Monad Composition via Preservation of Algebras

Louis Parlant

A thesis presented for the degree of

Doctor of Philosophy

Department of Computer Science

University College London

I, Louis Parlant, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Monads are a central object of category theory and constitute crucial tools for many areas of Computer Science, from semantics of computation to functional programming. An important aspect of monads is their correspondence with algebraic theories (their 'presentation'). As demonstrated by the history of this field, composing monads is a challenging task: the literature contains numerous mistakes and features no general method. One categorical construct, named 'distributive law' allows this composition, but its existence is not guaranteed.

This thesis addresses the question of monad composition by presenting a method for the construction of distributive laws. For this purpose, we introduce a notion of preservation of algebraic features: considering an arbitrary algebra for the theory presenting a monad S, we examine whether its structure is preserved when applying another monad T. In the case of success, it allows us to construct a distributive law and to compose our monads into TS.

In order to develop a general framework, we focus on the class of monoidal monads. If T is monoidal, the algebraic operations presenting S are preserved in a canonical fashion; it remains to examine whether the equations presenting S are also preserved by T.

As it turns out, the preservation of an equation depends on the layout of its variables: if each variable appears once on each side, the considered equation is automatically preserved by a monoidal monad. On the other hand, if a variable is duplicated or only appears on one side, preservation is not systematic. The main results of this thesis connect the preservation of such equations with structural properties of monads.

In the case where T does not preserve an equation presenting S, our distributive law cannot be built; we provide a series of methods to slightly modify our monads and overcome this issue, and we investigate some less conventional distributive laws.

Finally, we consider the presentations of both S and T and revisit our construction of distributive laws, this time with an algebraic point of view.

Overall, this thesis presents a general approach to the problem of monad composition by relating categorical properties of monads with preservation of algebras.

Impact Statement

The research work presented in this thesis is a study on the topic of category theory; we address the notoriously difficult problem of monad composition with a fresh approach focusing on the preservation of algebraic features. The impact of our findings is mostly theoretical and academic. We summarise below the main areas to which our work applies.

- On a purely theoretical level, this work answers the long-standing question of monad composition in many cases and offers a general method for this problem.
- The use of monads is widespread for the study of semantics of programming languages. Once again, this work resolves the issue of compositionality of monads modelling computations in diverse cases where our method applies.
- The categorical formalisation of automata, and in particular the process of generalised determinisation, benefits from both our study of preservation of algebraic features and our strategies for non-composing monads. We explicitly show in this thesis how our results relate to probabilistic automata and alternating automata.
- Because of the correspondence between monads and algebraic theories, our findings can prove useful to the study of universal algebra. Each distributive law obtained through our method offers a method to combine two algebraic theories in a consistent way.

Outside the academic community, our work has a potential impact through its applications to programming.

- In the field of functional programming, monads offer a unified framework for a wide range of computational features. It can be crucial to combine several of these programming effects, and our approach gives a canonical approach to the construction of valid distributive laws for this purpose.

Contents

Chapter 1

Introduction

The search for a theoretical system encompassing mathematical objects of a different nature is an everlasting challenge. Category theory, a field started by Saunders Mac Lane and Samuel Eilenberg in the 1950s [25], attempts to answer this question by placing the focus on the idea of a mathematical structure, and on the transformations which preserve it. This approach allows to unite diverse areas of mathematics under a single scheme, named 'category'. For instance, one well-known category contains sets and the functions between them. Other examples include ordered sets with order-preserving functions, groups with group morphisms, topological spaces with continuous functions and vector spaces with linear maps.

Monads are defined by Mac Lane and constitute a central object of category theory: at first glance, they define a transformation of our category, subject to strict structural conditions. For example, turning a set into the collection of its subsets constitutes a monadic transformation (called *powerset*). If we consider a set A, mapping A to the set of lists containing elements of A is also a monad (the *list* monad).

A crucial connection between monads and Computer Science is established in Moggi's work [30, 31]. As Moggi explains, monads correspond to 'notions of computations': they can be used to model computational effects such as exceptions, probabilistic calculation or non-determinism. In this spirit, monads are at the core of functional programming and languages like Haskell because they encapsulate particular patterns and side-effects of sequential computation (lists, exceptions...) in a unified way.

Another aspect of monads is the correspondence with algebraic theories. They are said to be 'presented' with a collection of operation symbols, and equations governing them. The list monad, for instance, corresponds to the theory of monoids: an associative binary operation and its unit. The powerset monad, on the other hand, is presented by the theory of complete lattices. This notion of presentation is a crucial asset for the study of universal algebra, since monads can now be seen as a categorical counterpart to algebraic theories.

One question that arises naturally is that of compositionality: can we combine two monads and obtain a third monad? In term of computational effects, can we for instance associate non-deterministic calculations with exceptions? Many attempts at combining the features of different monads can be found in the literature, e.g. [19, 13, 14]. Some methods to compose monads in a sequential way have been thoroughly examined [26, 27], and some results of incompatibility between monads have been established [21, 42]. However, this field of research has been the subject of numerous errors and the problem of monad composition is still lacking a general approach. As related in [42], the history of the aforementioned powerset monad demonstrates this difficulty. Manes and Mulry first claimed in [26] that this monad could be composed with itself. They later retracted this result and acknowledged their mistake, but a similar claim was made in [20] by Klin and Rot, only to be disproved by Klin and Salamanca in [21] as they show that no such composition can be constructed.

This thesis takes a fresh approach to the question of monad composition and connects it with the idea of algebraic presentation. Inspired by the works cited above, we develop a general method to compose monads as well as precise conditions for its success. In a nutshell, consider two monads that we wish to compose: we represent one of them with its algebraic features, and the other as a categorical transformation. We then examine whether this transformation preserves the aforementioned algebraic constructs, and show that a successful preservation allows to compose our two monads. In this introductory chapter we recall the main mathematical objects of interest, state our main contributions, review related work and outline the thesis.

Monads and Algebras

A collection of objects together with a collection of arrows (or *morphisms*) between objects constitutes a *category* [25]. As mentioned above, vector spaces and topological spaces are examples of categories, but we focus our work on the category of sets and functions, which we denote by **Set**.

Every monad T is based on a transformation from a category to itself (an *endofunctor*), and is provided with additional categorical structure. A typical monad is the powerset monad $\mathcal{P}$, which maps a set X to $\mathcal{P}(X) = \{U \mid U \subseteq X\}$. Other examples include the aforementioned list monad $\mathcal{L}$ and the distribution monad $\mathcal{D}$, which maps every set X to the set of probability distributions on X.

A capital aspect of monads is their relation with universal algebra. An *algebra for the monad* T (which we denote by $\mathcal{A}$) is given by a carrier set A and an algebraic structure defined by the monad T. All algebras for T form a category (called the *Eilenberg-Moore category* of T). For some monads called *finitary*, we can precisely describe these algebras as instances of an algebraic theory: a set Σ of operations of finite arity, and a set of equations E governing them. Consider for instance the list monad $\mathcal{L}$. If $\mathcal{A}$ is an algebra for $\mathcal{L}$, then A has a structure of monoid: we can define on it a binary operation $\bullet$ and a

unit 1 such that the following equations hold:

$$x \bullet 1 = 1 \bullet x = x \qquad x \bullet (y \bullet z) = (x \bullet y) \bullet z \tag{1.1}$$

The list monad is said to be *presented* by the theory of monoids. Similarly, the distribution monad is presented by the theory of convex algebras.

Two monads cannot always be composed. For instance, it has been shown in [21] that composing $\mathcal{P}$ with itself can never form a monad. One crucial object permits to compose monads T and S: a transformation called *distributive law* that we denote by λ [2], which allows to define a monad structure on the sequential composition TS. A large part of this thesis is dedicated to the general construction of distributive laws and the conditions under which they can be built.

Approach

Our aim is to compose monads by constructing distributive laws. For this purpose, we develop a method which relies on the preservation of algebraic features by a monad. In this section, we introduce the problem of algebraic preservation and present the starting point of our approach.

Consider for instance the transformation operated by the powerset monad, and the algebraic theory of monoids. Let us assume that the set A is provided with a structure of monoid (1.1). Can we say that $\mathcal{P}A$ has the same algebraic structure as A, namely is it still a monoid? Answering this question must be done in two steps: first, investigate what binary operation acts on $\mathcal{P}A$ and what its unit is, then examine if, for this new operation, the equations of a monoid hold on $\mathcal{P}A$.

In the case of $\mathcal{P}$, this question has been extensively studied in a pioneering paper by Gautam [11]. For the first step, he shows that an operation $\bullet$ on the set A and its unit 1 can be redefined in a very natural way on $\mathcal{P}A$. Gautam then proves that all the equations in (1.1) are preserved by $\mathcal{P}$, therefore the powerset monad preserves the structure of monoid — we say that $\mathcal{P}$ can be *lifted* to the category of monoids.

