mathrm{Id}}$ is the multiplication of the exception monad $\bar{E} + \mathrm{Id}$:

$$x : \bar{E} + \mathrm{Id},\ \ y : \bar{E} + \mathrm{Id} \ \vdash \mathit{case}\ x\ \mathit{of}\ \{\ \iota_1\, e \mapsto \iota_1\, e;\ \ \iota_2 * \mapsto y \} : \bar{E} + \mathrm{Id}.$$

On morphisms, $M_{\ddot{\Sigma}}$ sends a $\dot{\Sigma}$-homomorphism $h : A \to B$ to $h \circ (\bar{E} + \mathrm{Id})$.

We need to check that the family of functors $M_{\ddot{\Sigma}}$ satisfies the naturality square (7.1): for all $T : \ddot{\Sigma} \to \ddot{\Sigma}'$, every object $\langle A, \alpha, \eta^A, \mu^A \rangle$ of $\ddot{\Sigma}'$-Alg is mapped by functors $M_{\ddot{\Sigma}} \circ T$ and $(T + \mathrm{Et}_E) \circ M_{\ddot{\Sigma}'}$ to two objects in $(\ddot{\Sigma} + \mathrm{Et}_E)$-Alg with the same carrier C_A, the same Et_E-algebra structure, and the same monad structure by construction. We need to show that the respective $\ddot{\Sigma}$-algebras on C_A are also the same: the $\ddot{\Sigma}$-algebra on C_A from $M_{\ddot{\Sigma}} \circ T$ is

$$T\alpha \circ (\bar{E} + \mathrm{Id}) : S \circ A \circ (\bar{E} + \mathrm{Id}) \to A \circ (\bar{E} + \mathrm{Id}),$$

and the one from $(T + \mathrm{ET}_E) \circ M_{\ddot{\Sigma}'}$ is $T(\alpha \circ (\bar{E} + \mathrm{Id}))$. By Lemma 5∗13, it is sufficient to show that the following diagram commutes

$$S \xrightarrow{S \circ \eta^C} S \circ C_A \underset{T\alpha \circ (\bar{E}+\mathrm{Id})}{\overset{T(\alpha \circ (\bar{E}+\mathrm{Id}))}{\rightrightarrows}} C_A \tag{7.2}$$

To see this, we first observe that we have the following $\ddot{\Sigma}'$-homomorphism square:

$$\begin{array}{ccc} S' \circ A & \xrightarrow{\alpha} & A \\ {\scriptstyle S' \circ A \circ \iota_2}\downarrow & & \downarrow{\scriptstyle A \circ \iota_2} \\ S' \circ A \circ (\bar{E} + \mathrm{Id}) & \xrightarrow[\alpha \circ (\bar{E}+\mathrm{Id})]{} & A \circ (\bar{E} + \mathrm{Id}) \end{array}$$

and this square is mapped by the translation T to a commuting square

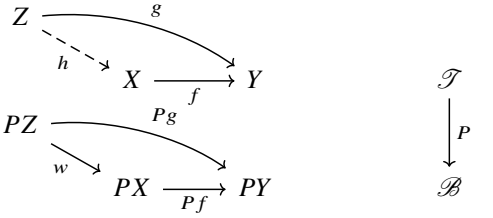
$$\begin{array}{ccc} S \circ A & \xrightarrow{T\alpha} & A \\ {\scriptstyle S \circ A \circ \iota_2}\downarrow & & \downarrow{\scriptstyle A \circ \iota_2} \\ S \circ A \circ (\bar{E} + \mathrm{Id}) & \xrightarrow[T(\alpha \circ (\bar{E}+\mathrm{Id}))]{} & A \circ (\bar{E} + \mathrm{Id}) \end{array}$$

This implies (7.2) because

$$\begin{aligned} & T(\alpha \circ (\bar{E} + \mathrm{Id})) \cdot (S \circ \eta^C) \\ = \quad & \{\text{definition of } \eta^C\} \\ & T(\alpha \circ (\bar{E} + \mathrm{Id})) \cdot (S \circ A \circ \iota_2) \cdot (S \circ \eta^A) \\ = \quad & \{\text{by the last commutativity square above}\} \\ & (A \circ \iota_2) \cdot T\alpha \cdot (S \circ \eta^A) \\ = \quad & \{\text{by naturality}\} \\ & (T\alpha \circ (\bar{E} + \mathrm{Id})) \cdot (A \circ \iota_2) \cdot (S \circ \eta^A) \\ = \; & (T\alpha \circ (\bar{E} + \mathrm{Id})) \cdot (S \circ \eta^C) \end{aligned}$$

7.1∗9. CAT-valued functors also known as (strict) *indexed categories*, which are equivalent to *split fibrations* via the *Grothendieck construction*, so we can alternatively formulate modular models based on fibrations. The fibrational formulation is usually easier to work with, especially when we talk about morphisms between modular models later, which would be a 3-categorical concept in the indexed-category formulation. Also, the fibrational formulation slightly simplifies some 'dependently typed' constructions such as mapping each $\ddot{\Sigma}$ in $\mathcal{F}$ to the initial algebra in $\ddot{\Sigma}$-Alg. We will only need the very basics about fibrations (see e.g. Jacobs (1999); Streicher (2023); Borceux (1994)), which we review below.

7.1∗10 (Fibrations). Let $P : \mathscr{T} \to \mathscr{B}$ be a functor. A morphism $f : X \to Y$ in $\mathscr{T}$ is called *cartesian* if for every $g : Z \to Y$ and $w : PZ \to PX$ such that $Pg = Pf \cdot w$, there is a unique $h : Z \to X$ in $\mathscr{T}$ satisfying $Ph = w$ and $f \cdot h = g$:



The functor P called a *(Grothendieck) fibration* if for every morphism $u : I \to J$ in $\mathscr{B}$ and object $Y \in \mathscr{T}$ such that $PY = J$, there exists a cartesian morphism $f : X \to Y$ in $\mathscr{T}$ with $Pf = u$. It is customary to call the category $\mathscr{T}$ the *total category* and $\mathscr{B}$ the *base category*. If an object X or a morphism f in $\mathscr{T}$ is sent by the functor P to an object I or a morphism u in $\mathscr{B}$, we colloquially say that X or f is *over* I or u. Given an object $I \in \mathscr{B}$, the *fiber category* $\mathscr{T}_I$ over I is the subcategory of $\mathscr{T}$ consisting of objects over I and morphisms over id_I.

A *cleavage* for a fibration $P : \mathscr{T} \to \mathscr{B}$ is an assignment κ of cartesian morphisms $\kappa(Y, u)$ over u to all pairs of $Y \in \mathscr{T}$ and $u : I \to PY$ for some $I \in \mathscr{B}$. A cleavage κ is said to be a *split cleavage* if it is functorial: $\kappa(Y, id_{PY}) = id_Y$ and $\kappa(Y, u \cdot v) = \kappa(Y, u) \cdot \kappa(\mathrm{Dom}\ \kappa(Y, u), v)$. A *split fibration* is a fibration $P : \mathscr{T} \to \mathscr{B}$ equipped with a split cleavage.

Let $P : \mathscr{T} \to \mathscr{B}$ be a fibration with a cleavage κ. For every morphism $u : I \to J$ in the base category $\mathscr{B}$, the *reindexing functor* $u^* : \mathscr{T}_J \to \mathscr{T}_I$ sends every object $Y \in \mathscr{T}_J$ to the domain object $X \in \mathscr{T}_I$ of the morphism $\kappa(Y, u) : X \to Y$ and sends every morphism $f : Y \to Y'$ in $\mathscr{T}_I$ to the morphism $h : u^*Y \to u^*Y'$ as follows:

$$\begin{array}{ccc} u^*Y & \xrightarrow{\kappa(Y,u)} & Y \\ & \searrow^{h} & \downarrow f \\ & u^*Y' & \xrightarrow[\kappa(Y',u)]{} Y' \end{array} \qquad \begin{array}{ccc} I & \xrightarrow{u} & J \\ & \searrow^{id} & \downarrow id \\ & I & \xrightarrow[u]{} J \end{array}$$

where h is obtained from the cartesianess of $\kappa(Y', u) : u^*Y' \to Y'$ and the fact that $P(f \cdot \kappa(Y, u)) = id \cdot u = u = P(\kappa(Y', u))$. The functoriality of $u^* : \mathscr{T}_J \to \mathscr{T}_I$ is a consequence of the uniqueness part of cartesianess.

A morphism $P \to P'$ of fibrations is a pair of functors $\langle F, G \rangle$ such that the diagram below commutes and F maps cartesian morphisms to cartesian morphisms. A morphism $\langle P, \kappa \rangle \to \langle P', \kappa' \rangle$ of split fibrations is a morphism $\langle F, G \rangle : P \to P'$ that strictly preserves the split cleavage: $F\kappa(Y, u) = \kappa(FY, Gu)$.

$$\begin{array}{ccc} \mathscr{T} & \xrightarrow{F} & \mathscr{T}' \\ {\scriptstyle P}\downarrow & & \downarrow{\scriptstyle P'} \\ \mathscr{B} & \xrightarrow[G]{} & \mathscr{B}' \end{array} \tag{7.3}$$

With component-wise identity functors and functor composition, fibrations (resp. split fibrations) and morphisms form a category Fib (resp. Fib^s). Also, given a category $\mathscr{B}$, there is a subcategory $\mathrm{Fib}_{\mathscr{B}} \subseteq \mathrm{Fib}$ (resp. $\mathrm{Fib}^s_{\mathscr{B}} \subseteq \mathrm{Fib}^s$) containing all (resp. split) fibrations over $\mathscr{B}$ and morphisms $\langle F, \mathrm{Id} : \mathscr{B} \to \mathscr{B} \rangle : P \to P'$.

Let $P : \mathscr{T} \to \mathscr{B}$ be a fibration and $F : \mathscr{C} \to \mathscr{B}$ be a functor. A basic result (Jacobs, 1999, Lemma 1.5.1) in fibred category theory is that the pullback $F^*P : F^*\mathscr{T} \to \mathscr{C}$ of P along F in the (1-)category CAT is still a fibration:

$$\begin{array}{ccc} F^*\mathscr{T} & \longrightarrow & \mathscr{T} \\ {\scriptstyle F^*P}\downarrow & \lrcorner & \downarrow{\scriptstyle P} \\ \mathscr{C} & \xrightarrow[F]{} & \mathscr{B} \end{array}$$

The fibration F^*P is called the *change of base* of P along F. Moreover, if P has a split cleavage then so does F^*P. Explicitly, the objects of $F^*\mathscr{T}$ are pairs $\langle I \in \mathscr{C}, X \in \mathscr{T} \rangle$ such that $FI = PX$, the morphisms $\langle I, X \rangle \to \langle J, Y \rangle$ in $F^*\mathscr{T}$ are pairs $\langle f : I \to J, g : X \to Y \rangle$ such that $Ff = Pg$. A morphism $\langle f, g \rangle$ in $F^*\mathscr{T}$ is cartesian if and only if g is cartesian.

7.1∗11 Definition. Given a functor $F : \mathscr{B}^{\mathrm{op}} \to \mathrm{CAT}$, the *Grothendieck construction* is a split fibration $P : \int F \to \mathscr{B}$ where the category $\int F$ has tuples $\langle I \in \mathscr{B}, a \in FI \rangle$ as objects, and its morphisms $\langle I, a \rangle \to \langle J, a' \rangle$ are pairs $\langle f, g \rangle$ for $f : I \to J$ in $\mathscr{B}$ and $g : a \to Ffa'$ in the category FI. Identity arrows are $\langle id, id \rangle$ and the composition $\langle f', g' \rangle \cdot \langle f, g \rangle$ is $\langle f' \cdot f, g' \cdot Ff'g \rangle$. The fibration $P : \int F \to \mathscr{B}$ is the first projection $\langle I, a \rangle \mapsto I$. The split cleavage $\kappa(\langle I, a \rangle, u)$ for some $u : J \to I$ is $\langle u, id \rangle : \langle J, (Fu)a \rangle \to \langle I, a \rangle$.

7.1∗12. Applying the Grothendieck construction to $(-)\text{-}\textsc{Alg} : \mathcal{F}^{\mathrm{op}} \to \mathrm{CAT}$, we obtain a (split) fibration $P : \mathcal{F}\text{-}\textsc{Alg} \to \mathcal{F}$, which we explicitly describe below. The intuition is that $\mathcal{F}\text{-}\textsc{Alg}$ is the category of *all models of all equational systems* in $\mathcal{F}$. The objects of the category $\mathcal{F}\text{-}\textsc{Alg}$ are tuples

$$\langle \dot{\Sigma} \in \textsc{Eqs}(\mathscr{E}), \quad T_\Sigma : \textsc{Mon} \to \dot{\Sigma}, \quad A \in \mathscr{E}, \quad \alpha : \Sigma A \to A \rangle$$

such that $\langle \dot{\Sigma}, T_\Sigma \rangle \in \mathcal{F}$ and $\langle A, \alpha \rangle \in \dot{\Sigma}\text{-}\textsc{Alg}$. Morphisms between two objects $\langle \dot{\Sigma}, T_\Sigma, A, \alpha \rangle$ and $\langle \dot{\Psi}, T_\Psi, B, \beta \rangle$ are pairs $\langle T, h \rangle$ where $T : \dot{\Sigma} \to \dot{\Psi}$ is a functorial translation in $\mathcal{F}$, and the other component $h : A \to B \in \mathscr{E}$ is a $\dot{\Sigma}$-algebra homomorphism from $\langle A, \alpha \rangle$ to $T\langle B, \beta \rangle$:

$$\begin{array}{ccc} \Sigma B & \xleftarrow{\Sigma h} & \Sigma A \\ {\scriptstyle T\beta}\downarrow & & \downarrow{\scriptstyle \alpha} \\ B & \xleftarrow{h} & A \end{array} \quad \text{in } \dot{\Sigma}\text{-}\textsc{Alg} \qquad \xleftarrow{T} \qquad \begin{array}{c} \Psi B \\ {\scriptstyle \beta}\downarrow \\ B \end{array} \quad \text{in } \dot{\Psi}\text{-}\textsc{Alg}$$

The identities in $\mathcal{F}\text{-}\textsc{Alg}$ are pairs $\langle \mathrm{Id} : \dot{\Sigma} \to \dot{\Sigma}, id : A \to A \rangle$, and the composition of two morphisms $\langle T, h \rangle$ and $\langle T', h' \rangle$ is $\langle T \circ T', h \cdot h' \rangle$.

The fibration $P : \mathcal{F}\text{-}\textsc{Alg} \to \mathcal{F}$ is the projection: $P\langle \dot{\Sigma}, T_\Sigma, A, \alpha \rangle = \langle \dot{\Sigma}, T_\Sigma \rangle$ and $P\langle T, h \rangle = T$. It has a split cleavage assigning to every pair of a morphism $T : \langle \dot{\Sigma}, T_\Sigma \rangle \to \langle \dot{\Psi}, T_\Psi \rangle \in \mathcal{F}$ and an object $\langle \dot{\Psi}, T_\Psi, B, \beta \rangle \in \mathcal{F}\text{-}\textsc{Alg}$ a cartesian morphism $\langle T, id \rangle : \langle \dot{\Sigma}, T_\Sigma, B, T\beta \rangle \to \langle \dot{\Psi}, T_\Psi, B, \beta \rangle$ in $\mathcal{F}\text{-}\textsc{Alg}$.

7.1∗13 Example. Given an object $\ddot{\Psi} \in \mathcal{F}$, the Grothendieck construction of the functor $(- + \ddot{\Psi})\text{-}\textsc{Alg} : \mathcal{F}^{\mathrm{op}} \to \mathrm{CAT}$ is a split fibration $Q : (\mathcal{F} + \ddot{\Psi})\text{-}\textsc{Alg} \to \mathcal{F}$. Explicitly, the objects of $(\mathcal{F} + \ddot{\Psi})\text{-}\textsc{Alg}$ are tuples:

$$\langle \dot{\Sigma} \in \textsc{Eqs}(\mathscr{E}), \quad T_\Sigma : \textsc{Mon} \to \dot{\Sigma}, \quad A \in \mathscr{E}, \quad \alpha : \Sigma A \to A, \quad \beta : \Psi A \to A \rangle$$

such that $\langle \dot{\Sigma}, T_\Sigma \rangle \in \mathcal{F}$, $\langle A, \alpha \rangle \in \dot{\Sigma}\text{-}\textsc{Alg}$, $\langle A, \beta \rangle \in \dot{\Psi}\text{-}\textsc{Alg}$, and $T_\Psi \alpha = T_\Sigma \beta$. Morphisms in the category $(\mathcal{F} + \dot{\Psi})\text{-}\textsc{Alg}$ are similar to those $\langle T, h \rangle$ in $\mathcal{F}\text{-}\textsc{Alg}$, but require h also to be a $\dot{\Psi}$-homomorphism. Therefore, objects of $(\mathcal{F} + \ddot{\Psi})\text{-}\textsc{Alg}$ are models of some equational systems in $\mathcal{F}$ that are additionally equipped with a $\ddot{\Psi}$-algebra. The functor Q is the projection $\langle \dot{\Sigma}, T, A, \alpha, \beta \rangle \mapsto \langle \dot{\Sigma}, T \rangle$.

The split fibration $Q : (\mathcal{F} + \ddot{\Psi})\text{-}\textsc{Alg} \to \mathcal{F}$ can be alternatively given as the change of base of the fibration $P : \mathcal{F}\text{-}\textsc{Mon} \to \mathcal{F}$ along the functor $(- + \ddot{\Psi}) : \mathcal{F} \to \mathcal{F}$, which is the following

pullback in the category CAT of large categories:

$$\begin{array}{ccc} (\mathcal{F}+\ddot{\Psi})\text{-Alg} & \xrightarrow{K} & \mathcal{F}\text{-Alg} \\ {\scriptstyle Q}\downarrow & \lrcorner & \downarrow{\scriptstyle P} \\ \mathcal{F} & \xrightarrow[-+\ddot{\Psi}]{} & \mathcal{F} \end{array} \tag{7.4}$$

The functor K maps objects $\langle \dot{\Sigma}, T_\Sigma, A, \alpha, \beta\rangle$ to $\langle\langle \dot{\Sigma}, T_\Sigma\rangle + \langle \dot{\Psi}, T_\Psi\rangle, A, [\alpha, \beta]\rangle$. The pair $\langle K, (- + \ddot{\Psi})\rangle$ is a morphism of split fibrations $Q \to P$.

7.1∗14 Definition. Given two fibrations $P : \mathscr{E} \to \mathscr{B}$ and $P' : \mathscr{E}' \to \mathscr{B}'$, a *lifting* of a functor $F : \mathscr{B} \to \mathscr{B}'$ along P and P' is a functor $G : \mathscr{E} \to \mathscr{E}'$ making the following diagram commute:

$$\begin{array}{ccc} \mathscr{E} & \xrightarrow{G} & \mathscr{E}' \\ {\scriptstyle P}\downarrow & & \downarrow{\scriptstyle P'} \\ \mathscr{B} & \xrightarrow[F]{} & \mathscr{B}' \end{array}$$

The lifting is called *fibred* if $\langle F, G\rangle$ is a morphism of fibrations (i.e. G preserves all cartesian morphisms). When P and P' have split cleavages, the lifting G is called *split* if $\langle F, G\rangle$ is a morphism of split fibrations.

A morphism between two liftings $G \to H$ of F is a natural transformation $\sigma : M \to N$ that is vertical, i.e. $P'\sigma = id_{T \circ P}$:

$$\begin{array}{ccc} & \overset{G}{\curvearrowright} & \\ \mathscr{E} & \Downarrow \sigma & \mathscr{E}' \\ {\scriptstyle P}\downarrow & \underset{H}{\curvearrowright} & \downarrow{\scriptstyle P'} \\ \mathscr{B} & \xrightarrow[F]{} & \mathscr{B}' \end{array}$$

Liftings of F along P and P' and morphisms between them form a category $\textsc{Lift}_{P,P'}(F)$, whose identity morphisms are the identity natural transformations, and composition is vertical composition of natural transformations.