More generally, Gautam shows that the equations preserved by $\mathcal{P}$ are exactly the ones where both sides feature the same variables, and with each variable appearing exactly once on each side. This type of equality is called *linear* - it is easy to see that for instance, all monoid laws (1.1) are linear. Incidentally, an equation such as $x \times x = x$ (*idempotence*) is not preserved by $\mathcal{P}$.

To move away from $\mathcal{P}$ and study the general question, we consider a monad T and an algebraic theory (Σ, E), and we examine the preservation of the latter by the former. Assume that T belongs to the class of *monoidal monads*; such monads are introduced by Kock [23] and happen to provide a canonical framework for the preservation of algebraic features. For a set A provided with an Σ-algebra structure, we take inspiration from [36] and make use of the monoidal structure of T to redefine algebraic operations, this time with on the set TA. In other words, we lift the monad T to the category of Σ-algebras.

This lifting is always possible for monoidal monads and defines a new monad $\widehat{T}$. It now remains to show whether the equations in E preserved by T, which constitutes the main challenge of this thesis.

Contributions

The central idea of this thesis is the construction of distributive laws via the preservation of algebraic features by monoidal monads; in this section we present our original findings.

We consider two monads T and S, and the algebraic theory (Σ, E) presenting S. Having first constructed the lifting $\widehat{T}$ to the category of Σ-algebras, we show that if $\widehat{T}$ preserves all equations in E, then there exists a distributive law permitting to compose T and S (Theorem 3.14). This offers a general method for the construction of a composite monad TS when T is monoidal and S is finitary.

Verifying that an equation of E is preserved by $\widehat{T}$ is however a tricky process. As explained above, preservation is not automatic and may depend on the variable layout on both sides of the equality. Inspired by [11], we outline two main classes of equations: *drop* equations, in which one variable appears on one side only (for instance $x \times 0 = 0$), and *dup* equations, where some variables appear more than once on one side ($x \times x = x$). We call the equations which are dup but not drop 'strict-dup' (and conversely 'strict-drop').

First, we consider any equation and construct a tailor-made categorical condition for its preservation.

- Sufficient condition for equation preservation (Theorem 4.11).

This process highlights the importance of the variable layout, and allows for the following classification of monads: we show that *affine* monads (a well-known monadic class, thoroughly studied in [15]) preserve equations that feature deleted variables. Then we focus on strict-dup equations and prove that they are preserved by *relevant* monads (also examined in [15]). The combination of affine and relevant properties results in *Cartesian* monads, and our results compose nicely to show that such monads preserve all equations.

- Affine monads preserve strict-drop equations (Theorem 4.28).
- Relevant monads preserve strict-dup equations (Theorem 4.34).
- Cartesian monads preserve all equations (Theorem 4.38).

So far our results on equation preservation are only sufficient conditions; we also assess whether they are necessary. If a monoidal monad T preserves a drop equation, does that mean that T is affine? If we consider any equation in a class slightly larger than strict-drop, then the answer is yes, which means that we obtain an equivalence between affineness and preservation of strict-drop equations. The case of dup equations and relevant monads turns out to be more complicated. For a large class of strict-dup equations, preservation does imply relevance and a similar equivalence is obtained. For

some other dup equations, we show that relevance is not necessary for T to preserve them; instead, a weaker property of n-relevance characterises preservation. Note that our results lead to an interesting corollary: the undecidability of the affineness of a monad.

- For strict-drop equations, preservation is equivalent to affineness (Theorem 5.5).
- Affineness of a monad is undecidable (Theorem 5.7).
- For some dup equations, preservation is equivalent to relevance (Theorem 5.17).
- For other dup equations, preservation is equivalent to n-relevance (Theorem 5.24).

Having shown that our categorical conditions are necessary, we now know where our method does not apply: for instance if T is not affine and S is presented by some strict-drop equations. In such cases, we present a series of strategies to overcome the incompatibility of the two monads. We operate increasingly precise modifications on our monads to enforce the preservation of some algebraic features, sometimes moving away from the strictly monoidal context to construct less conventional distributive laws.

- Methods to combine monads when some equations are not preserved (Chapter 6).

In the final part of this thesis, we focus our attention on the algebraic presentation of both T and S, and translate our previous categorical notions of monoidal, affine and relevant monads in algebraic terms (Theorems 7.4 to 7.21). Then we recall the distributive law constructed via our method and examine it under this new light. It then appears that the algebraic transformation it operates shows a very familiar pattern. Finally, we use algebraic techniques to prove a new characterisation of relevant monads and to study non-monoidal distributive laws .

- Algebraic characterisation of our distributive law (Theorem 7.23).
- Relevant monads are characterised by the preservation of discerning equations (Theorem 7.30).
- Properties of non-monoidal distributive laws (Theorems 7.31 and 7.32).

Generally speaking, the scope of this thesis covers monoidal monads on **Set**. In some cases, generalisations outside this setting are possible and will be discussed.

Related Work

The problem of combining the features of several monads has been examined by Moggi in [30, 31] with the notion of 'monad transformer'. Later works by Hyland and Power [13, 14] bring Moggi's concept further as they devise diverse options for combination. Our approach differs from theirs and is based on Beck's seminal paper on distributive

laws [2]. Constructing such a law has been the object of several papers; the closest ones to our work being by King and Wadler [19] and Manes and Mulry [26, 27]. Some of our results can be seen as an extention of the ideas of [26]: we confirm Manes and Mulry's findings on monoidal monads and generalise them to the classes of affine and relevant monads. Our construction of distributive laws is closely related to the methods of Bonsangue, Hansen, Kurz and Rot in [5], in which the authors exhibit a method to 'quotient' a distributive law by the equations of a theory.

Our work also finds its place in a more recent context of research on distributive laws. First, a proof strategy by Gordon Plotkin is used by Varacca in [38] to show that $\mathcal{P}$ and $\mathcal{D}$ cannot be composed via a distributive law. Inspired by this method, Klin and Salamanca's paper [21] provides a definite result on the incompatibility of some monads with $\mathcal{P}$, which resonates with our characterisation of relevant monads by their preservation of idempotence. On a more algebraic level, Zwart and Marsden generalise Plotkin's idea in [42] to classify a wide selection of cases where no distributive law can be found. As mentioned before, our approach of the preservation of algebraic features can also be related to Gautam's pioneering work in [11]. His study of 'complex algebras' (brought further by Grätzer in [12]) makes no use of category theory but describes exactly the monoidal lifting of $\mathcal{P}$ to $\mathbf{Alg}(\Sigma)$; this thesis generalises his findings to other monoidal monads. Our proof strategy for relevant monads and dup equations involves a graphical framework inspired by existing works. Describing monoidal functors with a box-like diagram dates back to [9] and more recently to [29]. McCurdy gives a detailed study of this approach in [28], however our framework differs slightly as he assumes properties of affineness and relevance to always hold, leading to a more robust graphical representation. Some of our methods for treating the case of incompatible monads are heavily inspired by works by Bonchi, Sokolova and Silva [3] as well as Bonchi, Sokolova and Vignudelli [4], as both papers describe a modification of the powerset monad enforcing the preservation of some dup equations.

Finally, the algebraic techniques we apply to monad composition are closely related to the ones employed by Zwart and Marsden in [42]. However contrary to them we focus on monoidal monads and our constructive results on the corresponding distributive law, which complements their findings on incompatible monads. Our work in the final chapter also relies on existing results on the combination of algebraic theories, mostly from [42, 40, 34].

Outline

In Chapter 2, we precisely define the notions of category, monoidal monad, algebraic theory and distributive law, as they form the theoretical backbone of this thesis. The end of the chapter features a summary of the different approaches to monad composition.

Chapter 3 introduces our method to compose a monoidal monad T with a finitary monad S. We present a canonical construction for the preservation of algebraic features by making use of the monoidal structure of T, and we build the lifting $\widehat{T}$. Finally,

we complete this chapter by showing that our construction yields a distributive law $\lambda\colon ST \to TS$ when the equations presenting S are preserved. Chapter 3 is based the following published paper.

- [10] Fredrik Dahlqvist, Alexandra Silva, and Louis Parlant. Layer by layer: Composing Monads. ICTAC 2018.

Chapter 4 focuses on the preservation of equations. Starting with the lifting defined in Chapter 3, we define *residual diagrams* which act as precise sufficient conditions for a given equation to be preserved. This construct allows to demonstrate that all monoidal monads preserve linear equations (confirming a result of [26]). We then move on to affine monads and show that they preserve strict-drop equations, and by the same mechanism we show that relevant monads preserve strict-dup equations. Finally, Cartesian monads (which are both relevant and affine) preserve all equations. Once more, these findings are based on the following publication.

- [10] Fredrik Dahlqvist, Alexandra Silva, and Louis Parlant. Layer by layer: Composing Monads. ICTAC 2018.

In Chapter 5, we aim to establish the necessity of the sufficient conditions constructed in Chapter 4. We succeed in the case of affine monads, which are characterised by the preservation of any strict-dup equation. This permits us to show that the question of whether a given monad is affine is undecidable. Moving on to strict-dup equations, we prove that the preservation of many of them implies the property of relevance. However, some strict-dup equations are shown to coincide with a weaker property called n-relevance. Chapter 5 is based on the following recent publication.

- [33] Louis Parlant, Jurriaan Rot, Alexandra Silva, and Bas Westerbaan. Preservation of Equations by Monoidal Monads. MFCS 2020.

Chapter 6 focuses on the cases where one monad T fails to preserve some equations presenting another monad S. We give a series of tactics to overcome this issue: the first one is a coarse modification of S to ensure that the composition succeeds. The second one is a restriction of T to its affine, relevant or Cartesian part which, depending on the equation, will automatically allow to preserve it. A more precise method is then presented: intuitively, we construct the lifting $\widehat{T}$ then restrict it to preserve the equations of S, forming a monad on S-algebras; this framework is inspired by unpublished joint works with Gerco Van Heerdt. Our last strategy leaves the framework of monoidal monads and builds an ad hoc lifting of T which, by construction, preserves the desired equations.

Finally, Chapter 7 examines again the concepts of monoidal, affine and relevant monads as well as the distributive law from Chapter 3 from the point of view of algebraic theories. This time, we consider the presentations of both S and T and we regard the

object TX as a set of algebraic terms verifying the equations presenting T. We give algebraic counterparts to categorical properties introduced in Chapters 2 and 4, and we study the action of the distributive law from Chapter 3 on terms of STX. The final two sections focus on a new characterisation of relevant monads and on a study of a particular non-monoidal distributive law, both explored via algebraic techniques.

Chapter 2

Preliminaries

We summarise in this chapter the mathematical basis of our work. We assume that the reader is familiar with category theory, but we recall the definitions of the main objects. We define monads, algebras and the existing methods to combine them.

2.1 Category Theory

This work arises in the context of category theory. This framework (see [25] for a comprehensive introduction) finds an essential use in many settings, of interest here are the study of universal algebra and the modelling of computation. We recall here the basic notions and refer to [25] and [1] for details.

Definition 2.1 (Category)**.** A *Category* **C** consists of:

- A collection of objects denoted $X, Y, Z \ldots$
- A collection of arrows denoted f, g, h (or *morphisms*) between two objects. We write $f \colon X \to Y$, X and Y are respectively denoted *domain* and *codomain* of f.

Arrows $f \colon X \to Y$ and $g \colon Y \to Z$ can be *composed* into an arrow $g \circ f \colon X \to Z$, and each object admits an *identity arrow* $\mathrm{id}_X \colon X \to X$. Composition of arrows is associative and identity arrows act as units for it.

Essentially, category theory places the focus on consistent structures (*objects*) and transformations that preserve them (*arrows*).

Example 2.2. We denote by **Set** the category whose objects are sets and whose arrows are functions between them. △

Most of our constructions are done in **Set**. Other examples of categories include groups with group homomorphisms and partially ordered sets (*posets*) with monotone functions.

Definition 2.3 (isomorphism)**.** The arrow $f\colon X \to Y$ is an *isomorphism* if there is an arrow $g\colon Y \to X$ such that :

$$f \circ g = \mathrm{id}_Y \quad \text{and} \quad g \circ f = \mathrm{id}_X$$

We say that two objects X and Y are isomorphic if there is an isomorphism between them, and write $X \simeq Y$.

Example 2.4. In **Set**, all finite sets of the same size are isomorphic. In particular, all singletons are isomorphic. Working up to isomorphism, we represent any singleton as the object $1 = \{*\}$. △

Definition 2.5 (Functor)**.** A *functor* $F\colon \mathbf{C} \to \mathbf{D}$ between two categories is a mapping of objects to objects and arrows to arrows such that:

- $F(f\colon A \to B) = F(f)\colon F(A) \to F(B)$
- $F(g \circ f) = F(g) \circ F(f)$
- $F(\mathrm{id}_X) = \mathrm{id}_{F(X)}$

An *endofuntor* is a functor $F\colon \mathbf{C} \to \mathbf{C}$ mapping a category to itself. The *identity functor* Id maps all objects and arrows to themselves.

Definition 2.6 (Natural Transformation)**.** For categories $\mathbf{C}$, $\mathbf{D}$ and functors $F, G\colon \mathbf{C} \to \mathbf{D}$, a *natural transformation* $\zeta\colon F \to G$ is a family of arrows in $\mathbf{D}$ indexed by objects in $\mathbf{C}$:

$$(\zeta_X\colon FX \to GX)_{X \in \mathbf{C}}$$

which make the following diagram commute for any $f\colon X \to Y$ arrow of $\mathbf{C}$.

$$\begin{array}{ccc} FX & \xrightarrow{\zeta_X} & GX \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FY & \xrightarrow[\zeta_X]{} & GY \end{array} \tag{2.1}$$

In a nutshell, a natural transformation defines a morphism between functors. This concept is crucial in our studies as it defines a consistent mapping across objects of a category. Another notion is central in our work: providing a category with the structure of a monoid. We recall here the corresponding definitions.

Definition 2.7 (Monoidal Category)**.** A *monoidal category* consists of a category $\mathbf{C}$ equipped with:

- a bifunctor $\otimes\colon \mathbf{C} \times \mathbf{C} \to \mathbf{C}$
- an object I called *unit* or *identity*
- a natural isomorphism $\alpha_{X,Y,Z}\colon (X \otimes Y) \otimes Z \to X \otimes (Y \otimes Z)$ called *associator*
- a natural isomorphism $\rho_X\colon X \otimes I \to X$ called *right unitor*
- a natural isomorphism $\rho'_X\colon I \otimes X \to X$ called *left unitor*

α, ρ and ρ' are subject to coherence conditions such as: $\rho_A \otimes \mathrm{id}_B = id_A \otimes \rho'_B \circ \alpha_{A,I,B}$. More details can be found in [25].

Example 2.8. The category **Set** together with the Cartesian product $\times$ and its unit 1 forms a monoidal category. Associators and unitors are given by natural isomorphisms such as $(X \times Y) \times Z \simeq X \times (Y \times Z)$ and $X \times 1 \simeq X$ △

We concept of *adjunction* is crucial in category theory. We give here the general definition, and give more details later when focusing on the adjunctions involved in algebraic structures.

Definition 2.9 (Adjoint)**.** Two functors $F\colon \mathbf{D} \to \mathbf{C}$ and $G\colon \mathbf{C} \to \mathbf{D}$ are said adjoint if there exist two natural transformations $\epsilon\colon FG \to 1_{\mathbf{C}}$ and $\eta\colon 1_{\mathbf{D}} \to GF$ such that:

$$\epsilon F \circ F\eta = \mathrm{id}_F$$

$$G\epsilon \circ \eta_G = \mathrm{id}_G$$

In this case, we write $F \dashv G$ and say that F is left adjoint to G.

2.2 Monads

Monads are the central object of our work. In this section, we define them and their attributes, and give some introductory examples.

Definition 2.10 (Monad)**.** A Monad (T, η, μ) on a category $\mathbf{C}$ is given by:

- An endofunctor $T\colon \mathbf{C} \to \mathbf{C}$
- A natural transformation $\eta\colon \mathrm{Id} \to T$ called the *unit*
- A natural transformation $\mu\colon TT \to T$ called the *multiplication*

subject to the following commuting diagrams, for X any object of $\mathbf{C}$:

$$\begin{array}{ccc} TTTX & \xrightarrow{T\mu_X} & TTX \\ {\scriptstyle \mu_{TX}}\downarrow & & \downarrow{\scriptstyle \mu_X} \\ TTX & \xrightarrow[\mu_X]{} & TX \end{array} \qquad \begin{array}{ccc} TX & \xrightarrow{\eta_{TX}} & TTX \\ {\scriptstyle T\eta_X}\downarrow & \searrow^{\mathrm{id}_{TX}} & \downarrow{\scriptstyle \mu_X} \\ TTX & \xrightarrow[\mu_X]{} & TX \end{array}$$

For the purpose of this study, we will consider monads on **Set**. Here are some of the most common ones:

Example 2.11. The *full (covariant) powerset monad* $(\mathcal{P}, \eta, \mu)$ is given by:

- $\mathcal{P}\colon \mathbf{Set} \to \mathbf{Set}$ maps a set X to the set of all (not necessarily finite) subsets of X. $\mathcal{P}$ acts pointwise on morphisms.
- The unit maps an element x to the singleton $\{x\}$.

$$\begin{aligned} \eta_X\colon X &\to \mathcal{P}X \\ x &\mapsto \{x\} \end{aligned}$$

- The multiplication takes the union of sets.

$$\begin{aligned} \mu_X\colon \mathcal{P}\mathcal{P}X &\to \mathcal{P}X \\ V &\mapsto \bigcup \{U \mid U \in V\} \end{aligned}$$

A variation on $\mathcal{P}$ is the *Non-Empty Powerset monad* $\mathcal{P}^+$, which maps a set X to all its subsets except for $\emptyset$. Its action on morphisms, its unit and its multiplication are all defined as above. △

Example 2.12. The *finite (covariant) powerset monad* $(\mathcal{P}_f, \eta, \mu)$ is given by:

- $\mathcal{P}_f \colon \mathbf{Set} \to \mathbf{Set}$ maps a set X to the set of all finite subsets of X. $\mathcal{P}_f$ acts pointwise on morphisms.
- The unit and the multiplication are defined as for $\mathcal{P}$.