7.1∗15 (Modular models as liftings). Using the language of fibrations, we have now a very concise alternative formulation of modular models: a (strong/strict) modular model M of $\ddot{\Psi} \in \mathcal{F}$ is just a (fibred/split) lifting of the endofunctor $(- + \ddot{\Psi}) : \mathcal{F} \to \mathcal{F}$ along the fibration $P : \mathcal{F}\text{-Alg} \to \mathcal{F}$ from 7.1∗12:

$$\begin{array}{ccc} \mathcal{F}\text{-Alg} & \xrightarrow{M} & \mathcal{F}\text{-Alg} \\ {\scriptstyle P}\downarrow & & \downarrow{\scriptstyle P} \\ \mathcal{F} & \xrightarrow[-+\ddot{\Psi}]{} & \mathcal{F} \end{array}$$

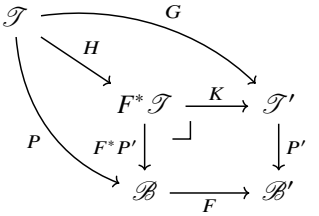
The commutativity of the square ensures that the functor M maps every object $\langle \ddot{\Sigma}, A, \alpha\rangle$ to an object $\langle \ddot{\Sigma} + \ddot{\Psi}, B, \beta\rangle$. Below, we show that this formulation of modular models is equivalent to Definition 7.1∗5 based on indexed categories.

7.1∗16 Lemma. Given two fibrations $P : \mathscr{T} \to \mathscr{B}$, $P' : \mathscr{T}' \to \mathscr{B}'$, and a functor $F : \mathscr{B} \to \mathscr{B}'$, let $F^*P' : F^*\mathscr{T}' \to \mathscr{B}$ be the change of base of P' along F.

1. Liftings of F along P and P' are in bijection with $\mathrm{CAT}/\mathscr{B}(P, F^*P')$.

2. Fibred liftings of F along P and P' are in bijection with $\mathrm{FIB}_{\mathscr{B}}(P, F^*P')$.
3. If P and P' are equipped with split cleavages, split liftings of F along P and P' are in bijection with $\mathrm{FIB}^{s}_{\mathscr{B}}(P, F^*P')$.

Proof. By definition, F^*P' is the pullback in the following diagram:



By definition, liftings of F are functors $G : \mathscr{T} \to \mathscr{T}'$ such that $F \circ P = P' \circ G$, so by the universal property of the pullback, liftings G are in bijection with functors $H : \mathscr{T} \to F^*\mathscr{T}'$ such that $(F^*P') \circ H = P$, and the backward direction is given by $H \mapsto K \circ H$. This is the first item in the statement.

This bijection cuts down to fibred (resp. split) liftings. The functor K always preserves cartesian morphisms and the split cleavage. Hence from one direction, if H preserves cartesian morphisms (resp. split cleavage), then so does $K \circ H$. From the other direction, if G preserves cartesian morphisms (resp. split cleavage), then the functor H sends a cartesian morphism f in $\mathscr{T}$ to the pair $\langle Pf, Gf\rangle$, which is cartesian for the fibration F^*P'. □

7.1∗17 Lemma. For a category $\mathscr{C}$ and two functors $F, G : \mathscr{C}^{\mathrm{op}} \to \mathrm{CAT}$, denote the Grothendieck construction of F and G by $p : \int F \to \mathscr{C}$ and $q : \int G \to \mathscr{C}$. Oplax transformations $F \to G$ are in bijection with $\mathrm{CAT}/\mathscr{C}(p, q)$.

Proof. For one direction, given an oplax transformation $\alpha : F \to G$, we define a functor $K : \int F \to \int G$ as follows. On objects, K sends every object $\langle I \in \mathscr{C}, a \in FI\rangle$ to $\langle I, \alpha_I a \in GI\rangle$. On morphisms, K sends every morphism

$$\langle f : I \to J, g : a \to (Ff)b\rangle : \langle I, a\rangle \to \langle J, b\rangle$$

in $\int F$ to $\langle f, g'\rangle$ where g' is the following morphism in the fiber category G_I:

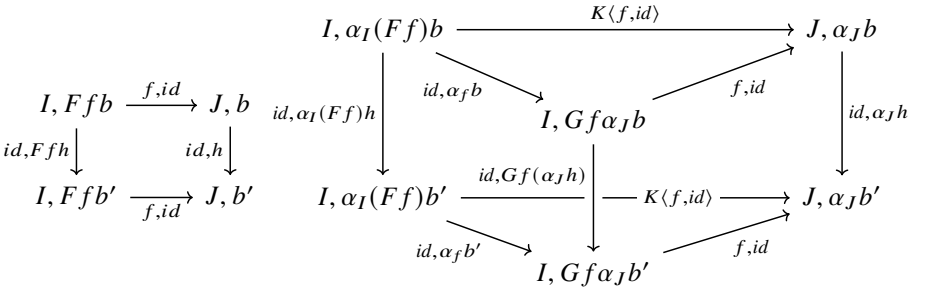
$$\alpha_I a \xrightarrow{\alpha_I g} \alpha_I(Ffb) \xrightarrow{\alpha_{f,b}} (Gf)\alpha_J b.$$

The functor preserves identities and composition following the unity and composition axioms of oplax transformations.

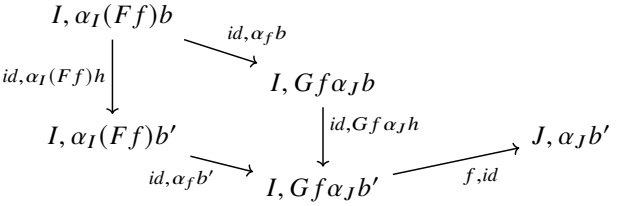
For the other direction, given a functor $K : \int F \to \int G$ with $q \circ F = p$, we define an oplax transformation $\alpha : F \to G$ as follows. For every object $I \in \mathscr{C}$, we define the functor $\alpha_I : FI \to GI$ to be K restricted to the fiber category of $\int F$ over I. For every $f : I \to J$ in $\mathscr{C}$, we need to define a natural transformation $\alpha_f : \alpha_I \circ Ff \to Gf \circ \alpha_J : FJ \to GI$. For each object $b \in FJ$, there is a morphism $\langle f, id\rangle : \langle I, (Ff)b\rangle \to \langle J, b\rangle$ in the total category $\int F$, and this morphism is mapped by K to some $\langle f, g\rangle : \langle I, \alpha_I(Ffb)\rangle \to \langle J, \alpha_J b\rangle$, we define the natural transformation α_f to be

$$\alpha_{f,b} := g : \alpha_I(Ffb) \to Gf(\alpha_J b)$$

in the fiber category FI. For every $h : b \to b'$ in FJ, the naturality of α_f follows from the fact that the diagram below left commutes in $\int F$:



and K maps this diagram to $\int G$, which is the back square in the diagram above right. We observe that in the diagram above right the two triangles commute by the definition of $\alpha_f b$ and $\alpha_f b'$; the right square commutes by the definition of arrow composition in $\int G$. Hence the following diagram commutes:



The square on the left is exactly the naturality square for α_f that we need, which also commutes because the morphism $\langle f, id\rangle$ on the right is cartesian. It can be checked that this natural transformations satisfies the axioms of oplax transformations and that the two directions define a bijection. □

7.1∗18 Theorem. *For every $\ddot{\Psi} \in \mathcal{F}$, the following are in bijection with each other*:

1. *modular models* (*resp. strong or strict modular models*) *as in Definition 7.1∗5,*
2. *functors in* $\mathrm{CAT}/\mathcal{F}(P, Q)$ (*resp. morphisms of fibrations or split fibrations from P to Q*) *where the split fibrations P and Q are as in (7.4), and*
3. *liftings* (*resp. fibred or split liftings*) *of the endofunctor $(- + \ddot{\Psi}) : \mathcal{F} \to \mathcal{F}$ along the fibration $P : \mathcal{F}\text{-}\textsc{Alg} \to \mathcal{F}$.*

Proof. The bijection between 2 and 3 is Lemma 7.1∗16. The bijection between modular models and morphisms in $\mathrm{CAT}/\mathcal{F}$ is Lemma 7.1∗17. The bijection between strong (resp. strict) modular models – which are exactly strong transformations (resp. natural transformations) between CAT-valued functors – and morphisms of fibrations (resp. split fibrations) is standard (Jacobs, 1999, §1.10). □

7.1∗19 Notation. In the future, we will leave implicit the conversion between modular models as oplax transformations and as liftings of functors, so we may say 'a modular model $M : \mathcal{F}\text{-}\textsc{Alg} \to \mathcal{F}\text{-}\textsc{Alg}$ of $\ddot{\Psi}$'.

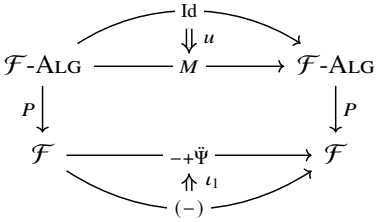
7.1∗20. One advantage of the fibrational formulation is that we avoid the need of a category of categories CAT bigger than the base category. Also, it reduces the 2-categorical notion of

oplax transformations to the 1-categorical notion of functors. Consequently, 3-categorical concepts can be avoided when talking about transformations of modular models, such as homomorphisms between modular models and the concept of *updaters* below.

7.1∗21 Definition. Let M be a modular model of $\ddot{\Psi} \in \mathcal{F}$ given in the lifting form. An *updater* u for M is a natural transformation $u : \mathrm{Id} \to M$ such that

$$(P \circ u) = (\iota_1 \circ P) : P \to (P \circ M) = (- + \ddot{\Psi}) \circ P$$

where $\iota_1 : (-) \to (-) + \ddot{\Psi}$ is the coprojection in $\mathcal{F}$:



7.1∗22. For an object $\langle \ddot{\Sigma}, A, \alpha \rangle \in \mathcal{F}\text{-ALG}$, the component of u at this object is a $\ddot{\Sigma}$-homomorphism from $\langle A, \alpha \rangle$ to the algebra $M\langle \ddot{\Sigma}, A, \alpha \rangle$ forgetting the $\ddot{\Psi}$-algebra. If we informally think of $\ddot{\Sigma}$ as a programming language, $\ddot{\Psi}$ as the new feature in a new release of the programming language, and $\langle A, \alpha \rangle$ as the existing compiled programs of $\ddot{\Sigma}$, then the role of u is updating existing compiled programs to the new version, hence its name 'updater'.

7.1∗23. In the setting of Example 7.1∗7, updaters correspond to exactly the *lifting operation* $l : \mathrm{Id} \to T : \textsc{Mon}(\mathscr{E}) \to \textsc{Mon}(\mathscr{E})$ of monad/monoid transformers (Jaskelioff and Moggi, 2010). We have switched to the terminology *lifting* to avoid the confusion with liftings along fibrations (Definition 7.1∗14).

7.1∗24. Assuming objects of $\mathcal{F}$ have initial algebras, the 'dependently typed' mapping sending every $\langle \dot{\Sigma}, T_\Sigma \rangle \in \mathcal{F}$ to its initial algebra $\mu\dot{\Sigma}$ can be conveniently formulated as a functor $(-)^\star : \mathcal{F} \to \mathcal{F}\text{-ALG}$ using the fibrational language:

$$\begin{aligned} \ddot{\Sigma}^\star &= \langle \dot{\Sigma},\ T_\Sigma,\ \mu\dot{\Sigma},\ \alpha^\Sigma : \Sigma(\mu\dot{\Sigma}) \to \mu\dot{\Sigma} \rangle \\ (T : \ddot{\Sigma} \to \ddot{\Psi})^\star &= \langle T,\ ! : \langle \mu\dot{\Sigma},\ \alpha^\Sigma \rangle \to T\langle \mu\dot{\Psi},\ \alpha^\Psi \rangle \rangle \end{aligned} \tag{7.5}$$

where $!$ is the unique $\dot{\Sigma}$-homomorphism out of the initial algebra $\mu\dot{\Sigma}$.

Let M be a modular model of some $\ddot{\Psi} \in \mathcal{F}$ in the lifting form. For every $\ddot{\Sigma} \in \mathcal{F}$, we have a unique $(\ddot{\Sigma} + \ddot{\Psi})$-homomorphism out of the initial algebra $(\ddot{\Sigma} + \ddot{\Psi})^\star$:

$$h^M_{\ddot{\Sigma}} : (\ddot{\Sigma} + \ddot{\Psi})^\star \to M\ddot{\Sigma}^\star, \tag{7.6}$$

which defines a natural transformation

$$h^M : (- + \ddot{\Psi})^\star \to M(-)^\star : \mathcal{F} \to \mathcal{F}\text{-ALG}. \tag{7.7}$$

This is how M modularly handles/interprets $\dot{\Psi}$-operations in programs $(\ddot{\Sigma} + \ddot{\Psi})^\star$ with both $\ddot{\Psi}$-operations and some other $\ddot{\Sigma}$-operations, leaving operations from the other theory $\ddot{\Sigma}$ uninterpreted. Specially, $\ddot{\Sigma}$ can be the initial object $\langle \textsc{Mon}, \mathrm{Id} \rangle$ of $\mathcal{F}$, whose initial algebra is the initial monoid I. In this case, the morphism $h^M_{\textsc{Mon},\mathrm{Id}} : \ddot{\Psi}^\star \to MI$ interprets the abstract syntax $\ddot{\Psi}^\star$ with no 'residual operations'.

7.2 Model Transformers

7.2∗1. In the previous section, we have seen modular models of theories of monoids with operations as liftings of endofunctors $- + \ddot{\Psi}$ along a fibration P:

$$\begin{array}{ccc} \mathcal{F}\text{-Alg} & \xrightarrow{M} & \mathcal{F}\text{-Alg} \\ {\scriptstyle P}\downarrow & & \downarrow{\scriptstyle P} \\ \mathcal{F} & \xrightarrow[-+\ddot{\Psi}]{} & \mathcal{F} \end{array} \tag{7.8}$$

But there is no reason that the idea of modular models is specific to coproducts $- + \ddot{\Psi}$ or the fibration $\mathcal{F}\text{-Alg} \to \mathcal{F}$ for monoids with operations, since we may be interested in ways of combining algebraic theories other than coproducts, and we may be interested in other fibrations of theories and algebras. Thus we shall just study liftings of arbitrary functors along two possibly different fibrations. In this section, we justify the generalisation by showing a few more instances of liftings along fibrations related to modularity.

7.2∗2 (Commutative combination). Consider $\mathscr{E} = \langle \text{End}\text{o}_\kappa(\mathscr{C}), \circ, \text{Id}\rangle$ for some $\mathscr{C}$ that is lκp as a cartesian closed category, as ωCpo and Set. For functors $A_1, A_2 \in \text{End}\text{o}_\kappa(\mathscr{C})$, we had seen the *Day tensor product* $A_1 \otimes A_2$ in Section 2.6:

$$(A_1 \otimes A_2)n = \int^{m,k \in \mathscr{C}_\kappa} A_1 m \times A_2 k \times n^{m \times k},$$

which is intuitively a pair of A_1-operation and A_2-operation that do not depend on each other. This is clearly symmetric, so we have an isomorphism

$$s : A_1 * A_2 \cong A_2 * A_1.$$

Also, there is a canonical morphism $i : A_1 * A_2 \to A_1 \circ A_2$ as shown in 2.6∗3.

We define a functor $\otimes : \text{Alg}(\mathscr{E}) \times \text{Alg}(\mathscr{E}) \to \text{Alg}(\mathscr{E})$ as follows, which is closely related to the *commutative combination*, also known as the *tensor*, of enriched algebraic theories (Hyland et al., 2006). Let $\ddot{\Sigma}_i \in \text{Alg}(\mathscr{E})$ be

$$\langle (A_i \,\square\, -)\text{-Mon} \upharpoonleft (\text{K}_{B_i} \vdash L_i = R_i),\ T_i \rangle, \text{ for } i = 1, 2.$$

We define $\ddot{\Sigma}_1 \otimes \ddot{\Sigma}_2$ to be $\ddot{\Sigma}_1 + \ddot{\Sigma}_2$ extended with the following (constant) equation

$$o : A_1 * A_2 \vdash f\ (i\ o) = f\ (i\ (s\ o)) : \tau.$$

where the term $o' : A_1 \circ A_2 \vdash f : \tau$ is defined by

$$o' : A_1 \circ A_2 \vdash \mathit{let}\ (a_1, a_2) = o'\ \mathit{in}\ \mu\ (op_1\ (a_1, \eta\ *),\ op_2\ (a_2, \eta\ *)) : \tau,$$

and $op_i : A_i \circ \tau \to \tau$, $\eta : I \to \tau$, $\mu : \tau \circ \tau \to \tau$ are respectively the algebraic operations, the unit, and the multiplication of the monoid.

Informally, a model of $\ddot{\Sigma}_1 \otimes \ddot{\Sigma}_2$ is a monad with an A_1-operation and an A_2-operation such that the order of an adjacent pair of an A_1-operation and an A_2-operation can be swapped:

$$\textbf{do}\ \{x \leftarrow op_1; y \leftarrow op_2; k\ x\ y\} = \textbf{do}\ \{y \leftarrow op_2; x \leftarrow op_1; k\ x\ y\}.$$

The Day tensor $A_1 * A_2$ shows up in the equation to ensure that each of the two operations do not depend on the other's output, so that it makes sense to swap them. The functor $\otimes$ can be extended to a monoidal product on $\text{Alg}(\mathscr{E})$ with the theory of monoids with no operations

as the unit. Let $\mathcal{F}$ be $\textsc{Alg}(\mathscr{E})$ and $\ddot{\Psi} \in \mathcal{F}$, a lifting of $- \otimes \ddot{\Psi}$ along $P : \mathcal{F}\text{-}\textsc{Alg} \to \mathcal{F}$ is then a modular model of $\ddot{\Psi}$ such that operations of $\ddot{\Psi}$ commute with all other operations, even if they may be unknown now. This may sound very strong, but the *state monad transformer* gives such a modular model (Yang and Wu, 2021, §6).

7.2∗3 (Modular handlers). The theory of modular models is inspired by the concept of *modular handlers* studied by Schrijvers et al. (2019) and Yang and Wu (2021) in Haskell. If we re-express them in category theory, a reasonable definition of *a modular handler of an equational system* $\dot{\Sigma}$ *over a category* $\mathscr{C}$ is a mapping (not necessarily a functor) from monads M on $\mathscr{C}$ to tuples $\langle A, a, f \rangle$ of an object $A \in \mathscr{C}$, a $\dot{\Sigma}$-algebra $a : \Sigma A \to A$ on A, and an Eilenberg-Moore algebra of M $f : MA \to A$ on A:

$$\Sigma A \xrightarrow{\ a\ } A \xleftarrow{\ f\ } MA$$

The idea is similar to our modular models: for whatever 'ambient effect' M, there is a model A of the effect $\dot{\Sigma}$, which also models any effect that M supports. In particular, if M has an algebraic operation $b : S \circ M \to M$ on M, then the object A can 'forward' this operation by

$$SA \xrightarrow{\ S\eta_A\ } S(MA) \xrightarrow{\ b\ } MA \xrightarrow{\ f\ } A.$$

Modular handlers in this sense are also an instance of liftings along fibrations. Let $|\textsc{Mnd}(\mathscr{C})|$ be the *discrete* category of monads over $\mathscr{C}$, then the identity functor $\mathrm{Id} : |\textsc{Mnd}(\mathscr{C})| \to |\textsc{Mnd}(\mathscr{C})|$ is a fibration. For every $\dot{\Sigma} \in \textsc{Eqs}(\mathscr{C})$, we define the functor $T_{\dot{\Sigma}} : |\textsc{Mnd}(\mathscr{C})| \to \textsc{Eqs}(\mathscr{C})$ to be $(-)\text{-}\textsc{Act} + \dot{\Sigma}$, where $(-)\text{-}\textsc{Act}$ is the equational system of monad algebras from 3.1∗8. Let $P : \textsc{Alg}(\mathscr{C}) \to \textsc{Eqs}(\mathscr{C})$ be the Grothendieck construction of $(-)\text{-}\textsc{Alg} : \textsc{Eqs}(\mathscr{C})^{\mathrm{op}} \to \mathrm{CAT}$. A modular handler H of $\dot{\Sigma}$ is then a lifting of $T_{\dot{\Sigma}}$ along Id, P:

$$\begin{array}{ccc} |\textsc{Mnd}(\mathscr{C})| & \xrightarrow{\ H\ } & \textsc{Alg}(\mathscr{C}) \\ {\scriptstyle \mathrm{Id}}\downarrow & & \downarrow{\scriptstyle P} \\ |\textsc{Mnd}(\mathscr{C})| & \xrightarrow[\ T_{\dot{\Sigma}}\ ]{} & \textsc{Eqs}(\mathscr{C}) \end{array}$$

In fact, most examples of modular handlers from Schrijvers et al. (2019) and Yang and Wu (2021) are covariant functors $\textsc{Mnd}(\mathscr{C}) \to \textsc{Alg}(\mathscr{C})$, apart from those based on *continuation monad transformers* $M \mapsto (- \Rightarrow MR) \Rightarrow MR$.