Again, the non-empty finite powerset monad also forms a monad, which we denote by ${\mathcal{P}_f}^+$. △

Example 2.13. The *List Monad* $(\mathcal{L}, \eta^{\mathcal{L}}, \mu^{\mathcal{L}})$ is given by:

- $\mathcal{L} \colon \mathbf{Set} \to \mathbf{Set}$ maps a set X to the set of all finite lists with elements in X. The action of $\mathcal{L}$ on morphism is pointwise: $\mathcal{L}(f[w_1, \ldots, w_n]) = [f(w_1), \ldots, f(w_n)]$
- The unit maps an element x to the singleton list $[x]$

$$\eta^{\mathcal{L}}_X \colon X \to \mathcal{L}X$$
$$x \mapsto [x]$$

- The multiplication concatenates a list of lists into a single list.

$$\mu^{\mathcal{L}}_X \colon \mathcal{L}\mathcal{L}X \to \mathcal{L}X$$
$$[[w_1, \ldots, w_{k_1}], \ldots, [w_{k_{n-1}}, \ldots, w_{k_n}]] \mapsto [w_1, \ldots, w_{k_n}]$$

△

Example 2.14. For a semiring $\mathbb{S}$, let $(\mathbf{M}_{\mathbb{S}}, \eta^{\mathbb{S}}, \mu^{\mathbb{S}})$ be the *generalised multiset* defined as follows.

- A multiset $\xi \in \mathbf{M}_{\mathbb{S}}X$ is a map $X \to \mathbb{S}$ with finite support, that we may also write as the formal sum $\sum_x \xi(x)x$. For $f\colon X \to Y$ we define $(\mathbf{M}_{\mathbb{S}}f)(\xi)(y) = \sum_{x;\ f(x)=y} \xi(x)$ which boils down to $(\mathbf{M}_{\mathbb{S}}f)(\sum_i s_i x_i) = \sum_i s_i f(x_i)$ when writing the multisets as formal sums.

- The unit is defined by $\eta^{\mathbb{S}}(x) = 1 \bullet x$

- the multiplication is given by $\mu^{\mathbb{S}}(\delta)(f) = \sum_x \delta(f)f(x)$, that is $\mu^{\mathbb{S}}(\sum_i s_i \sum_j t_{ij}x_{ij}) = \sum_{i,j} s_i t_{ij} x_{ij}$.

△

Example 2.15. A natural special case of $\mathbf{M}_{\mathbb{S}}$ is obtained by taking $\mathbb{S} = \mathbb{N}$; we simply call it the *Multiset Monad* (M, η^M, μ^M). $M\colon \mathbf{Set} \to \mathbf{Set}$ maps a set X to the set of all finite multisets with elements in X. M acts pointwise on morphisms. The unit maps an element x to the singleton $\{x\}$, which is the multiset containing one instance of x, and the multiplication takes the union of multisets (which may contain duplicates).

$$\begin{aligned} \eta^M_X &\colon X \to MX & \qquad \mu^M_X &\colon MMX \to MX \\ x &\mapsto \{x\} & V &\mapsto \bigcup\{U \mid U \in V\} \end{aligned}$$

△

Example 2.16. The *Distribution Monad* $(\mathcal{D}, \eta^{\mathcal{D}}, \mu^{\mathcal{D}})$ is given by:

- $\mathcal{D}\colon \mathbf{Set} \to \mathbf{Set}$ maps a set X to the set of all finitely supported probability distributions on X: $\mathcal{D}X = \{\mathbb{P}\colon X \to [0,1] \mid \sum_{x\in X} \mathbb{P}(x) = 1, \mathrm{supp}(\mathbb{P}) \text{ finite}\}$. The action on morphisms is defined as $\mathcal{D}f\colon \mathcal{D}X \to \mathcal{D}Y, \mathbb{P} \mapsto \lambda y. \sum_{x\in f^{-1}(y)} \mathbb{P}(x)$

- The unit maps an element x to the Dirac distribution at x.

$$\begin{aligned} \eta^{\mathcal{D}}_X &\colon X \to \mathcal{D}X \\ x &\mapsto \delta_x \end{aligned}$$

- The multiplication is defined as:

$$\mu_X^{\mathcal{D}} \colon \mathcal{D}\mathcal{D}X \to \mathcal{D}X$$
$$\mathbb{R} \mapsto \lambda x. \sum_{\mathbb{P} \in \mathrm{supp}(\mathbb{R})} \mathbb{R}(\mathbb{P}).\mathbb{P}(x)$$

△

Example 2.17. For a set E, the *Exception Monad* $((\_ + E), \eta^E, \mu^E)$ is defined as:

- $T_E = (\_ + E) \colon \mathbf{Set} \to \mathbf{Set}$ maps a set X to the disjoint union $X + E$. $T_E(f) = f + \mathrm{id}_E$.
- The unit is the inclusion $X \to X + E$.
- The multiplication $(X + E) + E \to X + E$ is the identity on X and maps both copies of E to a single one.

A special instance of the exception monad, called *Maybe Monad*, corresponds to the case $E = 1$. We denote this monad by $\perp$ or sometimes $X + 1$. △

Example 2.18. For a set C, we define the *Reader* monad $\mathcal{R}_C$ as $((-)^C, \eta^C, \mu^C)$, sometimes written X^C.

- $\mathcal{R}_C = (-)^C \colon \mathbf{Set} \to \mathbf{Set}$ maps a set X to the set of morphisms $C \to X$. For $c \in C$, $f \colon X \to Y$, $s \in X^C$, we have $\mathcal{R}_C(f)(c) \colon s \mapsto f(s(c))$.
- The unit maps $a \in X$ to the constant function $x \mapsto a$.
- The multiplication $(X^C)^C \to X^C$ evaluates twice and for $c \in C, t \in (X^C)^C$ yields $(\mu(t))(c) = t(c)(c)$

△

Example 2.19. For a commutative monoid $(M, \bullet, 1)$, we define the *Writer* monad $\mathcal{W}_M$ as $((-) \times M, \eta_M, \mu_M)$; we will sometimes write it $X \times M$.

- $\mathcal{W}$ maps a set X to the product $X \times M$; for $f \colon X \to Y$ we have $\mathcal{W}_M(f) = f \times \mathrm{id} \colon X \times M \to Y \times M$.

- The unit maps $a \in X$ to the pair $(x, 1)$.
- The multiplication $(X \times M) \times M \to X \times M$ maps $((x, m), n)$ to $(x, m \bullet n)$.

△

See for instance [32] for uses of the exception, reader and writer monads.

Example 2.20. For a set S, the *State Monad* $((S \times \_)^S, \eta^S, \mu^S)$, sometimes called *Side-effects Monad* is defined as:

- $(S \times \_)^S \colon \mathbf{Set} \to \mathbf{Set}$ maps a set X to the set of morphisms $S \to S \times X$. Given a morphism $f \colon X \to Y$, we define $(S \times f)^S \colon (S \times X)^S \to (S \times Y)^S$ in the following manner. If for $h \in (S \times X)^S$ we have $h \colon s \mapsto (s', x)$, we define $h' \colon s \mapsto (s', f(x))$. Finally, $(S \times f)^S \colon h \mapsto h'$.
- The unit maps an element x to the trivial morphism:

$$\begin{aligned} \eta^S_X \colon X &\to (S \times X)^S \\ x &\mapsto \lambda s.(s, x) \end{aligned}$$

- The multiplication successively applies morphisms:

$$\begin{aligned} \mu^{\mathcal{P}}_X \colon (S \times (S \times X)^S)^S &\to (S \times X)^S \\ f &\mapsto \lambda s.g(s') \text{ where } f(s) = (s', g) \end{aligned}$$

△

Definition 2.21. A *monad morphism* between (S, η^S, μ^S) and (T, η^T, μ^T) is a natural transformation $u \colon S \to T$ verifying the following properties:

$$u \circ \eta^S = \eta^T \qquad \mu^T \circ Tu \circ u_T = u \circ \mu^S$$

2.2.1 Strong Monads

We consider a monoidal category $(\mathbf{C}, \otimes, I)$, we write α for the associator of $(\mathbf{C}, \otimes, I)$ and ρ, ρ' the right and left unitors respectively. $\mathsf{swap}\colon (-) \otimes (-) \to (-) \otimes (-)$ is the argument-swapping transformation (natural in both arguments). In [31], Moggi defines the following concept, crucial for our study:

Definition 2.22 (Strong Monad)**.** A *strong monad* is a monad (T, η, μ) provided with a natural transformation $\mathsf{st}_{X,Y}\colon X \otimes TY \to T(X \otimes Y)$ subject to the following commuting diagrams:

$$\begin{array}{ccc} 1 \otimes TX & \xrightarrow{\rho'_{TX}} & TX \\ & {\scriptstyle \mathsf{st}_{1,X}} \searrow & \downarrow {\scriptstyle T\rho_X} \\ & & T(1 \otimes X) \end{array} \qquad (2.2)$$

$$\begin{array}{ccc} (X \otimes Y) \otimes TZ & \xrightarrow{\alpha_{TX,TY,TZ}} & X \otimes (Y \otimes TZ) \\ \downarrow {\scriptstyle \mathsf{st}_{X\otimes Y,Z} \otimes \mathrm{id}_{TZ}} & & \downarrow {\scriptstyle \mathrm{id}_X \otimes \mathsf{st}_{Y,Z}} \\ T((X \otimes Y) \otimes Z) & & X \otimes T(Y \otimes Z) \\ & {\scriptstyle T\alpha_{X,Y,Z}} \searrow & \downarrow {\scriptstyle \mathsf{st}_{X,Y\otimes Z}} \\ & & T(X \otimes (Y \otimes Z)) \end{array} \qquad (2.3)$$

$$\begin{array}{ccc} X \otimes Y & \xrightarrow{\mathrm{id} \otimes \eta_Y} & X \otimes TY \\ & {\scriptstyle \eta_{X\otimes Y}} \searrow & \downarrow {\scriptstyle \mathsf{st}_{X,Y}} \\ & & T(X \otimes Y) \end{array} \qquad (2.4)$$

$$\begin{array}{ccccc} X \otimes T^2Y & \xrightarrow{\mathsf{st}_{X,TY}} & T(X \otimes TY) & \xrightarrow{T\mathsf{st}_{X,Y}} & TT(X \otimes Y) \\ \downarrow {\scriptstyle \mathrm{id}_X \otimes \mu_Y} & & & & \downarrow {\scriptstyle \mu_{X\otimes Y}} \\ X \otimes TY & & \xrightarrow{\qquad \mathsf{st}_{X,Y} \qquad} & & T(X \otimes Y) \end{array} \qquad (2.5)$$

For instance, the powerset monad on **Set** is strong: we can define $\mathsf{st}_{X,Y}\colon X \otimes \mathcal{P}Y \to \mathcal{P}(X \otimes Y)$ by $\mathsf{st}_{X,Y}(a, U) = \{a\} \times U$. This construction can be generalised to the following result:

Theorem 2.23 ([31])**.** *All monads on* **Set** *are uniquely strong.*

For every strong monad T, a *costrength* $\mathsf{st}'_{X,Y}\colon TX \otimes Y \to T(X \otimes Y)$ can be defined as $T(\mathsf{swap}_{Y,X}) \circ \mathsf{st}_{Y,X} \circ \mathsf{swap}_{TX,Y}$. T is then said to be *commutative* when strength and costrength can be combined in any order, in other words when the following diagram commutes:

$$\begin{array}{ccccc} TX \otimes TY & \xrightarrow{\mathsf{st}_{TX,Y}} & T(TX \otimes Y) & \xrightarrow{T\mathsf{st}'_{X,Y}} & TT(X \otimes Y) \\ \downarrow {\scriptstyle \mathsf{st}'_{X,TY}} & & & & \downarrow {\scriptstyle \mu_{X\otimes Y}} \\ T(X \otimes TY) & \xrightarrow{T\mathsf{st}_{X,Y}} & TT(X \otimes Y) & \xrightarrow{\mu_{X\otimes Y}} & T(X \otimes Y) \end{array} \qquad (2.6)$$

2.2.2 Monoidal Monads

Some monads interact in a well-behaved way with the monoidal structure of the category. We define here their properties as they will be the center of our study.

Definition 2.24 (Monoidal Functor)**.** A *lax monoidal functor*, or simply a monoidal functor, is an endofunctor $F\colon \mathbf{C} \to \mathbf{C}$ together with natural transformations $\psi_{X,Y}\colon FX \otimes FY \to F(X \otimes Y)$ and $\psi^0\colon I \to FI$ satisfying the diagrams:

$$\begin{array}{ccc} FX \otimes I & \xrightarrow{\mathrm{id}_{FX} \otimes \psi^0} & FX \otimes FI \\ \downarrow{\scriptstyle \rho_{FX}} & & \downarrow{\scriptstyle \psi_{X,I}} \\ FX & \xleftarrow{F\rho_X} & F(X \otimes I) \end{array} \qquad \text{(MF.1)}$$

$$\begin{array}{ccc} I \otimes FX & \xrightarrow{\psi^0 \otimes \mathrm{id}_{FX}} & FI \otimes FX \\ \downarrow{\scriptstyle \rho'_{FX}} & & \downarrow{\scriptstyle \psi_{I,X}} \\ FX & \xleftarrow{F\rho'_X} & F(I \otimes X) \end{array} \qquad \text{(MF.2)}$$

$$\begin{array}{ccc} (FX \otimes FY) \otimes FZ & \xrightarrow{\alpha_{FX,FY,FZ}} & FX \otimes (FY \otimes FZ) \\ \downarrow{\scriptstyle \psi_{X,Y} \otimes \mathrm{id}_{FZ}} & & \downarrow{\scriptstyle \mathrm{id}_{FX} \otimes \psi_{Y,Z}} \\ F(X \otimes Y) \otimes FZ & & FX \otimes F(Y \otimes Z) \\ \downarrow{\scriptstyle \psi_{X \otimes Y, Z}} & & \downarrow{\scriptstyle \psi_{X, Y \otimes Z}} \\ F((X \otimes Y) \otimes Z) & \xrightarrow{F\alpha_{X,Y,Z}} & F(X \otimes (Y \otimes Z))) \end{array} \qquad \text{(MF.3)}$$

Definition 2.25. Finally, we require precise interactions between the monoidal structure and the unit and multiplication of the monad. A *monoidal monad* T on a monoidal category is a monad whose underlying functor is monoidal for a natural transformation $\psi_{X,Y}\colon TX \otimes TY \to T(X \otimes Y)$ and $\psi^0 = \eta_I$, the unit of the monad at I, and whose unit and multiplication are monoidal natural transformations, that is to say:

$$\begin{array}{ccc} X \otimes Y & \xrightarrow{\eta_X \otimes \eta_Y} & TX \otimes TY \\ & {\scriptstyle \eta_{X \otimes Y}} \searrow & \downarrow{\scriptstyle \psi_{X,Y}} \\ & & T(X \otimes Y) \end{array} \qquad \text{(MM.1)}$$

$$\begin{array}{ccccc} T^2X \otimes T^2Y & \xrightarrow{\psi_{TX,TY}} & T(TX \otimes TY) & \xrightarrow{T\psi_{X,Y}} & TT(X \otimes Y) \\ \downarrow{\scriptstyle \mu_X \otimes \mu_Y} & & & & \downarrow{\scriptstyle \mu_{X \otimes Y}} \\ TX \otimes TY & & \xrightarrow{\qquad\qquad \psi_{X,Y} \qquad\qquad} & & T(X \otimes Y) \end{array} \qquad \text{(MM.2)}$$

Moreover, a monoidal monad is called *symmetric monoidal* if

$$\begin{array}{ccc} TX \otimes TY & \xrightarrow{\psi_{X,Y}} & T(X \otimes Y) \\ {\scriptstyle \mathsf{swap}_{TX,TY}}\downarrow & & \downarrow{\scriptstyle T\mathsf{swap}_{X,Y}} \\ TY \otimes TX & \xrightarrow[\psi_{Y,X}]{} & T(Y \otimes X) \end{array} \tag{SYM}$$

Example 2.26. The powerset monad $\mathcal{P}$ is monoidal, by defining ψ as the *Cartesian product*.

$$\psi_{X,Y} : \mathcal{P}X \times \mathcal{P}Y \to \mathcal{P}(X \times Y)$$
$$(U, V) \mapsto U \times V$$

Moreover, $\mathcal{P}\mathsf{swap}(U \times V) = \{(y, x) \mid x \in U, y \in V\} = V \times U = \mathsf{swap}(U, V)$, therefore $\mathcal{P}$ is symmetric. △

Example 2.27. The distribution monad $\mathcal{P}$ is monoidal, by defining ψ as the *measure product*.

$$\psi_{X,Y} : \mathcal{D}X \times \mathcal{D}Y \to \mathcal{D}(X \times Y)$$
$$(\nu_1, \nu_2) \mapsto \nu_1 \times \nu_2$$

Again, for a set B we have $\mathcal{D}\mathsf{swap}(\nu_1 \times \nu_2)(B) = \nu_1 \times \nu_2\{(x, y) \mid \mathsf{swap}(x, y) \in B\} = \nu_2 \times \nu_1(B)$ thus $\mathcal{D}$ is symmetric. △

Example 2.28. The monad $\mathcal{R}_C$ is monoidal.

$$\psi_{X,Y} : X^C \times Y^C \to (X \times Y)^C$$
$$(f, g) \mapsto (c \mapsto (f(c), g(c)))$$

△

The following result classically relates strength and monoidality.