7.2∗4 (Output Effects). An intuition for effect handlers is that they *consume* effectful operations, but in almost all implementations of effect handlers, handlers can also *produce* effectful operations. For example, a handler may handle an exception by producing a nondeterministic failure, thus transforming the effect of exceptions to the effect of nondeterminism. Such handlers with both *input effect* (the operations to be handled) and *output effect* (the operations to be generated) can be modelled as liftings along fibrations as well.

Let $\mathcal{F}$ be an operation family (Definition 6∗2) closed under binary coproducts. For every $\ddot{\Psi} \in \mathcal{F}$, we denote by $\mathrm{U}_{\ddot{\Psi}} : \ddot{\Psi}/\mathcal{F} \to \mathcal{F}$ the forgetful functor from the coslice category under $\ddot{\Psi} \in \mathcal{F}$ to $\mathcal{F}$. We also denote by $P_{\ddot{\Psi}} : (\ddot{\Psi}/\mathcal{F})\text{-}\textsc{Alg} \to \ddot{\Psi}/\mathcal{F}$ the change of base of the fibration $P : \mathcal{F}\text{-}\textsc{Alg} \to \mathcal{F}$ along the functor $\mathrm{U}_{\ddot{\Psi}}$:

$$\begin{array}{ccc} (\ddot{\Psi}/\mathcal{F})\text{-}\textsc{Alg} & \longrightarrow & \mathcal{F}\text{-}\textsc{Alg} \\ {\scriptstyle P_{\ddot{\Psi}}}\downarrow & \lrcorner & \downarrow{\scriptstyle P} \\ \ddot{\Psi}/\mathcal{F} & \xrightarrow[\ \mathrm{U}_{\ddot{\Psi}}\ ]{} & \mathcal{F} \end{array}$$

For every pair of $\ddot{\Sigma}, \ddot{\Psi} \in \mathcal{F}$, we define a functor $T_{\ddot{\Sigma},\ddot{\Psi}} : \ddot{\Sigma}/\mathcal{F} \to \ddot{\Psi}/\mathcal{F}$

$$T_{\ddot{\Sigma},\ddot{\Psi}}\,\langle \ddot{\Phi} \in \mathcal{F}, S : \ddot{\Sigma} \to \ddot{\Phi}\rangle = \langle \ddot{\Phi} + \ddot{\Psi}, \iota_2 : \ddot{\Psi} \to \ddot{\Phi} + \ddot{\Psi}\rangle$$

A modular model M of input effect $\ddot{\Psi}$ and output effect $\ddot{\Sigma}$ is then a lifting of the functor $T_{\ddot{\Sigma},\ddot{\Psi}}$ along the fibrations $P_{\ddot{\Sigma}}$ and $P_{\ddot{\Psi}}$:

$$\begin{array}{ccc} (\ddot{\Sigma}/\mathcal{F})\text{-ALG} & \xrightarrow{M} & (\ddot{\Psi}/\mathcal{F})\text{-ALG} \\ {\scriptstyle P_{\ddot{\Sigma}}}\downarrow & & \downarrow{\scriptstyle P_{\ddot{\Psi}}} \\ \ddot{\Sigma}/\mathcal{F} & \xrightarrow[T_{\ddot{\Sigma},\ddot{\Psi}}]{} & \ddot{\Psi}/\mathcal{F} \end{array} \tag{7.9}$$

Note that it is $\ddot{\Psi}$ rather than $\ddot{\Sigma}$ that is the *input* effect, since $\ddot{\Psi}$ is the effect to be 'handled' by this modular model; this is similar to how translations of equational systems $\dot{\Psi} \to \dot{\Sigma}$ are functors $\dot{\Sigma}$-ALG $\to \dot{\Psi}$-ALG from the opposite direction.

7.2∗5. In the diagram (7.9), $P_{\ddot{\Psi}}$ is a pullback of $P : \mathcal{F}\text{-ALG} \to \mathcal{F}$ along $\mathrm{U}_{\ddot{\Psi}}$, so functors M making (7.9) commute are in bijection with functors N making

$$\begin{array}{ccccc} (\ddot{\Sigma}/\mathcal{F})\text{-ALG} & & \xrightarrow{\quad N \quad} & & \mathcal{F}\text{-ALG} \\ {\scriptstyle P_{\ddot{\Sigma}}}\downarrow & & & & \downarrow{\scriptstyle P} \\ \ddot{\Sigma}/\mathcal{F} & \xrightarrow[T_{\ddot{\Sigma},\ddot{\Psi}}]{} & \ddot{\Psi}/\mathcal{F} & \xrightarrow[\mathrm{U}_{\ddot{\Psi}}]{} & \mathcal{F} \end{array}$$

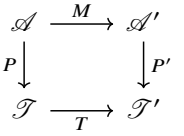
commute. When $\ddot{\Sigma}$ is the theory of monoids with no operations, this recovers our earlier definition of modular models without output effects (7.8).

7.2∗6. Moreover, the fibration $\mathcal{F}$-ALG $\to \mathcal{F}$ in 7.2∗4 can be replaced by many other fibrations whose base category is a category of some notion of algebraic theories and total category is a category of pairs of a theory and its model. For example, let FPCAT be the category of small categories with finite products and finite-product-preserving functors; there is a functor $[-, \text{SET}] : \text{FPCAT}^{\text{op}} \to \text{CAT}$ sending every $\mathscr{C} \in \text{FPCAT}$ to the category of finite-product-preserving functors $\mathscr{C} \to \text{SET}$, which induces a fibration $P : \text{FPMOD} \to \text{FPCAT}$, and we can talk about modular models of finite-product theories by replacing $\mathcal{F}$-ALG $\to \mathcal{F}$ in 7.2∗4 with this fibration. The same thing can be said for *generalised algebraic theories* (Cartmell, 1986), *second-order algebraic theories* (Fiore and Mahmoud, 2010), and so on for any framework of algebraic theories that have a fibration of models over theories and useful ways of combining theories such as coproducts.

7.2∗7. Motivated by the above examples of liftings, we will use 'model transformers' as a suggestive synonym for liftings along fibrations. (We could similarly call the functor $T : \mathscr{T} \to \mathscr{T}'$ a 'theory transformer'.) Incidentally, Hyland et al. (2006) have a concept of 'operation transformers', but they correspond to translations between algebraic theories rather than functors between the categories of theories.

7.2∗8 Definition. Given two fibrations $P : \mathscr{A} \to \mathscr{T}$, $P' : \mathscr{A}' \to \mathscr{T}'$ and a functor $T : \mathscr{T} \to \mathscr{T}'$, a (strong/strict) *model transformer* M of T is defined to a (fibred/split) lifting of T

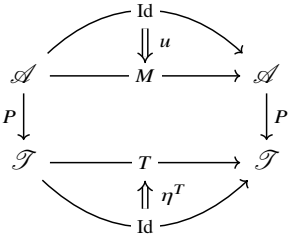
along P and P' (Definition 7.1∗14).



The category $\textsc{Lift}_{P,P'}(T)$ of liftings will also be denoted by $\textsc{Motr}(T)$.

Specially, model transformers of a coproduct functor $- + \Psi : \mathscr{T} \to \mathscr{T}$ will be called *modular models of* $\Psi \in \mathscr{T}$, reusing the terminology from the last section.

7.2∗9. We can similarly generalise *updaters* (Definition 7.1∗21) to general fibrations. It seems natural to require the two fibrations to be the same in this case. Given a fibration $P : \mathscr{A} \to \mathscr{T}$ and a pointed endofunctor $\langle T : \mathscr{T} \to \mathscr{T}, \eta^T : \mathrm{Id} \to \mathscr{T} \rangle$, an *updater* for a model transformer of T along P is a natural transformation $u : \mathrm{Id} \to M$ such that $P \circ u = \eta^T \circ P$:



We denote by $\textsc{Motr}_u(T)$ the category of model transformers of T equipped with an updater and whose morphisms $\langle M, u \rangle \to \langle N, v \rangle$ are morphisms $\sigma : M \to N$ that are compatible with the updaters: $v = \sigma \cdot u : \mathrm{Id} \to N$.

8 Constructions of Model Transformers

8∗1. The definition of model transformers is just a mathematical formulation of semantic modularity, so in this section we will have a look at some concrete examples and general constructions of model transformers. Each subsection below discusses a construction and is basically independent of each other:

- initial model transformers (Section 8.1),
- initial model transformers with an updater (Section 8.2),
- free model transformers (Section 8.3),
- modular models from monoids transformers (Section 8.4),
- limits and colimits of model transformers (Section 8.5),
- modular models in symmetric monoidal categories (Section 8.7).

8.1 Initial Model Transformer

8.1∗1. We begin with the *initial model transformer*. Let $P : \mathscr{A} \to \mathscr{T}$ be a fibration and $P' : \mathscr{A}' \to \mathscr{T}'$ be a fibration with a cleavage κ such that for every object $\Sigma \in \mathscr{T}'$, the fiber

category $\mathscr{A}'_\Sigma$ has a chosen initial object $\mu\Sigma$. Then for every functor $T : \mathscr{T} \to \mathscr{T}'$, we define a functor $0_T : \mathscr{A} \to \mathscr{A}'$ such that

- for every object $A \in \mathscr{A}$, $0_T A := \mu(TPA)$,
- for every morphism $f : A \to B \in \mathscr{A}$, $0_T f := \kappa(\mu(TPB), TPf) \cdot v$, where the morphism $v : \mu(TPA) \to X$ is the unique morphism out of the initial object $\mu(TPA)$ of the fiber category $\mathscr{A}'_{TPA}$:

$$\begin{array}{ccc} & & \mu(TPA) \\ & & \downarrow{\scriptstyle v} \\ A & \xrightarrow{f} & B \end{array} \qquad \begin{array}{ccc} & & \mu(TPA) \\ & & \downarrow{\scriptstyle v} \\ X & \xrightarrow{\kappa(\mu(TPB),TPf)} & \mu(TPB) \end{array}$$

$$PA \xrightarrow[Pf]{} PB \qquad TPA \xrightarrow[TPf]{} TPB$$

8.1∗2 Theorem. *In the situation of* 8.1∗1*, the functor* $0_T : \mathscr{A} \to \mathscr{A}'$ *is a model transformer of* $T : \mathscr{T} \to \mathscr{T}'$ *and is initial in the category* $\textsc{Motr}(T)$.

$$\begin{array}{ccc} \mathscr{A} & \xrightarrow{0_T} & \mathscr{A}' \\ {\scriptstyle P}\downarrow & & \downarrow{\scriptstyle P'} \\ \mathscr{T} & \xrightarrow[T]{} & \mathscr{T}' \end{array}$$

Moreover, when the cleavage κ *of* P' *preserves the chosen initial objects up to isomorphism (resp. strictly), i.e. for all* $f : \Gamma \to \Sigma \in \mathscr{T}'$*, the domain of* $\kappa(\mu\Sigma, f)$ *is initial in* $\mathscr{A}'_\Gamma$ *(resp. exactly* $\mu\Gamma$*),* 0_T *is a strong (resp. strict) model transformer.*

Proof. The functor 0_T satisfies $P' \circ 0_T = T \circ P$ by construction, so it is a model transformer by definition (Definition 7.2∗8). Given any $H : \mathscr{A} \to \mathscr{A}'$ such that $P' \circ H = T \circ P$, for every $A \in \mathscr{A}$, HA and $0_T A$ are both in the fibre category $\mathscr{A}'_{TPA}$, so by the initiality of $0_T A$, there is a *unique* vertical morphism $u_A : 0_T A \to HA$. To show that u_A is natural, consider every $f : A \to B$ and the morphism $\kappa(HB, TPf) : Y \to HB$, we have the following situation

$$\begin{array}{ccccc} 0_T A = \mu(TPA) & \xrightarrow{v} & X & \xrightarrow{\kappa(0_T B, TPf)} & \mu(TPB) = 0_T B \\ {\scriptstyle u_A}\downarrow & & \downarrow{\scriptstyle h} & & \downarrow{\scriptstyle u_B} \\ HA & \xrightarrow{k} & Y & \xrightarrow{\kappa(HB,TPf)} & HB \\ {\scriptstyle P}\downarrow & & & \searrow_{Hf} & \downarrow{\scriptstyle P'} \\ TPA & & & \xrightarrow[f]{} & TPB \end{array}$$

where $v : 0_T A \to X$ is the unique vertical morphism from $\mu(TPA)$ to X, and $h : X \to Y$ is the unique vertical morphism making the upper-right square commute, obtained from the cartesianess of $\kappa(HB, TPf) : Y \to HB$. Similarly, k is the unique vertical morphism satisfying $\kappa(HB, TPf) \cdot k = Hf$. Note that $0_T f$ is exactly the upper path $\kappa(0_T B, TPf) \cdot v$, and the upper-left square commutes by the initiality of $0_T A$. Hence we have the commutativity of the large rectangle, which is the naturality of $u : 0_T \to H$. This concludes the proof of the initiality of the model transformer 0_T in $\textsc{Motr}(T)$.

For the second part of the theorem, if the cleavage κ preserves initial algebras up to isomorphism, the object X in the diagram above is also initial among $\mathscr{A}'_{TPA}$, so the morphism $v : 0_T A \to X$ is a vertical isomorphism, and $0_T f = \kappa(0_T B, TPf) \cdot v$ is also cartesian. The case for splitting fibrations is similar. □

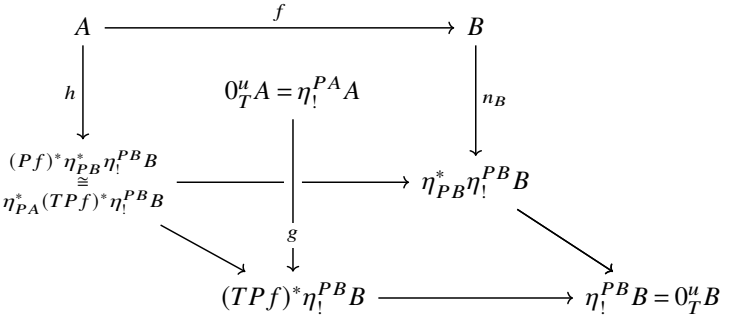
8.1∗3 Example. Let $\mathcal{F}$ be an operation family of monoids such that $\ddot{\Sigma}$-Alg has chosen initial algebras for all $\ddot{\Sigma} \in \mathcal{F}$. Applying the construction of initial model transformers to the situation of 7.1∗15, where $T = - + \ddot{\Psi}$ for some $\ddot{\Psi} \in \mathcal{F}$, the model transformer 0_T then maps every $\langle \ddot{\Sigma}, A, \alpha \rangle$ to the initial algebra of $\ddot{\Sigma} + \ddot{\Psi}$, ignoring the 'existing model' $\langle A, \alpha \rangle$ of the 'existing syntax' $\ddot{\Sigma}$ completely. Therefore the initial model transformer does not have an updater (Definition 7.1∗21) in general.

8.2 Initial Updatable Model Transformers

8.2∗1. Instead of completely ignoring the existing model, we can alternatively consider the *free model* of the new syntax over the existing model. As a special case, let $\mathcal{F}$ be an operation family closed under coproducts, for every $\ddot{\Sigma}, \ddot{\Psi} \in \mathcal{F}$, Theorem 3.2∗14 provides a sufficient condition for the forgetful functor $U : (\ddot{\Sigma} + \ddot{\Psi})$-Alg $\to \ddot{\Sigma}$-Alg to have a left adjoint $F : \ddot{\Sigma}$-Alg $\to (\ddot{\Sigma} + \ddot{\Psi})$-Alg, the idea is to define a model transformer of $\ddot{\Psi} \in \mathcal{F}$ by sending every $\langle \ddot{\Sigma}, A, \alpha \rangle \in \mathcal{F}$-Alg to $F\langle A, \alpha \rangle$. Moreover, the unit of the adjunction $F \dashv U$ defines an updater. The universal property of the obtained model transformer is that it is the initial one in the category of model transformers with an updater (7.2∗9), so we will call it the *initial updatable model transformer* for short.

8.2∗2 Theorem. *Let $P : \mathscr{A} \to \mathscr{T}$ be a fibration with a cleavage such that for every morphism $t : \Gamma \to \Sigma$ in $\mathscr{T}$, there is an adjunction $t_! \dashv t^* : \mathscr{A}_\Sigma \to \mathscr{A}_\Gamma$. Let $\langle T, \eta \rangle$ be a pointed endofunctor on $\mathscr{T}$. The category of* $\textsc{Motr}_u(T)$ *of model transformers of T with an updater as defined in 7.2∗9 has an initial object.*

Proof. We define a model transformer $0^u_T : \mathscr{A} \to \mathscr{A}$ as follows. For every object $A \in \mathscr{A}$, we have a functor $\eta_!^{PA} : \mathscr{A}_{PA} \to \mathscr{A}_{TPA}$ left adjoint to $\eta^*_{PA} : \mathscr{A}_{TPA} \to \mathscr{A}_{PA}$, and we define $0^u_T A := \eta_!^{PA} A$. For every $f : A \to B \in \mathscr{A}$, let $n_B : B \to \eta^*_{PB}\eta_!^{PB} B$ be the unit of the adjunction $\eta_!^{PB} \dashv \eta^*_{PB}$. Since n_B is a morphism in $\mathscr{A}_{PB}$, it is vertical: $Pn_B = id_{PB}$, and $P(n_B \cdot f) = Pf$. Therefore there is a unique $h : A \to (Pf)^*\eta^*_{PB}\eta_!^{PB} B$ making the upper square commute:



The unlabelled morphisms are the evident cartesian morphisms from the cleavage κ. By the naturality of $\eta : \mathrm{Id} \to T$, we have the commutativity of the bottom square in $\mathscr{T}$: $\eta_{PB} \cdot Pf = TPf \cdot \eta_{PA}$. Reindexing is pseudofunctorial, so we have

$$(Pf)^* \eta_{PB}^* \eta_!^{PB} B \quad \cong \quad \eta_{PA}^* (TPf)^* \eta_!^{PB} B.$$

Hence the morphism h determines a morphism $A \to \eta_{PA}^* (TPf)^* \eta_!^{PB} B$, which by the adjunction $\eta_!^{PA} \dashv \eta_{PA}^*$, determines a vertical morphism

$$g : 0_T^u A = \eta_!^{PA} A \to (TPf)^* \eta_!^{PB} B.$$

By composing g with $\kappa(0_T^u B, TPf) : (TPf)^* \eta_!^{PB} B \to 0_T^u B$, we obtain a morphism $0_T^u A \to 0_T^u B$, which is our definition of the action of 0_T^u the morphism $f : A \to B$. We omit the checking of the functoriality of $0_T^u : \mathscr{A} \to \mathscr{A}$ here.