Theorem 2.29 ([23])**.** *A monad is symmetric monoidal if and only if it is strong commutative.*

We now present a more general version of this theorem. For monoidal categories which are sufficiently similar to $(\mathbf{Set}, \times, 1)$, being monoidal is equivalent to being symmetric monoidal. The criteria on $(\mathbf{C}, \otimes, I)$ in the following theorem are due to [35] and generalize the strength unicity result of [31, Prop. 3.4].

Theorem 2.30. *Let $T : \mathbf{C} \to \mathbf{C}$ be a monad over a monoidal category $(\mathbf{C}, \otimes, I)$ whose tensor unit I is a separator of $\mathbf{C}$ (i.e. $f, g : X \to Y$ and $f \neq g$ implies $\exists x : I \to X$ s.th. $f \circ x \neq g \circ x$) and such that for any morphism $z : I \to X \otimes Y$ there exist $x : I \to X, y \to Y$ such that $z = (x \otimes y) \circ \rho_I^{-1}$. Then the following are equivalent*

(i) There exists a unique natural transformation $\psi_{X,Y} : TX \otimes TY \to T(X \otimes Y)$ making T monoidal

(ii) There exists a unique strength $\mathsf{st}_{X,Y} : X \times TY \to T(X \otimes Y)$ making T commutative

(iii) There exists a unique natural transformation $\psi_{X,Y} : TX \otimes TY \to T(X \otimes Y)$ making T symmetric monoidal

In particular, monoidal monads on $(\mathbf{Set}, \times, 1)$ are necessarily symmetric (and thus commutative), in other words diagram (SYM) always commutes for such monads. As we will see in Chapter 3, the class of (symmetric) monoidal monads on **Set** is the base of our work because of their compatibility with categorical algebraic structures. However, the conditions of Theorem 2.30 can be used to generalise our work outside **Set** ; we will see later how the uniqueness of the strength and the symmetry property are essential in our construction.

Example 2.31. Consider the writer monad $\mathcal{W}_M$ with $(M, \bullet, 1)$ a monoid. We show that this monad is monoidal if M is commutative.

$$\psi_{X,Y} : (X \times M) \times (Y \times M) \to ((X \times Y) \times M)$$
$$(x, m), (y, n) \mapsto (x, y, m \bullet n)$$

Let X, Y be sets and $x \in X, y \in Y, m, n \in M$. We have $\psi \circ \mathsf{swap}((x, m), (y, n)) = ((y, x), n \bullet m)$ and $\mathcal{W}_M \mathsf{swap} \circ \psi((x, m), (y, n)) = ((y, x), m \bullet n)$. Therefore $\mathcal{W}_M$ is symmetric (hence monoidal) if and only if $m \bullet n = n \bullet m$ for all $m, n \in M$. In the rest of this work, we focus on monoidal monads and will assume that M is commutative. $\triangle$

Theorem 2.30 gives us a powerful way to determine whether a monad is monoidal, as we only have to verify that their unique strength map is commutative.

Example 2.32. The exception monad is not monoidal, for $|E| > 1$. For $x \in X, y \in Y, e \in E$, we see that the unique arrow $X \times (Y + E) \to (X, Y) + E$ verifying the conditions for strength maps (x, y) to itself and (x, e) to e. Consider now $f \in E$ and the pair $(e, f) \in (X + E) \times (Y + E)$. For this element, the diagram 2.6 does not commute: one branch yields e while the other one returns f. Therefore $X + E$ is not commutative, and not monoidal. △

Example 2.33. For the same reason, the list monad is not monoidal. The unique strength arrow maps $(x, [y_1, \dots y_n])$ to $[(x, y_1), \dots (x, y_n)]$. Again, diagram 2.6 does not commute, which can be seen for sets X, Y such that $a, b \in X, c, d \in Y$ and a, b, c, d are distinct. On the input $[[a, b], [c, d]]$, one branch of the diagram yields the list of pairs $[(a, c), (a, d), (b, c), (b, d)]$ whereas the other branch results in $[(a, c), (b, c), (a, d), (b, d)]$. Hence $\mathcal{L}$ is not commutative, and is not a monoidal monad. △

Although ψ is essentially a binary operator, several instances of it may be composed in a natural way to form a n-ary monoidal map. Because of Diagram (MF.3), this composition is associative and gives rise to the following definition.

Definition 2.34. For any $n \geq 2$, we define $\psi^{(n)} \colon TX_1 \times \cdots \times TX_n \to T(X_1 \times \cdots \times X_n)$ as the n-ary version of ψ, constructed in an associative manner from the binary version.

The associativity of this construction follows from Diagram (MF.3).

We now present a few technical lemmas describing how monoidal monads interact with the monoidal structure of **Set**. They will become useful in some proofs appearing in Chapter 5.

Lemma 2.35. *Let T be a monoidal monad, let $k, l \in \mathbb{N}$. Then the following diagram commutes:*

$$\begin{array}{ccccc}
(TA)^k \times 1 \times (TA)^l & & \xrightarrow{\mathrm{id}^k \times \eta_1 \times \mathrm{id}^l} & & (TA)^k \times T1 \times (TA)^l \\
\downarrow{\scriptstyle \mathrm{id}^k \times \rho'} & & & & \downarrow{\scriptstyle \psi^{(k+l+1)}} \\
(TA)^{k+l} & \xrightarrow[\psi^{(k+l)}]{} & T(A^{k+l}) & \xleftarrow[T(\mathrm{id}^k \times \rho')]{} & T(A^k \times 1 \times A^l)
\end{array}$$

Proof.

$$
\begin{aligned}
&T(\mathrm{id}^k \times \rho') \circ \psi^{(k+l+1)} \circ (\mathrm{id}^k \times \eta_1 \times \mathrm{id}^l) \\
&\quad = T(\mathrm{id}^k \times \rho') \circ \psi^{(k+2)} \circ (\mathrm{id}^k \times \mathrm{id} \times \psi^{(l)}) \circ (\mathrm{id}^k \times \eta_1 \times \mathrm{id}^l) && \text{by associativity} \\
&\quad = T(\mathrm{id}^k \times \rho') \circ \psi^{(k+2)} \circ (\mathrm{id}^k \times \eta_1 \times \mathrm{id}^l) \circ (\mathrm{id}^k \times \mathrm{id} \times \psi^{(l)}) && \text{by product} \\
&\quad = T(\mathrm{id}^k \times \rho') \circ \psi^{(k+1)} \circ (\mathrm{id}^k \times \psi) \circ (\mathrm{id}^k \times \eta_1 \times \mathrm{id}^l) \circ (\mathrm{id}^k \times \mathrm{id} \times \psi^{(l)}) && \text{by associativity} \\
&\quad = \psi^{(k+1)} \circ (\mathrm{id}^k \times T\rho') \circ (\mathrm{id}^k \times \psi) \circ (\mathrm{id}^k \times \eta_1 \times \mathrm{id}^l) \circ (\mathrm{id}^k \times \mathrm{id} \times \psi^{(l)}) && \text{by naturality of } \psi^{(k+1)} \\
&\quad = \psi^{(k+1)} \circ (\mathrm{id}^k \times \rho') \circ (\mathrm{id}^k \times \mathrm{id} \times \psi^{(l)}) && \text{by } MF.2 \\
&\quad = \psi^{(k+1)} \circ (\mathrm{id}^k \times \psi^{(l)}) \circ (\mathrm{id}^k \times \rho') && \text{by naturality of } \rho' \\
&\quad = \psi^{(k+l)} \circ (\mathrm{id}^k \times \rho') && \text{by associativity}
\end{aligned}
$$

□

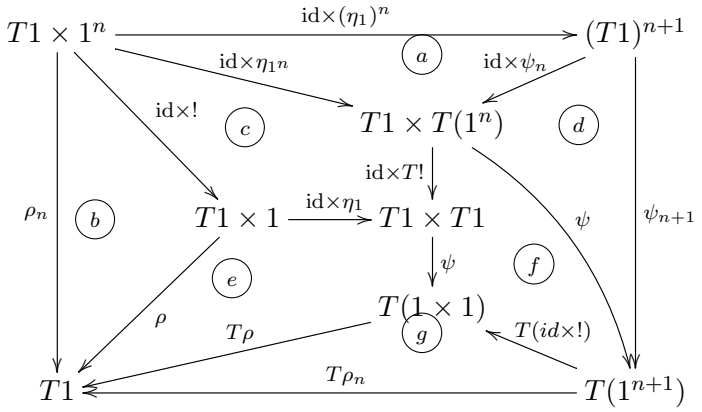
Recall the *unitor* isomorphism $\rho_X \colon X \times 1 \to X$. For $n \geq 0$, we denote by ρ_n the composition $\rho_n = \rho \circ (\mathrm{id} \times !) \colon X \times 1^n \to X$, and we obtain a property similar to a n-ary version of Diagram MF.1, but only involving the object 1.