The functor 0_T^u by construction is a lifting of $T : \mathscr{T} \to \mathscr{T}$ along the fibration $P : \mathscr{A} \to \mathscr{T}$. It has an updater $u : \mathrm{Id} \to 0_T^u$ given by

$$u_B := \kappa(\eta_{PB}, \eta_!^{PB} B) \cdot n_B : B \to \eta_!^{PB} B = 0_T^u B$$

for all $B \in \mathscr{A}$ as in the diagram (8.1) above. To see the naturality of u, for all $f : A \to B$, the two morphisms $u_B \cdot f$ and $0_T^u f \cdot u_A$ are over the same morphism in $\mathscr{T}$, i.e. $\eta_{PB} \cdot Pf = TPf \cdot \eta_{PA}$, so it is sufficient to show that their induced vertical morphisms in $\mathscr{A}_{PA}$ are equal. It can be calculated that the one for $u_B \cdot f$ is h as in (8.1) and the one for $0_T^u f \cdot u_A$ is

$$(\eta_{PA}^* g) \cdot n_A = \eta_{PA}^* (e_A \cdot \eta_!^{PA} h) \cdot n_A \tag{8.2}$$

where $e_A : \eta_!^{PA} \eta_{PA}^* A \to A$ is the counit of the adjunction $\eta_!^{PA} \dashv \eta_{PA}^*$. Using naturality and triangle identity of the unit n and counit e of the adjunction $\eta_!^{PA} \dashv \eta_{PA}^*$, the morphism (8.2) can be shown to be exactly h.

As for the initiality of $\langle 0_T^u, u \rangle$ in $\textsc{Motr}_u(T)$, for every $\langle M, v \rangle \in \textsc{Motr}_u(T)$ and $B \in \mathscr{A}$, $v_B : B \to MB$ is over η_{PB} by the definition of updaters. Hence there is a vertical morphism $w : B \to \eta_!^{PB} MB$ such that $v_B = \kappa(\eta_{PB}, MB) \cdot w$. By the universal property of $n_B : B \to \eta_{PB}^* \eta_!^{PB} B$, there is a *unique* morphism $\sigma_B : \eta_!^{PB} B = 0_T^u B \to MB$ such that $\eta_{PB}^* \sigma_B \cdot n_B = w$:

$$\begin{array}{ccc} B & \xrightarrow{n_B} & \eta_{PB}^* \eta_!^{PB} B \\ & \searrow^{w} & \downarrow{\scriptstyle \eta_{PB}^* \sigma_B} \\ & & \eta_{PB}^* MB \end{array}$$

We omit the checking of the naturality of σ here (it is similarly to the proof of Theorem 8.1∗2). Since every $\tau : \langle 0_T^u, u \rangle \to \langle M, v \rangle$ is a vertical natural transformation $\tau : 0_T^u \to M$ satisfying $\tau_B \cdot u_B = v_B$, it must satisfy that $\eta_{PB}^* \tau_B \cdot n_B = w$. Therefore, σ is the unique morphism $\langle 0_T^u, u \rangle \to \langle M, v \rangle$ in $\textsc{Motr}_u(T)$. □

8.2∗3 Example. Let $\mathscr{C}$ be a category and $\mathscr{A}$ be a *freeness condition* for $\mathscr{C}$ (Definition 3.1∗21) that satisfies the assumption of Theorem 3.2∗14 (for example, preserving colimits of α-chains for some limit ordinal α and $\mathscr{C}$ being cocomplete). The fibration $P : \textsc{Alg}_{\mathscr{A}}(\mathscr{C}) \to \textsc{Eqs}_{\mathscr{A}}(\mathscr{C})$ then satisfies the condition of Theorem 8.2∗2, so the pointed functor $(- + \dot{\Psi}) : \textsc{Eqs}_{\mathscr{A}}(\mathscr{C}) \to \textsc{Eqs}_{\mathscr{A}}(\mathscr{C})$ for every $\dot{\Psi} \in \textsc{Eqs}_{\mathscr{A}}(\mathscr{C})$ then has an initial updatable model transformer.

8.2∗4 Example. Let $\mathscr{E}$ be a cocomplete monoidal category such that the monoidal product $\Box : \mathscr{E} \times \mathscr{E} \to \mathscr{E}$ preserves colimits of α-chains for some limit ordinal α. Let $\mathcal{F} \subseteq \textsc{Mon}/\textsc{Eqs}(\mathscr{E})$ be the operation family containing all equational systems whose signature and context colimits of α-chains for some limit ordinal α. By Theorem 3.2∗14, for every morphism $T : \ddot{\Sigma} \to \ddot{\Psi} \in \mathcal{F}$, the corresponding functor $T : \ddot{\Psi}\text{-}\textsc{Alg} \to \ddot{\Sigma}\text{-}\textsc{Alg}$ has a left adjoint. Therefore the fibration $P : \mathcal{F}\text{-}\textsc{Alg} \to \mathcal{F}$ satisfies the condition of Theorem 8.2∗2, and for every $\ddot{\Psi} \in \mathcal{F}$, the functor $- + \ddot{\Psi} : \mathcal{F} \to \mathcal{F}$ has an initial updater model transformer $0^{u}_{-+\ddot{\Psi}}$, which maps every $\ddot{\Sigma}$-algebra A to the relative free $(\ddot{\Sigma} + \ddot{\Psi})$-algebra over A.

8.2∗5 Example. For a concrete example, let us instantiate $\mathscr{E}$ in the previous example to be $\langle \textsc{Set}^{\textsc{Fin}}, \bullet, V \rangle$ from 2.2∗2. As we mentioned in 6∗21, the syntax of untyped λ-calculus can be presented as an equational system $\Lambda = \Sigma_{\langle O,a \rangle}\text{-}\textsc{Mon}$ for the binding signature $O = \{app, abs\}$ with $a(abs) = \langle 1 \rangle$ and $a(app) = \langle 0, 0 \rangle$. Models of Λ can be obtained from *reflexive objects* $U \cong U^U$ in any cartesian closed category C: every U induces a functor $\bar{U} : \textsc{Fin} \to \textsc{Set}$ with $n \mapsto \mathscr{C}(U^n, U)$. The functor $\bar{U}$ has a monoid structure $[\eta_U, \mu_U]$ (similar to that of the continuation monad), and it is a model of Λ (Hyland, 2017):

$$abs_U : \bar{U}^V n \cong \mathscr{C}(U^{n+1}, U) \cong \mathscr{C}(U^n, U^U) \cong \mathscr{C}(U^n, U) \cong \bar{U}n$$
$$app_U : (\bar{U} \times \bar{U})n \cong \mathscr{C}(U^n, U \times U) \cong \mathscr{C}(U^n, U^U \times U) \xrightarrow{eval_{U^U} \cdot -} \mathscr{C}(U^n, U) \cong \bar{U}n$$

Now consider the theory $\textsc{St}_S$ of mutable state (Example 5∗9) for some finite set S. Its initial updatable model transformer maps the Λ-model on $\bar{U}$ to a $(\Lambda +_{\textsc{Mon}} \textsc{St}_S)$-model whose carrier is the initial algebra

$$\mu X.\, \bar{U} + X \bullet X + V + X^V + X \times X + \textstyle\prod_S X + \coprod_S X : \textsc{Fin} \to \textsc{Set}$$

quotiented by equations of Λ and $\textsc{St}_S$ and equations saying that Λ-operations of the initial algebra acting on $\bar{U}$ are the same as the model $[\eta_U, \mu_U, abs_U, app_U]$ of $\bar{U}$.

8.2∗6. Left adjoints to reindexing functors of a fibration are used for modelling Σ-types and $\exists$-quantification in categorical logic, where the fibration models types or predicates over a type, and reindexing models substitution. In this context, reindexing functors typically preserve the left adjoints suitably, called satisfying the *Beck-Chevalley condition* (Jacobs, 1999, Definition 1.9.4), reflecting the fact substitution commutes with Σ and $\exists$. Dually, in this context reindexing functors usually have right adjoints, which model Π or $\forall$.

However, for fibrations of algebras and theories, the reindexing functors typically do *not* have right adjoints (the translation functors between categories of algebras almost never preserve colimits), and reindexing functors typically do *not* preserve the left adjoints (relative free algebras), so initial updatable model transformers typically are not strong or strict. For example, consider $\mathscr{E} = \langle \textsc{Set}, \times, 1 \rangle$. We have the following theories in the family $\textsc{Mon}/\textsc{Eqs}_{\mathscr{A}}(\mathscr{E})$: Mon with the identity translation; Grp with the inclusion translation $T : \textsc{Mon} \to \textsc{Grp}$ (Example 3.2∗4); and the theory BLat of bounded lattices with the translation that maps monoid multiplication to lattice join $\vee$, monoid identity to lattice bottom $\bot$. The following diagram in CAT does not commute:

$$\begin{array}{ccc} \textsc{Mon}\text{-}\textsc{Alg} & \xleftarrow{\quad T \quad} & \textsc{Grp}\text{-}\textsc{Alg} \\ \Big\downarrow{\scriptstyle F_{\textsc{Mon}}} & & \Big\downarrow{\scriptstyle F_{\textsc{Grp}}} \\ (\textsc{Mon} + \textsc{BLat})\text{-}\textsc{Alg} & \xleftarrow{\quad T+\textsc{BLat} \quad} & (\textsc{Grp} + \textsc{BLat})\text{-}\textsc{Alg} \end{array}$$

The theory Mon + BLat, bounded lattices whose join $\vee$ and $\bot$ form a monoid, is isomorphic to BLat since $\langle \vee, \bot \rangle$ of a lattice is already a monoid, so $\mathrm{F}_{\mathrm{Mon}}(TG)$ for every group G is the free bounded lattice over G as a monoid. On the other hand, the theory Grp + BLat, bounded lattices whose $\langle \vee, \bot \rangle$ form a group, has only trivial models since for every element x, $\bot = x \vee x^{-1}$, and by the idempotent law of join, $\bot = (x \vee x) \vee x^{-1} = x \vee (x \vee x^{-1}) = x \vee \bot = x$. Therefore $(T + \mathrm{BLat})(\mathrm{F}_{\mathrm{Grp}} G)$ is always the trivial bounded lattice for every group G.

8.3 Free Model Transformers over Ordinary Models

8.3∗1. Consider model transformers of $- + \ddot{\Psi} : \mathcal{F} \to \mathcal{F}$ on an operation family $\mathcal{F}$: for every $\ddot{\Sigma} \in \mathcal{F}$ and every $A \in \ddot{\Sigma}$-Alg, the initial model transformer $0_{\ddot{\Psi}}$ just ignores the algebra A and freely generates a model of $\ddot{\Sigma} + \ddot{\Psi}$, so it is not a practically interesting model transformer. In comparison, the initial updatable model transformer $0^{u}_{\ddot{\Psi}}$ takes into account of A and freely generates a model of $\ddot{\Sigma} + \ddot{\Psi}$ that has a $\ddot{\Sigma}$-homomorphism from A.

The natural next step is then freely generating a model of $\ddot{\Sigma} + \ddot{\Psi}$ that has both a $\ddot{\Sigma}$-homomorphism from A and a $\ddot{\Psi}$-homomorphism from some fixed $B \in \ddot{\Psi}$-Alg, using the construction in Example 3.2∗18. In this way, we can turn an ordinary model B of $\ddot{\Psi}$ to a model transformer of $\ddot{\Psi}$, and it is also going to be the *free* way.

8.3∗2. In this subsection, we fix a fibration $P : \mathscr{A} \to \mathscr{T}$ with a cleavage such that $\mathscr{T}$ has finite coproducts, and we fix a functor $\oplus : \mathscr{T} \times \mathscr{T} \to \mathscr{T}$ equipped with a natural transformation $\tau : + \to \oplus$. For all $\Sigma, \Gamma \in \mathscr{T}$, we define

$$\kappa_1 := (\Sigma \xrightarrow{\iota_1} \Sigma + \Gamma \xrightarrow{\tau} \Sigma \oplus \Gamma) \qquad \kappa_2 := (\Gamma \xrightarrow{\iota_2} \Sigma + \Gamma \xrightarrow{\tau} \Sigma \oplus \Gamma)$$

The natural transformation $\kappa_1 : - \to - \oplus \Gamma$ makes $- \oplus \Gamma$ a pointed functor, enabling us to talk about updaters (7.2∗9) of the functor $- \oplus \Gamma$.

As usual, the category $\mathscr{T}$ is expected to be a category of some notion of algebraic theories and each fiber category $\mathscr{A}_\Sigma$ is the category of models of $\Sigma \in \mathscr{T}$. For example, P can be the fibration $\mathcal{F}$-Alg $\to \mathcal{F}$ for an operation family and $\oplus$ can be just the coproduct or the commutative combination (7.2∗2).

8.3∗3 Definition. When the fiber category $\mathscr{A}_0$ of the initial object $0 \in \mathscr{T}$ also has an initial object I, for every $\Gamma \in \mathscr{A}$ we define a functor $\mathrm{U}_{\oplus\Gamma} : \mathrm{Mod}_{\mathrm{TR}}(- \oplus \Gamma) \to \mathscr{A}_\Gamma$:

$$\mathrm{U}_{\oplus\Gamma}\, M := \kappa_2^* M I$$

where κ_2^* is the reindexing functor $\mathscr{A}_{0 \oplus \Gamma} \to \mathscr{A}_\Gamma$. The functor $\mathrm{U}_{\oplus\Gamma}$ is intuitively the forgetful functor from model transformers of ($\oplus$-combination with) Γ to ordinary models of Γ. Composing $\mathrm{U}_{\oplus\Gamma}$ with the functor that forgets updaters (7.2∗9), we also have a functor $\mathrm{Mod}_{\mathrm{TR}u}(- \oplus \Gamma) \to \mathscr{A}_\Gamma$ that we shall also denote by $\mathrm{U}_{\oplus\Gamma}$.

8.3∗4 Theorem. *Assume that all reindexing functors $!^* : \mathscr{A}_\Gamma \to \mathscr{A}_0$ to the fiber category over the initial object $0 \in \mathscr{T}$ is monadic and every fiber category $\mathscr{A}_\Sigma$ is finitely cocomplete. The functor $\mathrm{U}_{\oplus\Gamma} : \mathrm{Mod}_{\mathrm{TR}u}(- \oplus \Gamma) \to \mathscr{A}_\Gamma$ for every $\Gamma \in \mathscr{T}$ defined in Definition 8.3∗3 has a left adjoint $\mathrm{F}_{\oplus\Gamma} : \mathscr{A}_\Gamma \to \mathrm{Mod}_{\mathrm{TR}u}(- \oplus \Gamma)$.*

Proof. Every fibration is equivalent to a split one (Jacobs, 1999, Corollary 5.2.5), and the statement is stable under equivalence of fibrations, so we can assume without loss of

generality that P is a split fibration. By Mac Lane (1998, §IV.1 Theorem 2), it is sufficient to construct for every $\langle B, \beta\rangle \in \mathscr{A}_\Gamma$ a model transformer with an updater $M^B \in \textsc{Mod}_{\textsc{tr}_u}(- \oplus \Gamma)$ and a universal arrow $e : \langle B, \beta\rangle \to \mathrm{U}_{\oplus\Gamma} M^B$.

First we observe that for every $\Sigma \in \mathscr{T}$, we have the following functors:

$$\begin{array}{ccc} \mathscr{A}_\Sigma \times \mathscr{A}_\Gamma & \xleftarrow{\langle \kappa_1^*, \kappa_2^*\rangle} & \mathscr{A}_{\Sigma\oplus\Gamma} \\ \downarrow{\scriptstyle !_\Sigma^* \times !_\Gamma^*} & & \downarrow{\scriptstyle !_{\Sigma\oplus\Gamma}^*} \\ \mathscr{A}_0 \times \mathscr{A}_0 & \xleftarrow{\Delta} & \mathscr{A}_0 \end{array}$$

This diagram commutes strictly since $\kappa_1 \cdot !_\Sigma = !_{\Sigma\oplus\Gamma} = \kappa_2 \cdot !_\Gamma : 0 \to \Sigma \oplus \Gamma$, and we have assumed that P is a split fibration. By the assumption that each fiber category is finitely cocomplete, the category $\mathscr{A}_0$ has binary coproducts, so we have an adjunction $+ \dashv \Delta : \mathscr{A}_0 \to \mathscr{A}_0 \times \mathscr{A}_0$, and moreover the fiber category $\mathscr{A}_{\Sigma\oplus\Gamma}$ has coequalisers. By assumption $!_\Sigma^*$, $!_\Gamma^*$, and $!_{\Sigma\oplus\Gamma}^*$ are monadic functors. The product of monadic functors is also monadic, so $!_\Sigma^* \times !_\Gamma^*$ is monadic. By Borceux (1994, Theorem 4.5.6) (c.f. our discussion in 3.2∗16), the functor $\langle \kappa_1^*, \kappa_2^*\rangle$ on the top of the diagram has a left adjoint $F_{\Sigma\oplus\Gamma} : \mathscr{A}_\Sigma \times \mathscr{A}_\Gamma \to \mathscr{A}_{\Sigma\oplus\Gamma}$.

Now we define M^B via an oplax transformation by Lemma 7.1∗17:

$$M^B : \mathscr{A}_- \to \mathscr{A}_{-\oplus\Gamma} : \mathscr{T}^{\mathrm{op}} \to \mathrm{CAT}.$$

We define the component M^B_Σ at every $\Sigma \in \mathscr{T}$ to be $F_{\Sigma\oplus\Gamma}\langle -, B\rangle : \mathscr{A}_\Sigma \to \mathscr{A}_{\Sigma\oplus\Gamma}$. For every morphism $t : \Sigma \to \Phi$, we have the following functors:

$$\begin{array}{ccc} \mathscr{A}_\Phi \times \mathscr{A}_\Gamma & \underset{F_{\Phi\oplus\Gamma}}{\overset{\langle \kappa_1^*, \kappa_2^*\rangle}{\underset{\top}{\leftarrow}}} & \mathscr{A}_{\Phi\oplus\Gamma} \\ \downarrow{\scriptstyle t^* \times id} & & \downarrow{\scriptstyle (t\oplus id)^*} \\ \mathscr{A}_\Sigma \times \mathscr{A}_\Gamma & \underset{F_{\Sigma\oplus\Gamma}}{\overset{\langle \kappa_1^*, \kappa_2^*\rangle}{\underset{\top}{\leftarrow}}} & \mathscr{A}_{\Sigma\oplus\Gamma} \end{array}$$

By the naturality of κ, we have strict commutativity:

$$id : (t^* \times id) \circ \langle \kappa_1^*, \kappa_2^*\rangle = \langle t^* \kappa_1^*, \kappa_2^*\rangle = \langle \kappa_1^*, \kappa_2^*\rangle \circ (t \oplus id)^*,$$

which determines a canonical natural transformation

$$\tau : F_{\Sigma\oplus\Gamma} \circ (t^* \times id) \to (t \oplus id)^* \circ F_{\Phi\oplus\Gamma}$$

called the *mate* (nLab, 2024) or the *conjugate* of *id* (Mac Lane, 1998, §IX.7). Namely, τ is the transpose along the adjunction $F_{\Sigma\oplus\Gamma} \dashv \langle \kappa_1^*, \kappa_2^*\rangle$ of

$$(t^* \times id) \xrightarrow{\eta} \langle \kappa_1^*, \kappa_2^*\rangle \circ F_{\Sigma\oplus\Gamma} \circ (t^* \times id) \xrightarrow{id} \langle \kappa_1^*, \kappa_2^*\rangle \circ (t \oplus id)^* \circ F_{\Phi\oplus\Gamma}.$$

We define the 2-cell $M^B_t := (\tau \circ \langle \mathrm{Id}, \mathrm{K}_B\rangle) : M^B_\Sigma \circ t^* \to (t \oplus \Gamma)^* \circ M^B_\Phi$.

Now we define an updater $u : \mathrm{Id} \to M^B$ for M^B. For every $A \in \mathscr{A}$, letting $\Sigma := PA$, we define u_A to be the composite

$$A \xrightarrow{\pi_1 \eta_{A,B}} \kappa_1^* F_{\Sigma\oplus\Gamma}\langle A, B\rangle \xrightarrow{\overline{\kappa_1}} M^B A$$

where $\eta_{A,B} : \langle A, B\rangle \to \langle \kappa_1^*, \kappa_2^*\rangle(F_{\Sigma\oplus\Gamma}\langle A, B\rangle)$ is the unit of the adjunction, $\overline{\kappa_1}$ is the cartesian morphism over κ_1. The naturality of u is essentially a consequence of the naturality of η.

We have defined a model transformer M^B with an updater u for every $B \in \mathscr{A}_\Gamma$, and what remains is to define a universal arrow $e : B \to \mathrm{U}_{\oplus\Gamma}\langle M^B, u\rangle$. By Definition 8.3∗3, $\mathrm{U}_{\oplus\Gamma}\langle M^B, u\rangle$ is $\kappa_2^*(F_{0\oplus\Gamma}\langle I, B\rangle) \in \mathscr{A}_\Gamma$. Therefore we define $e : B \to \mathrm{U}_{\oplus\Gamma}\langle M^B, u\rangle$ to be the second projection of the unit

$$\eta_{I,B} : \langle I, B\rangle \to \langle \kappa_1^*, \kappa_2^*\rangle(F_{0\oplus\Gamma}\langle I, B\rangle).$$

To show the universality of e, given any model transformer $\langle N, v\rangle \in \textsc{Mod}_{\textsc{Tr}u}(- \oplus \Gamma)$ with a morphism $f : B \to \mathrm{U}_{\oplus\Gamma}\langle N, v\rangle$ in $\mathscr{A}_\Gamma$, we need to show that there is a unique $\sigma : \langle M^B, u\rangle \to \langle N, v\rangle$ in $\textsc{Mod}_{\textsc{Tr}}(- \oplus \Gamma)$ such that $(\mathrm{U}_{\oplus\Gamma}\sigma) \cdot e = f$.

First of all, the codomain of f is by definition $\kappa_2^*(NI)$, where I is the initial object of $\mathscr{A}_0$. It is not hard to see that the object I is also the initial object of the total category $\mathscr{A}$, so for every $A \in \mathscr{A}$, we have a morphism

$$\bar{f}_A \coloneqq ((\kappa_2^* N!) \cdot f) : B \to \kappa_2^* NA.$$

Now recall that a morphism σ in $\textsc{Mod}_{\textsc{Tr}u}(- \oplus \Gamma)$ is a vertical natural transformation $M^B \to N$ that commutes with the updaters u and v. For every $A \in \mathscr{A}$, letting $\Sigma \coloneqq PA$, the updater v at A is a morphism $v_A : A \to \kappa_1^*(NA)$. Paired with $\bar{f}_A$, we have a morphism $\langle v_A, \bar{f}_A\rangle : \langle A, B\rangle \to \langle \kappa_1^*, \kappa_2^*\rangle(NA)$, and we define $\sigma_A : M^B A = F_{\Sigma\oplus\Gamma}\langle A, B\rangle \to NA$ to be the transpose of $\langle v_A, \bar{f}_A\rangle$ along the adjunction $F_{\Sigma\oplus\Gamma} \dashv \langle \kappa_1^*, \kappa_2^*\rangle$. We omit the verification of naturality here.

What remains is to show that σ defined above is the unique morphism $\langle M^B, u\rangle \to \langle N, v\rangle$ satisfying $(\mathrm{U}_{\oplus\Gamma}\sigma) \cdot e = f$. First of all, σ satisfies this equation since by definition $\mathrm{U}_{\oplus\Gamma}\sigma = \kappa_2^*\sigma_I$ and σ_I makes the following triangle commute

$$\begin{array}{ccc} \langle I, B\rangle & \xrightarrow{\eta_{I,B}} & \langle \kappa_1^*, \kappa_2^*\rangle F_{I\oplus\Gamma}\langle I, B\rangle \\ & \searrow^{\langle v_I, \bar{f}_I\rangle} & \downarrow{\scriptstyle \langle \kappa_1^*, \kappa_2^*\rangle \sigma_I} \\ & & \langle \kappa_1^*, \kappa_2^*\rangle NI \end{array}$$

in the category $\mathscr{A}_0 \times \mathscr{A}_\Gamma$. The second projection of this commutativity diagram is exactly $(\mathrm{U}_{\oplus\Gamma}\sigma) \cdot e = f$. For the uniqueness of σ, given another τ satisfying $(\mathrm{U}_{\oplus\tau}\sigma) \cdot e = f$, for every $\Sigma \in \mathscr{T}$ and $A \in \mathscr{A}_\Sigma$, the naturality of τ for the unique morphism $! : I \to A$ implies the commutativity of the right trapezium below:

$$\begin{array}{ccccc} B & \xrightarrow{e} & \kappa_2^* F_{I\oplus\Gamma}\langle I, B\rangle & \xrightarrow{\kappa_2^* M^B !} & \kappa_2^* F_{\Sigma\oplus\Gamma}\langle A, B\rangle \\ & \searrow^{f} & \downarrow{\scriptstyle \kappa_2^*\tau_I} & & \downarrow{\scriptstyle \kappa_2^*\tau_A} \\ & & \kappa_2^* NI & \xrightarrow{\kappa_2^* N!} & \kappa_2^* NA \\ & \overset{\bar{f}_A}{\searrow} & & \nearrow & \end{array} \tag{8.3}$$

Moreover, it can be shown that the following diagram in $\mathscr{A} \times \mathscr{A}$ commutes by expanding out the definition of the action of $M^B : \mathscr{A} \to \mathscr{A}$ on morphisms:

$$\begin{array}{ccc} \langle I, B\rangle & \xrightarrow{\eta_{I,B}} & \langle \kappa_1^*, \kappa_2^*\rangle F_{0\oplus\Gamma}\langle I, B\rangle \\ {\scriptstyle\langle !, B\rangle}\downarrow & & \downarrow{\scriptstyle\langle \kappa_1^*, \kappa_2^*\rangle M^B !} \\ \langle A, B\rangle & \xrightarrow[\eta_{A,B}]{} & \langle \kappa_1^*, \kappa_2^*\rangle F_{\Sigma\oplus\Gamma}\langle A, B\rangle \end{array}$$

Applying the second projection to this commutative diagram,

$$\pi_2\eta_{A,B} = \pi_2(\eta_{A,B} \cdot \langle !, B\rangle) = \pi_2(\langle \kappa_1^*, \kappa_2^*\rangle M^B ! \cdot \eta_{I,B}) = \kappa_2^* M^B ! \cdot \pi_2\eta_{I,B}.$$

Recall that e is exactly $\pi_2\eta_{I,B}$, so the top horizontal path of the diagram (8.3) is equal to $\pi_2\eta_{A,B}$. Then the diagram (8.3) implies that $\kappa_2^*\tau_A \cdot \pi_2\eta_{A,B} = \overline{f}_A$. This, together with the fact τ is compatible with the updaters,

$$\begin{array}{ccc} A & \xrightarrow{u_A} & \kappa_1^* F_{\Sigma\oplus\Gamma}\langle A, B\rangle \\ & {\scriptstyle v_A}\searrow & \downarrow{\scriptstyle \kappa_1^*\tau_A} \\ & & \kappa_1^* NA \end{array}$$

implies that τ_I is the transpose of $\langle v_A, \bar{f}_A\rangle$ along $F_{\Sigma\oplus\Gamma} \dashv \langle \kappa_1^*, \kappa_2^*\rangle$, so $\tau_I = \sigma_I$. □

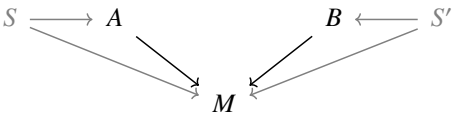
8.3∗5 Example. Let $\mathscr{C}$ be a cocomplete category and $\mathscr{A}$ be the freeness condition (3.1∗21) containing all pairs of endofunctors that preserve colimits of α-chains for some limit ordinal α. By Theorem 3.1∗12 and Theorem 3.2∗14, the fibration $P : \textsc{Alg}_{\mathscr{A}}(\mathscr{C}) \to \textsc{Eqs}_{\mathscr{A}}(\mathscr{C})$ of algebras and equational systems in $\mathscr{A}$ then satisfies the assumptions of Theorem 8.3∗4. For every $\dot{\Psi} \in \textsc{Eqs}_{\mathscr{A}}(\mathscr{C})$, instantiating $\oplus$ to be $+ : \textsc{Eqs}_{\mathscr{A}}(\mathscr{C}) \times \textsc{Eqs}_{\mathscr{A}}(\mathscr{C}) \to \textsc{Eqs}_{\mathscr{A}}(\mathscr{C})$, Theorem 8.3∗4 constructs a modular model F_B of $\dot{\Psi}$ from an ordinary model B of $\dot{\Psi}$.

8.3∗6 Example. Similarly, let $\mathcal{F}$ and $\mathscr{E}$ be the operation family and monoidal category in Example 8.2∗4. Theorem 8.3∗4 lets us construct modular models of theories of monoids with operations from ordinary models.

In particular, if $\mathscr{E} = \langle \textsc{Endo}_\kappa(\mathscr{C}), \circ, \mathrm{Id}\rangle$ for some lκp $\mathscr{C}$, and $\ddot{\Psi} \in \mathcal{F}$ be the theory of monads with some scoped operations, then Theorem 8.3∗4 lets us construct a modular model F_B of $\ddot{\Psi}\mathcal{F}$ from a monad B equipped with a $\ddot{\Psi}$-operation. For any theory $\ddot{\Sigma} \in \mathcal{F}$ of monads with operations, every monad A equipped with a $\ddot{\Sigma}$-operation is sent by the modular model F_B to a new monad C with monad morphisms $A \to C$ and $B \to C$ that are respectively a $\ddot{\Sigma}$-homomorphism and a $\ddot{\Psi}$-homomorphism.

8.3∗7 Example. Let $\mathscr{E}$ be a monoidal category, $\mathcal{F}$ be the operation family $\textsc{Alg}(\mathscr{E})$ of algebraic operations on $\mathscr{E}$-monoids, and $\ddot{\Psi} \in \mathcal{F}$. If the fibration $P : \mathcal{F}\text{-}\textsc{Alg} \to \mathcal{F}$ satisfies the assumption of Theorem 8.3∗4, the free modular model F_B of $\ddot{\Psi}$ over some $B \in \ddot{\Psi}\text{-}\textsc{Alg}$ has a simple characterisation – for every $\ddot{\Sigma} \in \textsc{Alg}(\mathscr{E})$ and $A \in \ddot{\Sigma}\text{-}\textsc{Alg}$, F_B simply maps A to the coproduct of B and A treated as monoids in $\mathscr{E}$. This is because an algebraic operation $S \circ M \to M$ on a monoid M is equivalently a morphism $S \to M$ (Lemma 5∗13), so the initial monoid with both $\ddot{\Sigma}$ and $\ddot{\Psi}$ operations together with a $\ddot{\Sigma}$-homomorphism from A and a $\ddot{\Psi}$-homomorphism from B is the same thing as the initial monoid with monoid morphisms

from A and B, i.e. the coproduct of A and B as monoids:



8.4 Modular Models from Monoid Transformers

8.4∗1. The concept of modular models is directly inspired by Moggi's *monad transformers* and their generalisation, *monoid transformers* (Jaskelioff and Moggi, 2010), to monoids in monoidal categories. A monoid transformer maps every monoid M in a monoidal category $\mathscr{E}$ to a monoid TM together with a monoid morphism $i : M \to TM$. The central question about monoid transformers is:

If there is an operation on the monoid M, can this operation be transformed to an operation on TM too?

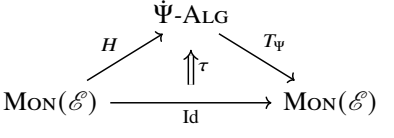
The standard terminology here is to *lift* the operation to TM rather than to *transform* but we use the latter to avoid the confusion with liftings along fibrations.

This question was first formulated by Moggi (1989, §4.1), accompanied by a basic result (Moggi, 1989, Proposition 4.1.3): operations of the form $\alpha : A \to M$ for some fixed endofunctor A can always be transformed to $A \to TM$, namely $i \cdot \alpha : A \to TM$. About 20 years later, Jaskelioff and Moggi (2010) gave a new result: for *functorial monoid transformers* T on a *left-closed* monoidal category, every operation on M of the form $A \,\square\, M \to M$ can be lifted to $A \,\square\, TM \to TM$.

What we have done in this paper is bringing *equations* on operations into the view and formulating transformations of operations as model transformers. In this subsection, we put the old wine by Moggi (1989) and Jaskelioff and Moggi (2010) in our new bottle.

8.4∗2. In this subsection, we fix a monoidal category $\mathscr{E}$ with finite coproducts that is right-distributive: $(\coprod_{i\in S} A_i) \,\square\, B \;\cong\; \coprod_{i\in S}(A_i \,\square\, B)$, which ensures that $\textsc{Alg}(\mathscr{E})$ and $\textsc{Scp}(\mathscr{E})$ from Section 6 are closed under coproducts.

8.4∗3 Theorem. *Let $\mathcal{F}$ be $\textsc{Alg}(\mathscr{E})$ and $\ddot{\Psi} = \langle \dot{\Psi}, T_\Psi \rangle \in \mathcal{F}$. Every functor $H : \textsc{Mon}(\mathscr{E}) \to \dot{\Psi}\text{-}\textsc{Alg}$ together with a natural transformation $\tau : \mathrm{Id} \to T_\Psi \circ H$*



defines a strict modular model M of $\dot{\Psi} \in \mathcal{F}$ making the following commute:

$$\begin{array}{ccc} \mathcal{F}\text{-}\textsc{Alg} & \xrightarrow{\bar{M}} & (\mathcal{F} + \ddot{\Psi})\text{-}\textsc{Alg} \\ \downarrow & & \downarrow \\ \textsc{Mon}(\mathscr{E}) & \xrightarrow[H]{} & \dot{\Psi}\text{-}\textsc{Alg} \end{array} \tag{8.4}$$

where $\bar{M} : \mathcal{F}\text{-Alg} \to (\mathcal{F} + \dot{\Psi})\text{-Alg}$ *is the functor corresponding to* M *by items* 1 *and* 2 *of Theorem* 7.1∗18*, and the unlabelled vertical arrows are the evident projection functors. Moreover,* M *has an updater* $u_{\langle \dot{\Sigma}, T_\Sigma, A, \alpha\rangle} = \tau_{\langle A, T_\Sigma \alpha\rangle}$.

Proof. For every object $\langle \dot{\Sigma}, T_\Sigma, A, \alpha\rangle$ of $\mathcal{F}$-Alg with

$$\dot{\Sigma} = (S \circ -)\text{-Mon} \upharpoonleft (\mathrm{K}_G \vdash L = R),$$

we define $M : \mathcal{F}\text{-Alg} \to \mathcal{F}\text{-Alg}$ to send it to an $(\dot{\Sigma} + \dot{\Psi})$-algebra with the same carrier of $H\langle A, T_\Sigma \alpha\rangle \in \dot{\Psi}\text{-Alg}$. Since $H\langle A, T_\Sigma \alpha\rangle$ already has a $\dot{\Psi}$-algebra, we only need to equip it with an $(S \circ -)$-operation. This can be done by the observation (Jaskelioff and Moggi, 2010, Theorem 3.4) that *algebraic* operations can be transformed along monoid morphisms. Namely, the transformed operation is

$$\alpha^\sharp = [\![s : S, h : H_A \vdash \mu^H(\tau_A(\alpha_S(s, \eta^A)),\ h) : H_A]\!] \tag{8.5}$$

where H_A and τ_A stand for the carrier of $H\langle A,\ T_\Sigma \alpha\rangle$ and $\tau_{\langle A, T_\Sigma \alpha\rangle} : A \to H_A$ respectively, and $\alpha_S : S \circ A \to A$ is the component of α for the algebraic operation on A.

We also need to show that the operation (8.5) satisfies the equation $\mathrm{K}_G \vdash L = R$. This follows from the functoriality of

$$L, R : ((S \circ -) + \Sigma_{\text{Mon}})\text{-Alg} \to G\text{-Alg},$$

which implies that the following diagrams commute

$$\begin{array}{ccc} G & \xrightarrow{L\langle A,\ \alpha\rangle} & A \\ & {\scriptstyle L\langle H_A,\ \alpha^\sharp\rangle}\searrow & \downarrow{\scriptstyle \tau_A} \\ & & H_A \end{array} \qquad \text{and} \qquad \begin{array}{ccc} G & \xrightarrow{R\langle A,\ \alpha\rangle} & A \\ & {\scriptstyle R\langle H_A,\ \alpha^\sharp\rangle}\searrow & \downarrow{\scriptstyle \tau_A} \\ & & H_A \end{array}$$

If α satisfies the equation $L = R$ (i.e. $L\langle A, \alpha\rangle = R\langle A, \alpha\rangle$), so does $\alpha^\sharp$. It can be shown that M is a *strict* modular model of $\dot{\Psi}$ by the same argument for the special case of exception monad transformers in Example 7.1∗8. □

8.4∗4. Example 7.1∗8 is exactly this theorem applied to the exception monad transformer. The *state monad transformer* $A \mapsto (A(S \times -))^S$ for a set S with $|S| < \kappa$ together with its model for the theory St_S of *mutable state* (Example 5∗9) yields a modular model of St_S in $\text{Alg}(\text{End}_\kappa(\mathscr{C}))$. The *list monad transformer* $A \mapsto \mu X.A(1 + (- \times X))$ (Jaskelioff and Moggi, 2010) with its model for *explicit nondeterminism* also gives rise to a modular model.

8.4∗5. Now we move on to scoped operations. First we recall that $\text{Scp}_l(\mathscr{E})$ from 6∗19 is the operation family of scoped operations on monoids and *transliterations*. Let us again start with a concrete example.

8.4∗6 Example. The theory Ec of *exception throwing and catching* in Example 5∗15 is in the family $\text{Scp}_l(\mathscr{E})$ for $\mathscr{E} = \langle \text{End}_\kappa(\text{Set}), \circ, \text{Id}\rangle$. A strict modular model for it can be constructed by extending the modular model of throwing in Example 7.1∗8 with (1) a model of catching on $C_A = A \circ (1 + \text{Id})$ and (2) a way to transform scoped operations on A to C_A.

For (1), we define $\mathit{catch} : (C_A \times C_A) \circ C_A \to C_A$ by $\mathit{catch} := \mu^C \cdot ((c \cdot s) \circ C_A)$ where $\mu^C : C_A \circ C_A \to C_A$ is the multiplication on C_A defined in Example 7.1∗8, and s is the following morphism in which the unlabelled arrow is the canonical strength for the functor

$A \in \textsc{Endo}_\kappa(\textsc{Set})$:

$$s : C_A \times C_A = (A \circ (1 + \mathrm{Id})) \times C_A \to A \circ ((1 + \mathrm{Id}) \times C_A) \cong A \circ (C_A + \mathrm{Id} \times C_A),$$

and lastly the morphism $c : A \circ (C_A + \mathrm{Id} \times C_A) \to C_A$ is denoted by

$$a : A,\ b : (C_A + \mathrm{Id} \times C_A) \vdash \mu^C \Big((a, \iota_2\, *),\\ \mathit{case}\ b\ \mathit{of}\ \{\iota_1\, h \mapsto h;\ \iota_2\, ih \mapsto \eta^C\ (\pi_1\, ih)\} \Big) : C_A$$

The operational idea for this term is that the computation a is first executed and b is its result. The case $b = \iota_1\ h$ means that an exception is thrown and h is the exception handler, so h is executed in this case. On the other hand, the case $b = \iota_2\ ih$ means a normal termination, and the handler is ignored by $\pi_1\ ih$.