Lemma 2.36. *For a monoidal monad T on a Cartesian monoidal category* **C***, the diagram*

$$
\begin{array}{ccc}
T1 \times 1^n & \xrightarrow{\mathrm{id} \times \eta_1^n} & T1 \times (T1)^n \\
{\scriptstyle \rho_n}\downarrow & & \downarrow{\scriptstyle \psi} \\
T1 & \xleftarrow[T\rho_n]{} & T(1 \times 1^n)
\end{array}
\tag{2.7}
$$

commutes for all $n \geq 1$.

Proof. For $n = 1$ the result is simply (MF.1). Assuming we've shown it for n, consider the following diagram.



(a) commutes by (MM.1). (b) and (g) follow from the definition of ρ_n. (c) and (f) commute by naturality of ψ, (d) by monoidality, and (e) by the (MF.1) again. □

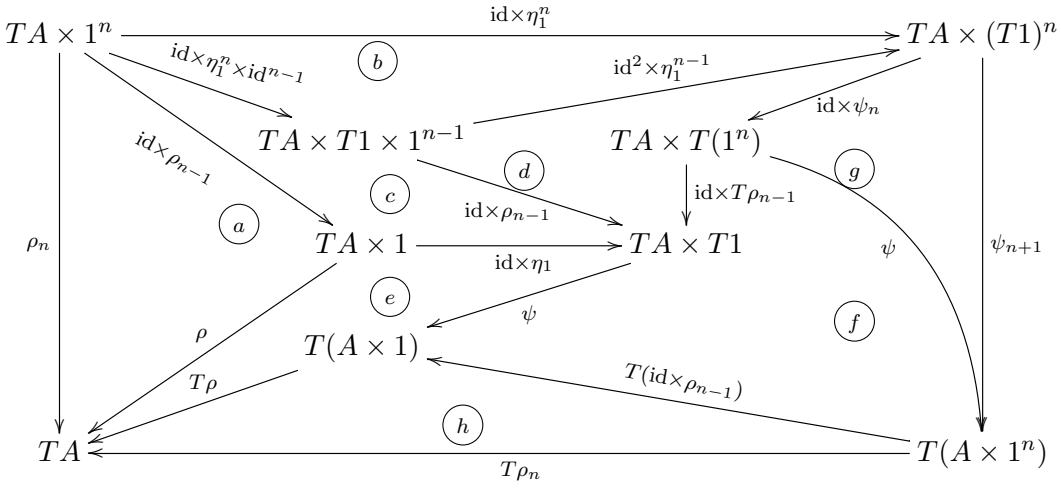
We can now generalise the previous lemma to obtain a n-ary version of Diagram MF.1.

Lemma 2.37. *For a monoidal monad T on a Cartesian monoidal category* **C**, *the diagram*

$$\begin{array}{ccc} TA \times 1^n & \xrightarrow{\mathrm{id}\times\eta_1^n} & TA \times (T1)^n \\ {\scriptstyle\rho_n}\downarrow & & \downarrow{\scriptstyle\psi} \\ TA & \xleftarrow[T\rho_n]{} & T(A \times 1^n) \end{array} \tag{2.8}$$

commutes for all $n \geq 1$.

Proof.



(a) and (h) commute by definition of ρ_n. (b) commutes trivially; (c) commutes by naturality of ρ_{n-1}; (d) by Lemma 2.36; (e) by (MF.2); (f) by naturality of ψ. □

Finally, we show that different instances of n-ary and m-ary unitors interact in a natural way.

Lemma 2.38. *Let T be a monoidal monad on a Cartesian monoidal category $\mathbf{C}$, let A be an object of $\mathbf{C}$. Let $m, n > 0$, let f be a morphism $1^m \to 1^n$. Then the following diagram commutes:*

$$\begin{array}{ccc} TA \times 1^m & \xrightarrow{\mathrm{id}\times f} & TA \times 1^n \\ & {\scriptstyle \rho_m} \searrow \quad \nearrow {\scriptstyle (\rho_n)^{-1}} & \\ & TA & \end{array} \tag{2.9}$$

Proof. Consider the following commuting diagram.

$$\begin{array}{ccccc} TA \times 1^m & & \xrightarrow{\mathrm{id}\times f} & & TA \times 1^n \\ & {\scriptstyle \mathrm{id}\times !} \searrow & & \swarrow {\scriptstyle \mathrm{id}\times !} & \\ & & TA \times 1 & & \\ & {\scriptstyle \rho_m} \searrow & \downarrow {\scriptstyle \rho} & \swarrow {\scriptstyle \rho_n} & \\ & & TA & & \end{array} \tag{2.10}$$

Thus $\rho_m = \rho_n \circ (\mathrm{id} \times f)$, whence $(\rho_n)^{-1} \circ \rho_m = \mathrm{id} \times f$. □

2.2.3 Algebras and Theories

In the rest of this work, we make extensive use of category theory and monads to model algebraic structures. First we will define those structures as generated by operation symbols and laws governing them, then we will examine the relations with categorical constructs.

Algebraic Theories

Definition 2.39 (Signature)**.** A *signature* Σ is a set of operation symbols, each provided with a natural number defined as its *arity*. We write $|\sigma|$ for the arity of $\sigma \in \Sigma$.

Definition 2.40 (Σ-terms)**.** For X a set, the set of $\Sigma-$*terms* over X is defined as follows:

- For $x \in X$, X is a term (sometimes called 'generator' or 'variable').

- If $t_1, \dots t_n$ are terms and σ is a symbol of Σ with arity n, then $\sigma(t_1, \dots t_n)$ is a term.

For a term t with variables in Y, f a function mapping variables in Y to terms, we write $t[f(y)/y]$ or $t[f]$ the corresponding substitution.

Definition 2.41 (Algebraic Theory)**.** An *Algebraic Theory* $\mathbb{T}$ is defined as:

- A signature Σ
- A set X of variables
- A set E of pairs of Σ-terms over X. We refer to E as *equations* or *axioms* of $\mathbb{T}$ and the pair $(u, v) \in E$ will be written $u = v$.

We will often leave the set of variables implicit. When two Σ-terms t_1, t_2 can be proved equal using equational logic and the axioms of $\mathbb{T}$, we write $t_1 = t_2$.

Definition 2.42 (Linear Theory)**.** An equation $u = v$ is called *linear* if u and v are formed on the same variables, with each variable appearing exactly once in each term. A theory $\mathbb{T}$ is said *linear* if all of its axioms are linear.

Example 2.43. The theory of binary trees has a signature containing a binary symbol $\bullet$, and no axiom. △

Example 2.44. The theory of semigroups has a signature containing a binary symbol $\bullet$, and a law of associativity:

$$x \bullet (y \bullet z) = (x \bullet y) \bullet z$$

△

Example 2.45. The theory of monoids has a signature containing one constant e and a binary symbol $\bullet$. The axioms express associativity and unit:

$$x \bullet e = x$$

$$e \bullet x = x$$

$$x \bullet (y \bullet z) = (x \bullet y) \bullet z$$

△

Example 2.46. The theory of Abelian groups has a signature containing one constant 0, one unary symbol $-$ and one binary symbol $+$. The axioms extend the axioms of monoids with laws of commutativity and inverse:

$$x + 0 = x \qquad 0 + x = x$$
$$x + (y + z) = (x + y) + z \qquad x + (-x) = 0$$
$$x + y = y + x$$

△

Example 2.47. The theory of join semilattices has signature $\Sigma = \{+\}$ and axioms expressing associativity, commutativity and idempotence:

$$x + (y + z) = (x + y) + z$$
$$x + y = y + x$$
$$x + x = x$$

△

Example 2.48. The theory of bounded join semilattices extends the previous theory with a unit. This time we write $\Sigma = \{+, 0\}$, and the equations are:

$$x + 0 = x \qquad 0 + x = x$$
$$x + y = y + x \qquad x + x = x$$
$$x + (y + z) = (x + y) + z$$

△

The first two theories presented here are linear, but the theories of join semilattices and bounded join semilattices are not, as the variable x appears twice in the same term in the law of idempotence $x + x$. We will see later in Chapters 4 and 5 that linear and non-linear equations interact with categorical constructs in very different ways.