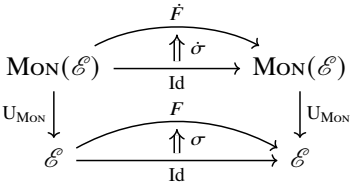
For (2), to transform a scoped operation $\alpha : S \circ A \circ A \to A$ on A to C_A, we define $\alpha^\sharp : S \circ C_A \circ C_A \to C_A$ by the term

$$s : S,\ a : A,\ m : 1 + \mathrm{Id},\ k : C_A \vdash \mu^C((\alpha(s, a, \eta^A), m),\ k) : C_A.$$

The transformed operation $\alpha^\sharp$ satisfies any constant equation $\mathrm{K}_C \vdash L = R$ whenever α does by the same argument as in the proof of Theorem 8.4∗3.

8.4∗7. Theorem 8.4∗3 constructs modular models for algebraic operations from monoid transformers. Jaskelioff and Moggi (2010) shows that this is also possible for scoped operations, provided that the monoid transformer is *functorial*.

8.4∗8 Definition (Jaskelioff and Moggi (2010)). A *functorial monoid transformer* on a monoidal category $\mathscr{E}$ consists of two functors $\dot{F} : \textsc{Mon}(\mathscr{E}) \to \textsc{Mon}(\mathscr{E})$ and $F : \mathscr{E} \to \mathscr{E}$ and two natural transformations $\dot{\sigma} : \mathrm{Id} \to \dot{F}$ and $\sigma : \mathrm{Id} \to F$:



such that $\mathrm{U}_{\textsc{Mon}} \circ \dot{\sigma} = \sigma \circ \mathrm{U}_{\textsc{Mon}}$

8.4∗9. As shown by Jaskelioff and Moggi (2010), many monad transformers in programming languages are functorial, including the exception monad transformer $M(E + -)$ for monads M, the state monad transformer $S \Rightarrow M(S \times -)$, the writer monad transformer $M(W \times -)$, and the (generalised) resumption monad transformer $\mu X.\, M(\Sigma X + -)$ (Cenciarelli and Moggi, 1993). However, the seemingly functorial list transformer $L_M := \mu X.\, M(1 + (- \times X))$ is in fact not functorial, because the associated natural transformation $\dot{\sigma} : M \to L_M$ defined by $a : M \vdash (a, \iota_2\, \langle *, (\eta^M, \iota_1\, \langle\rangle)\rangle) : M \circ (1 + \mathrm{Id} \times L_M)$ refers to the unit η^M of M, which is not a part of the underlying functor of M.

8.4∗10 Theorem. *Assume that the monoidal category $\mathscr{E}$ is left-closed. Let $\ddot{\Psi}$ be some $\langle \dot{\Psi}, T_\Psi \rangle \in \textsc{Scp}_l(\mathscr{E})$. A functorial monoid transformer $\langle \dot{F}, F, \dot{\sigma}, \sigma \rangle$ and a functor $H : \textsc{Mon}(\mathscr{E}) \to \dot{\Psi}\text{-}\textsc{Alg}$ such that $\dot{F} = T_\Psi \cdot H$ induce a strict modular model M of $\ddot{\Psi} \in \textsc{Scp}_l(\mathscr{E})$ with an updater $u_{\langle \dot{\Sigma}, T_\Sigma, A, \alpha \rangle} = \sigma_{\langle A,\ T_\Sigma \alpha \rangle}$.*

Proof sketch. Compared to Theorem 8.4∗3, what is new is how scoped operations $\alpha : S \,\square\, A \,\square\, A \to A$ on a monoid $\langle A, \eta^A, \mu^A \rangle$ are transformed to FA. Such a transformation (not necessarily the unique one) is given by Jaskelioff and Moggi (2010), whose insight is that that scoped operations on A are the same as an algebraic operation on the monoid A/A, which embed A by Cayley's theorem (Example 4.2∗4), and we already know how to transform algebraic operations along monoid transformers. This is why we need (left) closedness in the assumption. We briefly record the transformation below and refer the reader to Jaskelioff and Moggi (2010, §5.1) for more details.

Firstly, recall that we have the Cayley embedding $e : A \to A/A$ and its retraction $r : A/A \to A$ defined as follows:

$$e = (a : A \vdash \lambda x.\, \mu^A(a, x) : A/A) \qquad\qquad r = (f : A/A \vdash f\, \eta^A : A).$$

We can similarly transpose the scoped operation $\alpha : S \,\square\, A \,\square\, A \to A$ on A to obtain a morphism $\tilde{\alpha} : S \to A/A$:

$$\tilde{\alpha} = (s : S \vdash \lambda x.\, \alpha(s, x, \eta^A) : A/A)$$

The transformed operation $\alpha^\sharp : S \,\square\, FA \,\square\, FA \to FA$ is then defined by

$$s : S, a : FA, b : FA \vdash \mu^{FA} \left(Fr \left(\mu^{F(A/A)} \left(\sigma_{A/A} \left(\tilde{\alpha}\ s \right), Fe\ a \right) \right), b \right) : FA$$

This transformed operation $\alpha^\sharp$ preserves any constant equation $\mathrm{K}_C \vdash L = R$ satisfied by α by the same argument for Theorem 8.4∗3. Moreover, the definition of $\alpha^\sharp$ is natural w.r.t. $\textsc{Scp}_l(\mathscr{E})$: given any transliteration $f : S' \to S$ between scoped operations, we have

$$\alpha^\sharp \cdot (f \,\square\, FA \,\square\, FA) = (\alpha \cdot (f \,\square\, A \,\square\, A))^\sharp,$$

which can be directly checked or deduced from the general fact that all the term formers of monoidal algebraic theories are natural, similar to the abstraction theorem of simply typed lambda calculus (Reynolds, 1983). □

8.4∗11 Remark. The theorem above needs the monoid transformer $\dot{F}$ to be over some $F : \mathscr{E} \to \mathscr{E}$ because the retract $r : A/A \to A$ of the Cayley embedding (Example 4.2∗4) used in the proof is not a monoid morphism, so we need $F : \mathscr{E} \to \mathscr{E}$ to have $Fr : F(A/A) \to A$. The requirement of having $\sigma : \mathrm{Id} \to F$ below $\dot{\sigma} : \mathrm{Id} \to \dot{F}$ is also essential. It is needed for showing that the updater is an algebra-homomorphism (Jaskelioff and Moggi, 2010, Lemma 5.3) and the equations are preserved, which are omitted in the proof sketch above.

8.4∗12. A mistake in the earlier paper (Yang and Wu, 2023) by the author is that the strict modular models from Theorem 8.4∗10 and Example 8.4∗6 were claimed to be w.r.t. the family $\textsc{Scp}(\mathscr{E})$ rather than $\textsc{Scp}_l(\mathscr{E})$. This is wrong because the operation transformation $\alpha^\sharp : S \,\square\, FA \,\square\, FA \to FA$ from a scoped operation $\alpha : S \,\square\, A \,\square\, A \to A$ in these modular models is not natural with respect to translations in $\textsc{Scp}(\mathscr{E})$, i.e. there exist translations T such that $(T\alpha)^\sharp \neq T(\alpha^\sharp)$.

For a counterexample, consider the modular model of exception catching in from Example 8.4∗6. Let $\ddot{\Sigma} \in \textsc{Scp}(\textsc{Endo}_\kappa(\textsc{Set}))$ be the theory of monads with a binary scoped operation $b : (\mathrm{Id} \times \mathrm{Id}) \circ A \circ A \to A$, and let $\ddot{\Sigma}' \in \textsc{Scp}(\textsc{Endo}_\kappa(\textsc{Set}))$ be an arbitrary theory. In the category Scp (but not in $\textsc{Scp}_l$), we have a translation $T : \ddot{\Sigma} \to \ddot{\Sigma}'$ that as a functor

sends every $\langle A, \mu^A, \eta^A, \alpha\rangle \in \ddot{\Sigma}'$-Alg to the $\ddot{\Sigma}$-algebra $\langle A, \mu^A, \eta^A, \mu^A \cdot (c \circ A)\rangle$ where c is

$$(\mathrm{Id} \times \mathrm{Id}) \circ A = A \times A \xrightarrow{s} A \circ (\mathrm{Id} \times A) \xrightarrow{A \circ \pi_2} A \circ A \xrightarrow{\mu^A} A$$

and the arrow s is the canonical strength $s_n : An \times An \to A(n \times An)$ for the set-endofunctor A. Note that this translation completely ignores the original operation α. It is perhaps more intuitive to use the syntax of an ordinary programming language here, say Haskell, which would be the following:

$$\begin{aligned} &T\alpha :: A\ x \to A\ x \to (x \to A\ y) \to A\ y \\ &T\alpha\ m\ n\ k = \mathbf{do}\ \_ \leftarrow m; x \leftarrow n; k\ x \end{aligned}$$

In prose, T translates the binary scoped operation $b(x, y)$ to the computation that first runs x, ignores its result, and then runs y.

The operation lifting $\alpha^\sharp$ in Example 8.4∗6 for a binary scoped operation $\alpha :: A\ x \to A\ x \to (x \to A\ y) \to A\ y)$ would be the following in Haskell:

$$\begin{aligned} &\mathbf{data}\ \mathit{Maybe}\ x \quad\ = \mathit{Nothing} \mid \mathit{Just}\ x \\ &\mathbf{data}\ \mathit{MaybeT}\ A\ x = \mathit{MaybeT}\ (A\ (\mathit{Maybe}\ x)) \\ &\alpha^\sharp :: \mathit{MaybeT}\ A\ x \to \mathit{MaybeT}\ A\ x \to (x \to \mathit{MaybeT}\ A\ y) \to \mathit{MaybeT}\ A\ y \\ &\alpha^\sharp\ (\mathit{MaybeT}\ m')\ (\mathit{MaybeT}\ n')\ k = \mathbf{do}\ x \leftarrow \mathit{MaybeT}\ (\alpha\ m'\ n'\ \mathit{return}); k\ x \end{aligned}$$

Now we can see that the two binary scoped operation $(T\alpha)^\sharp$ and $T(\alpha^\sharp)$ are not equal. The operation $T(\alpha^\sharp)$ written in Haskell would be

$$\begin{aligned} &b_1 :: \mathit{MaybeT}\ A\ x \to \mathit{MaybeT}\ A\ x \to (x \to \mathit{MaybeT}\ A\ y) \to \mathit{MaybeT}\ A\ y \\ &b_1\ m\ n\ k = \mathbf{do}\ \_ \leftarrow m; x \leftarrow n; k\ x \end{aligned}$$

while the operation $(T\alpha)^\sharp$ would be

$$b_2\ (\mathit{MaybeT}\ m')\ (\mathit{MaybeT}\ n')\ k = \mathbf{do}\ x \leftarrow \mathit{MaybeT}\ (\mathbf{do}\ \_ \leftarrow m'; n'); k\ x$$

The difference is that when m throws an exception, i.e. when m' returns *Nothing*, b_1 will stop after m, whereas b_2 will continue as n'.

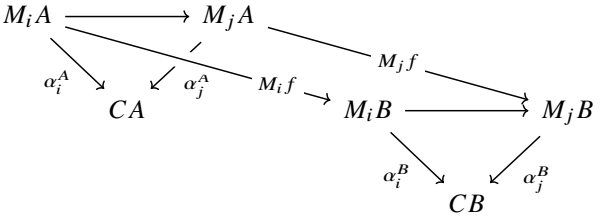
8.5 Colimits and Limits of Model Transformers

8.5∗1. The category of model transformers (i.e. liftings along fibrations) inherits many properties of categories of ordinary models. As a first step, in the following we show colimits and *reindexing-stable limits* of ordinary models can be lifted to model transformers.

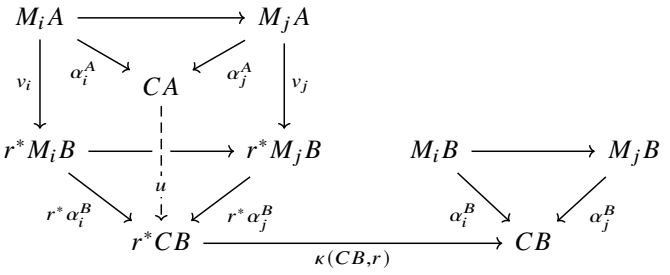
8.5∗2 Theorem. *Let $P : \mathscr{A} \to \mathscr{T}$ be a fibration and $P' : \mathscr{A}' \to \mathscr{T}'$ be a fibration with a cleavage κ, and let $T : \mathscr{T} \to \mathscr{T}'$ be a functor and $\mathscr{D}$ be a category. If every fiber category $\mathscr{A}'_\Sigma$ of P' has (chosen) $\mathscr{D}$-indexed colimits, the category* Motr(T) *of model transformers of T also has $\mathscr{D}$-indexed colimits, which are computed fiberwise.*

Moreover, if reindexing functors of P' preserve (or strictly preserve) $\mathscr{D}$-indexed colimits, the full subcategory of Motr(T) *containing strong (or strict) model transformers is closed under $\mathscr{D}$-indexed colimits in* Motr(T).

Proof. Let $M : \mathscr{D} \to \textsc{Mtr}(T)$ be a $\mathscr{D}$-diagram of model transformers. We define a functor $C : \mathscr{A} \to \mathscr{A}'$ that sends every object $A \in \mathscr{A}$ to the colimit of $M_i A$ in the fiber category $\mathscr{A}_{TPA}$. For every morphism $f : A \to B$ in $\mathscr{A}$, let $\alpha_i^A : M_i A \to CA$ and $\alpha_i^B : M_i B \to CB$ be the colimiting cocones:



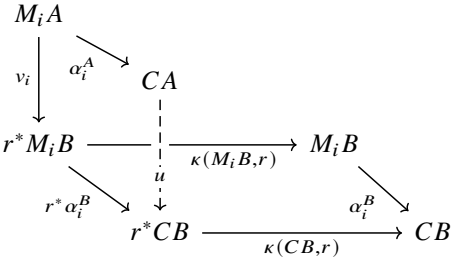
Writing $r := TPf$, for every $i \in \mathscr{D}$, the morphism $M_i f : M_i A \to M_i B$ factors as a vertical morphism $v_i : M_i A \to r^*(M_i B)$ in $\mathscr{A}'_{TPA}$ followed by a cartesian morphism. The reindexing functor r^* sends the cocone α_i^B in $\mathscr{A}'_{TPB}$ to a cocone $r^*\alpha_i^B : r^* M_i B \to r^* CB$ in $\mathscr{A}'_{TPA}$:



(8.6)

The composite $(r^*\alpha_i^B) \cdot v_i : M_i A \to r^* CB$ can be checked to be a cocone too. By the universal property of CA as a colimit of $M_i A$, we have a unique vertical morphism $u : CA \to r^* CB$ such that $u \cdot \alpha_i^A = r^* \alpha^B \cdot v_i$. We define the action of C on the morphism $f : A \to B$ to be $\kappa(CB, r) \cdot u : CA \to CB$. The functoriality of C is a consequence of the functoriality of M_i and the universal property of CA as colimits. For example, if $f : A \to B$ above is $id_A : A \to A$, it can be checked by diagram chasing that for all $i \in \mathscr{D}$, $Cid_A \cdot \alpha_i^A = \alpha_i^A \cdot M_i id_A = \alpha_i^A$, so $Cid_A = id_A$. The case for $C(g \cdot f) = Cg \cdot Cf$ is more complex but similar.

The functor C is by construction a lifting of T. For each i, we show that the family of morphisms $\alpha_i^A : M_i A \to CA$ is natural in A. In the following diagram,



we have $Cf \cdot \alpha_i^A = \kappa(CB, r) \cdot u \cdot \alpha_i^A = \kappa(CB, r) \cdot r^*(\alpha_i^B) \cdot v_i$. The morphism $r^*\alpha_i^B$, which is the image of α_i^B under reindexing r^*, is by definition the unique morphism making the square at the bottom commute, so we have $\kappa(CB, r) \cdot r^*(\alpha_i^B) \cdot v_i = \alpha_i^B \cdot \kappa(M_i B, r) \cdot v_i = \alpha_i^B \cdot M_i f$.

Hence we have shown the required naturality: $Cf \cdot \alpha_i^A = \alpha_i^B \cdot M_i f$, and thus we have a cocone $\langle \alpha_i \rangle_{i \in \mathscr{D}}$ in $\textsc{Motr}(T)$.

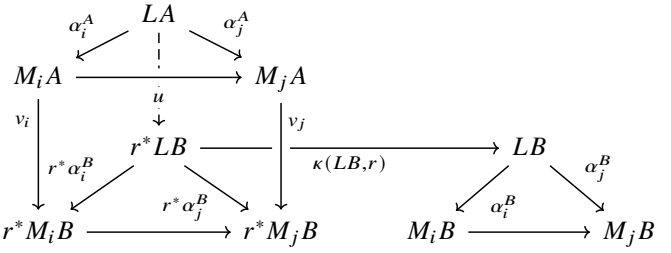
Given any cocone $\langle \beta_i : M_i \to N \rangle_{i \in \mathscr{D}}$, for every $A \in \mathscr{A}$, $\langle \beta_i^A \rangle$ is a cocone in $\mathscr{A}_{TPA}$ from $M_i A$ to NA, so there is a unique mediating morphism $\sigma^A : CA \to NA$ such that $\sigma^A \cdot \alpha_i^A = \beta_i^A$. It can be checked by diagram chasing that the family of morphisms σ^A is natural in A, so C is the colimit of M_i in $\textsc{Motr}(T)$.

Finally, by the construction of the colimit C above, we can see that if reindexing functors of P' (strictly) preserve $\mathscr{D}$-indexed colimits in fiber categories, then when $f : A \to B$ is cartesian and all M_i are strong, the two cocones in the left of (8.6) are isomorphic (the same), therefore C is strong (strict) too. □

8.5∗3. The situation for limits is slightly different: we need reindexing functors to preserve limits in fiber categories for $\textsc{Motr}(T)$ to inherit these limits. This requirement is not too demanding though, since in many fibrations of algebras and theories, reindexing functors are right adjoints so they preserve all limits.

8.5∗4 Theorem. *Let $P : \mathscr{A} \to \mathscr{T}$ be a fibration and $P' : \mathscr{A}' \to \mathscr{T}'$ be a fibration with a cleavage κ, and let $T : \mathscr{T} \to \mathscr{T}'$ be a functor and $\mathscr{D}$ be a category. If every fiber category $\mathscr{A}'_\Sigma$ of P' has (chosen) $\mathscr{D}$-indexed limits, and reindexing functors preserve these limits, then the category $\textsc{Motr}(T)$ has $\mathscr{D}$-indexed limits. Moreover, the subcategory containing strong/strict model transformers are closed under these limits.*

Proof sketch. Similar to the case of colimits above, the limit L of a diagram M_i of model transformers is defined fiberwise: for every object $A \in \mathscr{A}$, LA is defined to be the (chosen) limit of $M_i A$ in the fiber category $\mathscr{A}'_{PTA}$. However, the action of L on a morphism $f : A \to B$ is different from the situation of colimits:



By reindexing the limiting cone $\alpha_i^B : LB \to M_i B$ along $r := TPf$, we have a cone $r^* \alpha_i^B : r^* LB \to r^* M_i B$. Let $v_i : M_i A \to r^* M_i B$ be the unique vertical morphism $\kappa(M_i B) \cdot v_i = M_i$. We have a cone $(v_i \cdot \alpha_i^A) : LA \to r^* M_i B$. Now we use the assumption that r^* preserves $\mathscr{D}$-limits, so $r^* \alpha_i^B : r^* LB \to r^* M_i B$ is still a limiting cone, and we have a vertical morphism $u : LA \to r^* LB$. The rest of this proof is similar to the proof of Theorem 8.5∗2. □

8.5∗5. Under the assumptions of Theorem 8.5∗4 and additionally that P, P' are the same fibration, and $T : \mathscr{T} \to \mathscr{T}$ is equipped with $\eta : \mathrm{Id} \to T$, the category $\textsc{Motr}_u(T)$ of updatable model transformers also has $\mathscr{D}$-indexed limits. In fact, limits in $\textsc{Motr}_u(T)$ are *strictly created* by the forgetful functor $\mathrm{U} : \textsc{Motr}_u(T) \to \textsc{Motr}(T)$, which means that for every diagram $D : \mathscr{D} \to \textsc{Motr}_u(T)$, whenever $\mathrm{U} \circ D$ has a limiting cone $\alpha_i : L \to \mathrm{U} D_i$ in $\textsc{Motr}(T)$, there exists a unique updater u for L making $\alpha_i : \langle L, u \rangle \to D_i$ a limiting cone in $\textsc{Motr}_u(T)$.

To prove this, recall that an updater u for a model transformers M is a natural transformation $u : \mathrm{Id} \to M$ over $\eta : \mathrm{Id} \to T$. For a $\mathscr{D}$-indexed diagram $\langle M_i, u_i \rangle$ in $\textsc{Mod}_{\textsc{Tr}_u}(T)$, for every $A \in \mathscr{A}$, we have a vertical morphism $u_i^A : A \to \eta_A^* M_i A$. Since morphisms in $\textsc{Mod}_{\textsc{Tr}_u}(T)$ are compatible with updaters, u_i^A is a cone over $\eta_A^* M_i A$. In the proof of Theorem 8.5∗4, the limit L of M_i is computed pointwise and fiberwise, so LA is the limit of $M_i A$ in the fiber $\mathscr{A}_{TPA}$. Moreover, the limit LA is preserved by reindexing η_A^*, so $\eta^* LA$ is a limit of $\eta_A^* M_i A$ in $\mathscr{A}_{PA}$, and the cone $u_i^A : A \to \eta_A^* M_i A$ then gives us a unique mediating morphism $u_A : A \to \eta^* LA$. It can be checked that this u is natural and is an updater for L.

Note however, the forgetful functor $\textsc{Mod}_{\textsc{Tr}_u}(T) \to \textsc{Mod}_{\textsc{Tr}}(T)$ does not create colimits: a cone $u_i^A : A \to \eta_A^* M_i A$ does not give us a morphism $A \to \eta_A^* CA$ into the colimit that commutes with $\eta_A^* \alpha_i^A : \eta_A^* M_i A \to \eta_A^* CA$ for all $i \in \mathscr{D}$.

8.5∗6. There are many more properties that we may wish to lift from ordinary models to model transformers. In particular, a question for the future is

If every fiber category is locally κ-presentable, under what conditions the category of model transformers is also locally κ-presentable?

8.6 Composition and Fusion of Model Transformer

8.6∗1. Model transformers are readily composable horizontally. Let M and N be two (strict/strong) model transformers of functors S and T respectively,

$$\begin{array}{ccccc} \mathscr{A} & \xrightarrow{M} & \mathscr{A}' & \xrightarrow{N} & \mathscr{A}'' \\ {\scriptstyle P}\downarrow & & \downarrow{\scriptstyle P'} & & \downarrow{\scriptstyle P''} \\ \mathscr{T} & \xrightarrow[S]{} & \mathscr{T}' & \xrightarrow[T]{} & \mathscr{T}'' \end{array}$$

it is immediate that the composite functor $N \circ M$ is a (strict/strong) model transformers of $T \circ S : \mathscr{T}' \to \mathscr{T}''$. Moreover, when P, P', and P'' are the same fibration, and the functors S and T are pointed, an updater u of M and an updater v of N can be composed horizontally to an updater $v \circ u : \mathrm{Id} \to N \circ M$ as well.

In particular, the composition of a modular model M of $\Sigma \in \mathscr{T}$ (i.e. a model transformer of $- + \Sigma : \mathscr{T} \to \mathscr{T}$) and a modular model N of $\Phi \in \mathscr{T}$ gives us a modular model of $\Sigma + \Phi$ via the isomorphism $- + (\Sigma + \Phi) \cong (- + \Sigma) + \Phi$.

8.6∗2 Example. Let M_E be the modular model of *exception* throwing and catching (Example 8.4∗6), and M_S be the modular model of *mutable state* arising from the state monad transformer by Theorem 8.4∗3. The composite $M_S \circ M_E$ is a modular model of $\textsc{Ec} + \textsc{St}_S$, the theories of exception and mutable state.

8.6∗3. Coproducts of theories are commutative, $\Sigma + \Phi \cong \Phi + \Sigma$, but the composition of modular models is of course not. For example, the opposite order $M_E \circ M_S$ of composing the modular models in Example 8.6∗2 gives rise to another modular model of the coproduct $\textsc{Ec} + \textsc{St}_S$. Both $M_S \circ M_E$ and $M_E \circ M_S$ satisfy the respective equations of exception and mutable state, but they validate different interaction equations: $M_S \circ M_E$ additionally validates commutativity of stateful operations and exception throwing, so the following

program equivalence is validated by $M_S \circ M_E$:

$$\mathit{catch}\ (\mathbf{do}\ \mathit{put}\ s; \mathit{throw})\ h = \mathit{catch}\ (\mathbf{do}\ \mathit{throw}; \mathit{put}\ s)\ h = \mathit{catch}\ \mathit{throw}\ h = h,$$

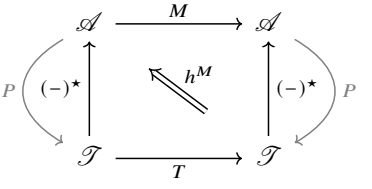
where the second step $\mathit{throw}; \mathit{put}\ s = \mathit{throw}$ is due to the algebraicity of *throw* as a nullary operation. On the other hand, $M_E \circ M_S$ validates

$$\mathit{catch}\ (\mathbf{do}\ \mathit{put}\ s; p)\ h = \mathbf{do}\ \mathit{put}\ s; \mathit{catch}\ p\ h,$$

so $\mathit{catch}\ (\mathbf{do}\ \mathit{put}\ s; \mathit{throw})\ h = \mathbf{do}\ \mathit{put}\ s; \mathit{catch}\ \mathit{throw}\ h = \mathbf{do}\ \mathit{put}\ s; h$. An operational interpretation is that when an exception is caught, $M_S \circ M_E$ rolls back to the state before the *catch*, whereas $M_E \circ M_S$ keeps the state as it is. Both behaviours are desirable depending on the scenario. More discussion about interaction of effects can be found in Yang and Wu (2021).

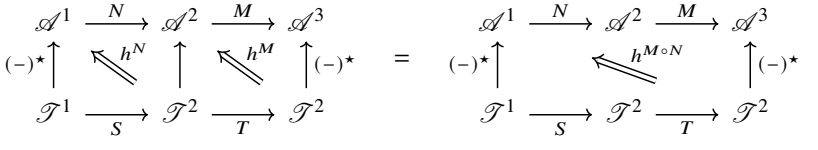
8.6∗4. A straightforward but useful result about composites of model transformers is the *fusion lemma* below: interpreting a term with two model transformers sequentially is equal to interpreting with the composite model transformer. Therefore two consecutive interpretations can be combined into one, eliminating the need to generate the intermediate result that is consumed immediately, a program optimisation known as *short-cut fusion* (Gill et al., 1993; Hinze et al., 2011).

Generalising the natural transformation $h^M : (- + \ddot{\Psi})^\star \to M(-)^\star$ in 7.1∗24, let $P : \mathscr{A} \to \mathscr{T}$ be a fibration with a cleavage such that that all fiber categories have initial objects. We then have a functor $(-)^\star : \mathscr{T} \to \mathscr{A}$ that maps every object $\Sigma \in \mathscr{T}$ to the initial object 0_Σ in the fiber $\mathscr{A}_\Sigma$, and $(-)^\star$ maps every morphism $t : \Sigma \to \Gamma$ to the unique morphism $0_\Sigma \to t^* 0_\Gamma$ followed by the cartesian morphism over t. For every model transformer $M : \mathscr{A} \to \mathscr{A}$ of some functor $T : \mathscr{T} \to \mathscr{T}$, we then have a unique natural transformation $h^M : (T-)^\star \to M(-)^\star$:



that interprets the abstract syntax $(T\Sigma)^\star$ with the model $M\Sigma^\star$ for every $\Sigma \in \mathscr{T}$.

8.6∗5 Lemma (Fusion). For $i \in \{1, 2, 3\}$, let $P^i : \mathscr{A}^i \to \mathscr{T}^i$ be a cloven fibration such that every fiber category has initial objects. Given model transformers $N : \mathscr{A}^1 \to \mathscr{A}^2$ of $S : \mathscr{T}^1 \to \mathscr{T}^2$ and $M : \mathscr{A}^2 \to \mathscr{A}^3$ of $T : \mathscr{T}^2 \to \mathscr{T}^3$, we have



i.e. $h^{M \circ N}_\Sigma = (M h^N_\Sigma) \cdot h^M_{S\Sigma} : (TS\Sigma)^\star \to MN\Sigma^\star$ for every $\Sigma \in \mathscr{T}^1$.

Proof. The component at Σ of these two natural transformations are both the *unique* morphism out of the initial object of the fiber category over $TS\Sigma$. □

8.7 Modular Models in Symmetric Monoidal Categories

8.7∗1. In this subsection, we will have a look at some constructions of modular models that are only possible in symmetric monoidal categories $\mathscr{E}$, such as $\langle \mathscr{C}, \times, 1\rangle$ for cartesian monoids and $\langle \textsc{Endo}_\kappa(\textsc{Set}), *, \mathrm{Id}\rangle$ for applicative functors.

8.7∗2. To begin with, we can upgrade ordinary models of algebraic and scoped operations to modular models by using the fact that in a symmetric $\mathscr{E}$, two monoids $\langle A, \mu^A, \eta^A\rangle$ and $\langle B, \mu^B, \eta^B\rangle$ induces a monoid structure on $A \,\square\, B$.

8.7∗3 Theorem (Independent Combination). *Let $\mathscr{E}$ be a symmetric monoidal category and $\mathcal{F}$ be* $\textsc{Alg}(\mathscr{E})$ *or* $\textsc{Scp}_l(\mathscr{E})$. *For each $\ddot{\Psi} \in \mathcal{F}$, every $\dot{A} \in \ddot{\Psi}$-*Alg *induces a strict modular model M of $\ddot{\Psi} \in \mathcal{F}$ such that $M\langle \ddot{\Sigma}, B, \beta\rangle$ is carried by $A \,\square\, B$, and M has an updater $u_{\ddot{\Sigma},B,\beta} = (B \cong I \,\square\, B \xrightarrow{\eta^A \square B} A \,\square\, B)$.*

Proof. Given a monoid $\langle B, \mu^B, \eta^B\rangle$, $A \,\square\, B$ has the following monoid structure:

$$\eta^{A\square B} = (I \cong I \,\square\, I \xrightarrow{\eta^A \square \eta^B} A \,\square\, B)$$
$$\mu^{A\square B} = \big((A \,\square\, B) \,\square\, (A \,\square\, B) \cong (A \,\square\, A) \,\square\, (B \,\square\, B) \xrightarrow{\mu^A \square \mu^B} A \,\square\, B\big)$$

Moreover, we can transform a scoped operation $\alpha : C \,\square\, A \,\square\, A \to A$ on A to a scoped operation $\alpha^\sharp$ on $A \,\square\, B$ as follows:

$$C \,\square\, (A \,\square\, B) \,\square\, (A \,\square\, B) \cong C \,\square\, (A \,\square\, A) \,\square\, (B \,\square\, B) \xrightarrow{\alpha \square \mu^B} A \,\square\, B$$

Symmetrically, every scoped operation on B can also be transformed to $A \,\square\, B$. Furthermore, algebraic operations are special cases of scoped operations, so they can be transformed in the same way. The preservation of (constant) equations of the operation transformation is the same as the proof of Theorem 8.4∗3. □

8.7∗4. For $\mathscr{E} = \langle \textsc{Endo}_\kappa(\textsc{Set}), *, \mathrm{Id}\rangle$, the intuition for $A * B$ is that two applicative-computations A and B are combined in the way that they execute *independently*, and operations act on $A * B$ pointwise.

There is another way to compose two applicatives, namely $A \circ B$ (McBride and Paterson, 2008). In this way, the B-computation can *depend* on the result of A.

8.7∗5 Theorem (Dependent Combination). *Let $\mathscr{E}$ be $\langle \textsc{Endo}_\kappa(\textsc{Set}), *, \mathrm{Id}\rangle$ and $\mathcal{F}$ be* $\textsc{Alg}(\mathscr{E})$ *or* $\textsc{Scp}_l(\mathscr{E})$. *For each $\ddot{\Psi} \in \mathcal{F}$, every $\dot{A} \in \ddot{\Psi}$-*Alg *induces a strict modular model M of $\ddot{\Psi}$ such that $M_{\ddot{\Sigma}}\langle B, \beta\rangle$ is carried by $A \circ B$, and M has an updater $u_{\ddot{\Sigma},B,\beta} = \eta^A \circ B$.*

Proof sketch. Given two applicative functors $\langle A, \mu^A, \eta^A\rangle$ and $\langle B, \mu^B, \eta^B\rangle$, their composition $A \circ B$ as functors can be equipped with an applicative structure with unit $\eta^{A\circ B} = \eta^A \circ \eta^B$ and the following multiplication $\mu^{A\circ B}$:

$$\begin{aligned}((A \circ B) * (A \circ B))n \cong{}& \textstyle\int^{m,k} A(Bm) \times A(Bk) \times n^{m\times k}\\ \xrightarrow{f}{}& \textstyle\int^{m,k} A(Bm) \times A(Bk) \times (Bn)^{Bm\times Bk}\\ \xrightarrow{g}{}& \textstyle\int^{m',k'} Am' \times Ak' \times (Bn)^{m'\times k'}\\ \cong{}& (A * A)(Bn) \xrightarrow{\mu^A} A(Bn)\end{aligned}$$

where the arrow g is the substitution of the bound variables of the coend $m' = Bm$ and $k' = Bk$; the arrow f uses functoriality of the coend and the morphism $n^{m\times k} \to (Bn)^{Bm\times Bk}$ given by the transpose of the following morphism:

$$n^{m\times k} \times Bm \times Bk \xrightarrow{\iota_{m,k}} \int^{m,k} Bm \times Bk \times n^{m,k} \cong (B * B)n \xrightarrow{\mu^B} Bn.$$

To transform a scoped operation $\alpha : S * A * A \to A$ to $A \circ B$, we use the fact that there is a canonical morphism $s : S * (A \circ B) \to (S * A) \circ B$ as follows:

$$\begin{aligned}(S * (A \circ B))n &\cong \textstyle\int^{m,k} Sm \times A(Bk) \times n^{m\times k}\\ &\to \textstyle\int^{m,k} Sm \times A(Bk) \times (Bn)^{m\times Bk}\\ &\to \textstyle\int^{m,k'} Sm \times Ak' \times (Bn)^{m\times k'}\\ &\cong (S * A)(Bn)\end{aligned}$$

where the first step uses the action of the functor B on morphisms: $n^k \to (Bn)^{(Bk)}$, and the second steps is the substitution of the bound variable $k' = Bk$. We define the transformation of α to $A \circ B$ to be

$$\begin{aligned}S * (A \circ B) * (A \circ B) &\xrightarrow{s*(A\circ B)} ((S * A) \circ B) * (A \circ B)\\ &\xrightarrow{(\overline{\alpha}\circ B)*(A\circ B)} (A \circ B) * (A \circ B) \xrightarrow{\mu} A \circ B\end{aligned}$$

where $\overline{\alpha} = (S * A \xrightarrow{S*A*\eta^A} S * A * A \xrightarrow{\alpha} A)$.

To transform a scoped operation $\beta : G * B * B \to B$ to $A \circ B$, we need the following canonical morphism $t : G * (A \circ B) \to A \circ (G * B)$:

$$\begin{aligned}(G * (A \circ B))n &\cong \textstyle\int^{m,k} Gm \times A(Bk) \times n^{m\times k}\\ &\to \textstyle\int^{m,k} A(Gm \times Bk \times n^{m\times k})\\ &\to \textstyle\int^{m,k} A((G * B)n)\\ &\cong A((G * B)n)\end{aligned}$$

where the first step uses the canonical strength of A to push Gm and $n^{m\times k}$ inwards; the second step uses the coprojection $\iota_{m,k} : Gm \times Bk \times n^{m\times k} \to (G * B)n$. With t we define the transformed scoped operation on $A \circ B$:

$$\begin{aligned}G * (A \circ B) * (A \circ B) &\xrightarrow{t*(A\circ B)} (A \circ (G * B)) * (A \circ B)\\ &\xrightarrow{(A\circ\overline{\beta})*(A\circ B)} (A \circ B) * (A \circ B)\\ &\xrightarrow{\mu^{A\circ B}} A \circ B\end{aligned}$$

where $\overline{\beta} = (G * B \xrightarrow{G*B*\eta^B} G * B * B \xrightarrow{\beta} G)$. □

8.7∗6. To see the difference between Theorem 8.7∗3 and Theorem 8.7∗5, let $\dot{A}$ be the applicative functor induced by the exception monad $\bar{E} + \mathrm{Id}$. It is a model of the applicative version of the theory ET_E of exception *throwing*, equipped with an operation *throw* : $\bar{E} * (\bar{E} + \mathrm{Id}) \to (\bar{E} + \mathrm{Id})$. Using Theorem 8.7∗5, it can be extended to a modular model using $(\bar{E} + \mathrm{Id}) \circ B \cong (\bar{E} + B)$ for all applicatives B. In this model, it holds that for all

elements $x, y \in (\bar{E} + B)X$,

$$throw\ \langle e, x\rangle = \iota_1 e = throw\ \langle e, y\rangle,$$

which means that exception throwing discards any B-computation. But it is not true for the independent composition $(\bar{E} + \mathrm{Id}) * B$.

8.7∗7. Our final example is an interesting modular model for *phased computation*, generalising the construction that Kidney and Wu (2021) and Gibbons et al. (2022) use for *breadth-first search*. The theory $\textsc{Pha} \in \textsc{Scp}(\mathscr{E})$ has a unary scoped operation *later*. The intention is that a program may have multiple phases of execution, and the operation *later p* delays the execution of p to the next phase.

For example, if F is an applicative functor in Haskell with $later :: F\,a \to F\,a$, given $p_i :: F\,a$, the following Haskell program of type $F\,a$

$$\begin{array}{l}\textbf{do}\ later\ (\textbf{do}\ later\ p_3\\ \qquad\qquad\quad p_{21})\\ \quad p_{11}\\ \quad later\qquad p_{22}\\ \quad p_{12}\end{array}$$

is supposed to execute p_{11} and p_{12} at phase 1, p_{22} and p_{21} at phase 2, and p_3 at phase 3. A standard example of such an applicative functor F is the nested list functor $[\,[\,a\,]\,]$, where the i-th element of the outer list contains all possible outcomes of the computation at phase i, and $later\ xs = [\,[\,] : xs\,]$.

In a symmetric closed monoidal category $\mathscr{E}$ such that every object has a free monoid over it, given a monoid $\dot{A} = \langle A, \mu^A, \eta^A\rangle$, Kidney and Wu's [2021] idea can be abstracted as equipping (the carrier of) the free monoid $S_A = \mu X.\ A \,\square\, X + I$ over A with a nonstandard monoid structure $\langle S_A, \mu^{S_A}, \eta^{S_A}\rangle$ with $\eta^{S_A} : I \to \mu X.\ A \,\square\, X + I$ given by $\vdash in\ (\iota_2\ *)$ where $out : (S_A \cong A \,\square\, S_A + I) : in$ is the isomorphism for the initial algebra, and μ^{S_A} is denoted by $s : S_A, t : S_A \vdash m : S_A$ where m is

$$\begin{array}{lll}case\ (out\ s, out\ t)\ of & & \\ \quad (\iota_1\ (a, x), \iota_1\ (a', y)) & \mapsto in\ (\iota_1\ (\mu^A(a, a'), \mu^{S_A}(x, y)) \\ \quad (\iota_2\ *, \quad y) & \mapsto in\ y \\ \quad (x, \quad \iota_2\ *) & \mapsto in\ x\end{array}$$

Note that the use of variable x and a' in the first case does not match their order in the context, so we need a symmetric monoidal category, and we also need closedness for interpreting structural recursion on the initial algebra S_A. This construction is essentially the same idea as the list object in the category of (ordinary) monoids that we saw in 3.1∗16. The intuition is that $S_A = \mu X.A \,\square\, X + I$ is a list of A-computations at each phase, and μ^{S_A} merges two lists by multiplying computations at the same phase. The *later* operation on S_A is defined as

$$p : S_A, k : S_A \vdash \mu^{S_A}\ (in\ (\iota_1\ (\eta^A, p), k)) : S_A.$$

The construction $A \mapsto S_A$ is a functorial monoid transformer, so we can use Theorem 8.4∗10 to obtain a modular model of phasing in $\textsc{Scp}_l(\mathscr{E})$.

8.7∗8. To summarise this section, we have studied modular constructions of algebraic structures in the framework of lifting functors $T : \mathscr{T} \to \mathscr{T}'$ along two fibrations $P : \mathscr{A} \to \mathscr{T}'$ and $P' : \mathscr{A}' \to \mathscr{T}'$. The base categories $\mathscr{T}$ and $\mathscr{T}'$ of the fibrations contain some notion of algebraic theories, and the total categories $\mathscr{A}$ and $\mathscr{A}'$ contain models of all these theories. The functor $T : \mathscr{T} \to \mathscr{T}'$ transforms every theory in a certain way, for example, by combining it with another fixed theory. Liftings of T along P and P' sends an object in every fiber category $\mathscr{A}_\Sigma$ to an object in the fiber $\mathscr{A}'_{T P \Sigma}$, so we call them *model transformers*. We can intuitively think of a functor $T : \mathscr{T} \to \mathscr{T}$ as a theory $T\Sigma$ parameterised by some potential future extension with $\Sigma \in \mathscr{T}$, then a model transformer M of T can be thought of as a model of the 'parameterised theory' T.

We have also seen a handful of universal constructions of model transformers as well as some more concrete constructions, such as using monoid transformers. Lastly, we comment that liftings along fibrations have many other applications in computer science, such as in logical relations for computational types (lifting a computational monad T along a fibration of predicates over sets) (Katsumata, 2005), in Hoare logics (lifting a computation monad T along a fibration of specifications over types) (Aguirre et al., 2022), and in behaviour metrics of states of automata (lifting a coalgebra encoding an automaton along a fibrations of metric spaces over sets) (Baldan et al., 2014). It is an interesting question for the future to find out whether the lifting techniques developed in these contexts give interesting modular models of algebraic theories.

References

Abramsky, S. & Jung, A. (1995) Domain theory. Handbook of Logic in Computer Science. Oxford University PressOxford. p. 1–168.

Adamek, J., Rosicky, J., Vitale, E. M. & Lawvere, F. W. (2010) *Algebraic Theories*. Cambridge University Press. Cambridge.

Adámek, J. (1974) Free algebras and automata realizations in the language of categories. *Commentationes Mathematicae Universitatis Carolinae*. **015**(4), 589–602.

Adámek, J. & Rosicky, J. (1994) *Locally Presentable and Accessible Categories*. London Mathematical Society Lecture Note Series. Cambridge University Press.

Aguirre, A., Katsumata, S.-y. & Kura, S. (2022) Weakest preconditions in fibrations. *Mathematical Structures in Computer Science*. **32**(4), 472–510.

Altenkirch, T., Capriotti, P., Dijkstra, G., Kraus, N. & Nordvall Forsberg, F. (2018) Quotient inductive-inductive types. Foundations of Software Science and Computation Structures. Springer International Publishing. pp. 293–310.

Altenkirch, T. & Kaposi, A. (2016) Type theory in type theory using quotient inductive types. Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. New York, NY, USA. Association for Computing Machinery. p. 18–29.

Arkor, N. (2022) *Monadic and Higher-Order Structure*. Ph.D. thesis. University of Cambridge.

Arkor, N. & Fiore, M. (2020) Algebraic models of simple type theories: A polynomial approach. Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science. New York, NY, USA. Association for Computing Machinery. p. 88–101.

Atkey, R. (2009) Parameterised notions of computation. *Journal of Functional Programming*. **19**(3–4), 335–376.

Awodey, S., Frey, J. & Speight, S. (2018) Impredicative encodings of (higher) inductive types. Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science. Association for Computing Machinery. p. 76–85.

Bach Poulsen, C. & van der Rest, C. (2023) Hefty algebras: Modular elaboration of higher-order algebraic effects. *Proc. ACM Program. Lang.* **7**(POPL).

Bainbridge, E. S., Freyd, P. J., Scedrov, A. & Scott, P. J. (1990) Functorial polymorphism. *Theoretical Computer Science*. **70**(1), 35–64.

Baldan, P., Bonchi, F., Kerstan, H. & König, B. (2014) Behavioral metrics via functor lifting. 34th International Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS 2014). Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. pp. 403–415.

Bauer, A. & Pretnar, M. (2015) Programming with algebraic effects and handlers. *Journal of Logical and Algebraic Methods in Programming*. **84**(1), 108–123.

Borceux, F. (1994) *Handbook of Categorical Algebra: Volume 2, Categories and Structures*. vol. 2. Cambridge University Press.

Borceux, F. (1994) *Handbook of Categorical Algebra: Volume 3, Sheaf Theory*. vol. 3. Cambridge University Press.

Cartmell, J. (1978) *Generalised Algebraic Theories and Contextual Categories*. Ph.D. thesis. University of Oxford.

Cartmell, J. (1986) Generalised algebraic theories and contextual categories. *Annals of Pure and Applied Logic*. **32**, 209–243.

Castellano, L., De Michelis, G. & Pomello, L. (1987) Concurrency vs interleaving: An instructive example. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*. **31**.

Cenciarelli, P. & Moggi, E. (1993) A syntactic approach to modularity in denotational semantics. Technical report. In Proceedings of the Conference on Category Theory and Computer Science.

Cohn, P. (1981) *Universal Algebra*. Mathematics and Its Applications. Springer Dordrecht.

Coquand, T. & Huet, G. (1988) The calculus of constructions. *Information and Computation*. **76**(2), 95–120.

Crole, R. L. (1994) *Categories for Types*. Cambridge University Press.

Day, B. (1970) On closed categories of functors. Reports of the Midwest Category Seminar IV. Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 1–38.

Filinski, A. (1994) Representing monads. Proceedings of the 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. New York, NY, USA. Association for Computing Machinery. p. 446–457.

Filinski, A. (1999) Representing layered monads. Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. New York, NY, USA. Association for Computing Machinery. p. 175–188.

Fiore, M. (2008) Second-order and dependently-sorted abstract syntax. Proceedings of the 2008 23rd Annual IEEE Symposium on Logic in Computer Science. IEEE Computer Society. p. 57–68.

Fiore, M. & Hur, C.-K. (2007) Equational systems and free constructions. Proceedings of the 34th International Conference on Automata, Languages and Programming. Berlin, Heidelberg. Springer-Verlag. p. 607–618.

Fiore, M. & Hur, C.-K. (2009) On the construction of free algebras for equational systems. *Theoretical Computer Science*. **410**(18), 1704–1729.

Fiore, M. & Hur, C.-K. (2010) Second-order equational logic. Computer Science Logic. Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 320–335.

Fiore, M. & Mahmoud, O. (2010) Second-order algebraic theories. Mathematical Foundations of Computer Science 2010. Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 368–380.

Fiore, M. & Mahmoud, O. (2014) Functorial semantics of second-order algebraic theories.

Fiore, M., Pitts, A. & Steenkamp, S. (2022) Quotients, inductive types, & quotient inductive types. *Logical Methods in Computer Science*. **18**(2), 15:1–15:37.

Fiore, M., Plotkin, G. & Turi, D. (1999) Abstract syntax and variable binding. Proceedings. 14th Symposium on Logic in Computer Science. pp. 193–202.

Fiore, M. & Saville, P. (2017) List objects with algebraic structure. 2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017). Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. pp. 16:1–16:18.

Fiore, M. & Staton, S. (2014) Substitution, jumps, and algebraic effects. *Proceedings of the Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic, CSL 2014 and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2014*.

Fiore, M. & Szamozvancev, D. (2022) Formal metatheory of second-order abstract syntax. *Proc. ACM Program. Lang.* **6**(POPL).

Ghani, N. & Uustalu, T. (2004) Coproducts of ideal monads. *RAIRO - Theoretical Informatics and Applications*. **38**(4), 321–342.

Ghani, N., Uustalu, T. & Hamana, M. (2006) Explicit substitutions and higher-order syntax. *Higher-Order and Symbolic Computation*.

Gibbons, J., Kidney, D. O., Schrijvers, T. & Wu, N. (2022) Breadth-first traversal via staging. Mathematics of Program Construction. Cham. Springer International Publishing. pp. 1–33.

Gill, A. & Kmett, E. (2012) `mtl`: Monad classes, using functional dependencies. https://hackage.haskell.org/package/mtl.

Gill, A., Launchbury, J. & Peyton Jones, S. (1993) A short cut to deforestation. Proceedings of the Conference on Functional Programming Languages and Computer Architecture. New York, NY, USA. Association for Computing Machinery. p. 223–232.

Girard, J.-Y. (1986) The System F of variable types, fifteen years later. *Theoretical Computer Science*. **45**, 159–192.

Goguen, J. A., Thatcher, J. W., Wagner, E. G. & Wright, J. B. (1977) Initial algebra semantics and continuous algebras. *Journal of the ACM*. **24**(1), 68–95.

Gratzer, D. (2023) *Syntax and semantics of modal type theory*. Ph.D. thesis. Aarhus University.

Hinze, R. (2012) Kan extensions for program optimisation or: Art and Dan explain an old trick. Mathematics of Program Construction. Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 324–362.

Hinze, R., Harper, T. & James, D. W. H. (2011) Theory and practice of fusion. Implementation and Application of Functional Languages. Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 19–37.

Hughes, J. (1986) A novel representation of lists and its application to the function "reverse". *Inf. Process. Lett.* **22**, 141–144.

Hughes, J. (2000) Generalising monads to arrows. *Science of Computer Programming*. **37**(1), 67–111.

Hyland, J. (1988) A small complete category. *Annals of Pure and Applied Logic*. **40**(2), 135–165.

Hyland, J., Plotkin, G. & Power, J. (2006) Combining effects: Sum and tensor. *Theoretical Computer Science*. **357**(1), 70–99.

Hyland, M. (2017) Classical lambda calculus in modern dress. *Mathematical Structures in Computer Science*. **27**(5), 762–781.

Jacobs, B. (1999) *Categorical Logic and Type Theory*. Number 141 in Studies in Logic and the Foundations of Mathematics. North Holland. Amsterdam.

Jacobs, B., Heunen, C. & Hasuo, I. (2009) Categorical semantics for arrows. *Journal of Functional Programming*. **19**(3-4), 403–438.

Jaskelioff, M. (2009) Modular monad transformers. Programming Languages and Systems. Berlin, Heidelberg. Springer Berlin Heidelberg. pp. 64–79.

Jaskelioff, M. & Moggi, E. (2010) Monad transformers as monoid transformers. *Theoretical Computer Science*. **411**, 4441–4466.

Johnson, N. & Yau, D. (2020) 2-dimensional categories.

Kammar, O., Lindley, S. & Oury, N. (2013) Handlers in action. Proceedings of the 18th ACM SIGPLAN International Conference on Functional Programming. New York, NY, USA. Association for Computing Machinery. p. 145–158.

Katsumata, S.-y. (2005) A semantic formulation of $\top\top$-lifting and logical predicates for computational metalanguage. Computer Science Logic. Springer Berlin Heidelberg. p. 87–102.

Katsumata, S.-y. (2014) Parametric effect monads and semantics of effect systems. Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. New York, NY, USA. Association for Computing Machinery. p. 633–645.

Katsumata, S.-y., McDermott, D., Uustalu, T. & Wu, N. (2022) Flexible presentations of graded monads. *Proc. ACM Program. Lang.* **6**(ICFP).

Kelly, G. & Power, J. (1993) Adjunctions whose counits are coequalizers, and presentations of finitary

enriched monads. *Journal of Pure and Applied Algebra*. **89**(1), 163–179.
Kelly, G. M. (1982) Structures defined by finite limits in the enriched context, i. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*. **23**(1), 3–42.
Kidney, D. O. & Wu, N. (2021) Algebras for weighted search. *Proc. ACM Program. Lang.* **5**(ICFP).
Kiselyov, O. & Ishii, H. (2015) Freer monads, more extensible effects. *SIGPLAN Not.* **50**(12), 94–105.
Kock, A. (1972) Strong functors and monoidal monads. *Archiv der Mathematik*. **23**, 113–120.
Kovács, A. (2023) *Type-theoretic signatures for algebraic theories and inductive types*. Ph.D. thesis. Eötvös Loránd University.
Lambek, J. (1958) The mathematics of sentence structure. *The American Mathematical Monthly*. **65**(3), 154–170.
Lambek, J. (1968) A fixpoint theorem for complete categories. *Mathematische Zeitschrift*. **103**, 151–161.
Lambek, J. & Scott, P. J. (1986) *Introduction to higher order categorical logic*. Cambridge University Press.
Lamiaux, T. & Ahrens, B. (2024) An introduction to different approaches to initial semantics.
Liang, S., Hudak, P. & Jones, M. (1995) Monad transformers and modular interpreters. ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. ACM. pp. 333–343.
Lindley, S., Wadler, P. & Yallop, J. (2011) Idioms are oblivious, arrows are meticulous, monads are promiscuous. *Electronic Notes in Theoretical Computer Science*. **229**(5), 97–117.
Loregian, F. (2021) *(Co)end Calculus*. London Mathematical Society Lecture Note Series. Cambridge University Press.
Luo, Z. (1994) *Computation and reasoning: a type theory for computer science*. Oxford University Press.
Mac Lane, S. (1998) *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer. Berlin. second edition.
MacLane, S. (1965) Categorical algebra. *Bulletin of the American Mathematical Society*. **71**(1), 40 – 106.
Markl, M. (2006) Operads and PROPs.
McBride, C. & Paterson, R. (2008) Applicative programming with effects. *Journal of Functional Programming*. **18**(1), 1–13.
McDermott, D. & Uustalu, T. (2022) Flexibly graded monads and graded algebras. Mathematics of Program Construction. Cham. Springer International Publishing. pp. 102–128.
McDermott, D. & Uustalu, T. (2022) What makes a strong monad? *Electronic Proceedings in Theoretical Computer Science*. **360**, 113–133.
Moggi, E. (1989) An abstract view of programming languages. Technical Report ECS-LFCS-90-113. Edinburgh University, Department of Computer Science.
Moggi, E. (1989) Computational lambda-calculus and monads. Proceedings. Fourth Annual Symposium on Logic in Computer Science. pp. 14–23.
Moggi, E. (1991) Notions of computation and monads. *Information and Computation*. **93**(1), 55–92.
Mokhov, A., Mitchell, N. & Peyton Jones, S. (2018) Build systems à la carte. *Proc. ACM Program. Lang.* **2**(ICFP).
nLab. (2024) Cantor's theorem. https://ncatlab.org/nlab/show/Cantor%27s+theorem. Revision 25.
nLab. (2024) complete small category. https://ncatlab.org/nlab/show/complete+small+category. Revision 16.
nLab. (2024) concept with an attitude. https://ncatlab.org/nlab/show/concept+with+an+attitude. Revision 23.
nLab. (2024) free monad. https://ncatlab.org/nlab/show/free+monad. Revision 20.
nLab. (2024) mate. https://ncatlab.org/nlab/show/mate. Revision 26.
Oosten, J. v. (2008) *Realizability: an introduction to its categorical side*. Studies in logic and the foundations of mathematics. Elsevier. Oxford. first edition.
O'Hearn, P. W. & Pym, D. J. (1999) The logic of bunched implications. *Bulletin of Symbolic Logic*. **5**(2), 215–244.

Paterson, R. (2012) Constructing applicative functors. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. **7342 LNCS**, 300–323.

Phoa, W. (1992) An introduction to fibrations, topos theory, the effective topos and modest sets. Technical Report ECS-LFCS-92-208. University of Edinburgh.

Pieters, R. P., Rivas, E. & Schrijvers, T. (2020) Generalized monoidal effects and handlers. *Journal of Functional Programming*. **30**, e23.

Piróg, M. (2016) Eilenberg–Moore monoids and backtracking monad transformers. *Electronic Proceedings in Theoretical Computer Science*. **207**, 23–56.

Piróg, M., Schrijvers, T., Wu, N. & Jaskelioff, M. (2018) Syntax and semantics for operations with scopes. Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science. Association for Computing Machinery. p. 809–818.

Plotkin, G. & Power, J. (2001) Semantics for algebraic operations. *Electronic Notes in Theoretical Computer Science*. **45**, 332–345.

Plotkin, G. & Power, J. (2002) Notions of computation determine monads. Foundations of Software Science and Computation Structures, 5th International Conference. Springer. pp. 342–356.

Plotkin, G. & Power, J. (2003) Algebraic Operations and Generic Effects. *Applied Categorical Structures*. **11**(1), 69–94.

Plotkin, G. & Power, J. (2004) Computational effects and operations: An overview. *Electronic Notes in Theoretical Computer Science*. **73**, 149–163.

Plotkin, G. & Pretnar, M. (2009) Handlers of algebraic effects. Programming Languages and Systems. Springer Berlin Heidelberg. pp. 80–94.

Plotkin, G. & Pretnar, M. (2013) Handling algebraic effects. *Logical Methods in Computer Science*. **9**(4).

Power, J. (1999) Enriched lawvere theories. *Theory and Applications of Categories*. **6**(7), 83–93.

Reynolds, J. C. (1983) Types, abstraction and parametric polymorphism. IFIP Congress.

Rivas, E. & Jaskelioff, M. (2017) Notions of computation as monoids. *Journal of Functional Programming*. **27**(September).

Robinson, E. (2002) *Variations on algebra: Monadicity and generalisations of equational theories*. **13**(3), 308–326.

Román, M. (2022) Promonads and string diagrams for effectful categories. ACT '22: Applied Category Theory, Glasgow, United Kingdom, 18–22 July, 2022.

Schrijvers, T., Piróg, M., Wu, N. & Jaskelioff, M. (2019) Monad transformers and modular algebraic effects: What binds them together. Proceedings of the 12th ACM SIGPLAN International Symposium on Haskell, Haskell@ICFP 2019, Berlin, Germany, August 18-23, 2019. pp. 98–113.

Spivey, M. (1990) A functional theory of exceptions. *Science of Computer Programming*. **14**(1), 25–42.

Stark, I. (2008) Free-algebra models for the π-calculus. *Theoretical Computer Science*. **390**(2), 248–270. Foundations of Software Science and Computational Structures.

Streicher, T. (2023) Fibered categories a la Jean Benabou.

Swierstra, W. (2008) Data types à la carte. *Journal of Functional Programming*. **18**(4), 423–436.

van den Berg, B. & Schrijvers, T. (2024) A framework for higher-order effects & handlers. *Sci. Comput. Program.* **234**(C).

Wadler, P. (1990) Comprehending monads. Proceedings of the 1990 ACM Conference on LISP and Functional Programming. ACM. pp. 61–78.

Wadler, P. (1995) Monads for functional programming. Advanced Functional Programming, First International Spring School on Advanced Functional Programming Techniques-Tutorial Text. Berlin, Heidelberg. Springer-Verlag. p. 24–52.

Wu, N., Schrijvers, T. & Hinze, R. (2014) Effect handlers in scope. *Proceedings of the 2014 ACM SIGPLAN Symposium on Haskell - Haskell '14*. pp. 1–12.

Yang, Z., Paviotti, M., Wu, N., van den Berg, B. & Schrijvers, T. (2022) Structured handling of scoped effects. Berlin, Heidelberg. Springer-Verlag. p. 462–491.

Yang, Z. & Wu, N. (2021) Reasoning about effect interaction by fusion. *Proc. ACM Program. Lang.*

5(ICFP).

Yang, Z. & Wu, N. (2023) Modular models of monoids with operations. *Proc. ACM Program. Lang.* **7**(ICFP).