Example 2.49. The theory of convex algebras is described as follows: the signature contains an operator $\oplus_\lambda$ for each $\lambda \in]0,1[$ and the axioms are:

$$x \oplus_\lambda x = x$$

$$x \oplus_\lambda y = y \oplus_{1-\lambda} x$$

$$x \oplus_\lambda (y \oplus_\tau z) = (x \oplus_{\frac{\lambda}{\lambda+(1-\lambda)\tau}} y) \oplus_{\lambda+(1-\lambda)\tau} z$$

△

Example 2.50. As shown in [3], there exists another theory representing convex algebras. Its signature includes the symbols of Example 2.49, but also convex sum operations $\oplus_0$ and $\oplus_1$. Their axioms are laws of projection:

$$x \oplus_0 y = y$$

$$x \oplus_1 y = x$$

It is easy to see that these operators are not much expressive power, they are sometimes called 'trivial'. We will see in later chapters that this dual definition of convex algebras leads to interesting distinctions in terms of categorical behaviour. △

So far we focused solely on algebraic constructs; they are however related with categorical structures, as signatures and equations correspond to notions of algebras that we detail in the next section.

Σ-Algebras and (Σ, E)-Algebras

We now introduce two categorical notions: first, the algebra defined by a signature, then the one generated by the whole theory.

Definition 2.51 (Σ-algebra)**.** For Σ a signature, a Σ-algebra $\mathcal{A}$ is formed of:

- A carrier set A
- For each symbol $\sigma \in \Sigma$ of arity $|\sigma|$, a morphism $\sigma_{\mathcal{A}} \colon A^{|\sigma|} \to A$.

Categorically speaking, a Σ-algebra is in fact an algebra for the polynomial functor $\mathsf{H}_\Sigma = \coprod_{\sigma\in\Sigma}(-)^{|\sigma|}$.Therefore a Σ-algebra morphism between $\beta\colon \mathsf{H}_\Sigma X \to X$ and $\gamma\colon \mathsf{H}_\Sigma Y \to Y$ is a map $f\colon X \to Y$ such that $\gamma \circ \mathsf{H}_\Sigma f = f \circ \beta$. The category of Σ-algebras and Σ-algebra morphisms is denoted $\mathbf{Alg}(\Sigma)$.

Definition 2.52 (Free and Forgetful Functors)**.** We define two structural functors:

- $\mathsf{F}_\Sigma\colon \mathbf{Set} \to \mathbf{Alg}(\Sigma)$ is the functor building free Σ-algebras: $\mathsf{F}_\Sigma X$ is the set of all Σ-terms built on X.
- $\mathsf{U}_\Sigma\colon \mathbf{Alg}(\Sigma) \to \mathbf{Set}$ is the functor that 'forgets' the structure of a Σ-algebra and returns its carrier set.

Definition 2.53. Given a Σ-algebra $\mathcal{A}$, a free Σ-term s built over variables in a set V, and a valuation map $v\colon V \to \mathsf{U}_\Sigma\mathcal{A}$, we define the interpretation $[\![s]\!]_v$ of s in $\mathcal{A}$ recursively in the obvious manner. We say that an equation $s = t$ between free Σ-terms is valid in $\mathcal{A}$, denoted $\mathcal{A} \models s = t$, if for every valuation $v\colon V \to \mathsf{U}_\Sigma\mathcal{A}$, $[\![s]\!]_v = [\![t]\!]_v$.

Definition 2.54 ((Σ, E)-Algebra)**.** For a signature Σ and a set E of equations we define a (Σ, E)-algebra as a Σ-algebra in which all the equations in E are valid.

We denote by $\mathbf{Alg}(\Sigma, E)$ the subcategory of $\mathbf{Alg}(\Sigma)$ consisting of (Σ, E)-algebras.

Algebras and Adjunctions

Recall the concept of adjunction from Definition 2.9. It is related to the categorical study of algebraic theories since categories of Σ-algebras and (Σ, E)-algebras admit adjunctions with **Set** through the well-known construction using free and forgetful functors.

- F_Σ and U_Σ form an adjunction:

$$\mathsf{F}_\Sigma \dashv \mathsf{U}_\Sigma\colon \mathbf{Alg}(\Sigma) \to \mathbf{Set} \tag{2.11}$$

- Similarly, there exists a functor $\mathsf{F}\colon \mathbf{Set} \to \mathbf{Alg}(\Sigma, E)$ building free (Σ, E)-algebras which is left adjoint to the obvious forgetful functor:

$$\mathsf{F} \dashv \mathsf{U}\colon \mathbf{Alg}(\Sigma, E) \to \mathbf{Set} \tag{2.12}$$

For a set A, $\mathsf{F}(A)$ is the free (Σ, E)-algebra generated on A, in other words it can be seen as the set of equivalence classes of Σ-terms built on A under provable equality in equational logic with axioms in E.

Definition 2.55 (Free Model Monad)**.** For a theory $\mathbb{T}$ with signature Σ and equations in E, we say that the *free model monad* induced by $\mathbb{T}$ is the monad arising from the previous adjunction as $\mathsf{U} \circ \mathsf{F}$.

It is well-known [25] that every finitary monad on **Set** arises from an adjunction 2.12. Specifically in this context, every finitary monad on **Set** is the free model monad induced by some algebraic theory.

Definition 2.56 (Presentation)**.** We say that the monad T *corresponds to the theory* $\mathbb{T}$, or that T is *presented by* $\mathbb{T}$, if T is a free model monad induced by $\mathbb{T}$.

We can now associate previous monads with algebraic theories:

- The List Monad $\mathcal{L}$ is presented by the theory of monoids (see Example 2.45)
- The Non-empty Finite Powerset Monad ${\mathcal{P}_f}^+$ is presented by the theory of join-semilattices (see Example 2.47)
- The Finite Powerset Monad $\mathcal{P}_f$ is presented by the theory of bounded join-semilattices (see Example 2.48)
- The Distribution Monad $\mathcal{D}$ is presented by the theory of convex algebras. (see Example 2.49)

The powerset monad $\mathcal{P}$ is presented by the theory of complete lattices, which requires an operation symbol of infinite arity to represent the infimum of an infinite subset. Therefore $\mathcal{P}$ is not finitary; when our study requires the use of an algebraic presentation we will instead focus on its finite version $\mathcal{P}_f$.

We can now consider new monads that arise as the free model monad of a theory: we will write $\mathcal{A}$ the *monad of Abelian groups*, obtained via Definition 2.55 from the theory of Example 2.46.

Eilenberg-Moore categories

In this section, we define categories of algebras which arise from monads. An algebra for the monad T is a set X together with an map $\alpha \colon TX \to X$ such that the diagrams in (2.13) commute. A morphism $(X, \alpha) \xrightarrow{f} (Y, \beta)$ of T-algebras (also called *algebra*

homomorphism) is a morphism $X \xrightarrow{f} Y$ in **Set** verifying $\beta \circ Tf = f \circ \alpha$.

$$\begin{array}{ccc} TTX & \xrightarrow{\mu_X} & TX \\ {\scriptstyle T\alpha}\downarrow & & \downarrow{\scriptstyle \alpha} \\ TX & \xrightarrow{\alpha} & X \end{array} \qquad\qquad \begin{array}{ccc} X & \xrightarrow{\eta_X} & TX \\ & {\scriptstyle \mathrm{id}_X}\searrow & \downarrow{\scriptstyle \alpha} \\ & & X \end{array} \tag{2.13}$$

The category of T-algebras and T-algebra morphisms is called the *Eilenberg-Moore* category of the monad T, and denoted $\mathcal{EM}(T)$. There is an obvious forgetful functor $\mathsf{U}_E \colon \mathcal{EM}(T) \to \mathbf{Set}$ which sends an algebra to its carrier, it has a left adjoint $\mathsf{F}_E \colon \mathbf{Set} \to \mathcal{EM}(T)$ which sends a set X to the free T-algebra (TX, μ_X). Note that the adjunction $\mathsf{F}_E \dashv \mathsf{U}_E$ gives rise to the monad T.

Definition 2.57 (Lifting to $\mathcal{EM}(T)$)**.** A *lifting* of a functor $F \colon \mathbf{Set} \to \mathbf{Set}$ to $\mathcal{EM}(T)$ is a functor $\hat{F}$ on $\mathcal{EM}(T)$ such that $\mathsf{U}_E \circ \widehat{F} = F \circ \mathsf{U}_E$

$$\begin{array}{ccc} \mathcal{EM}(T) & \xrightarrow{\hat{F}} & \mathcal{EM}(T) \\ {\scriptstyle U_E}\downarrow & & \downarrow{\scriptstyle U_E} \\ \mathbf{Set} & \xrightarrow{F} & \mathbf{Set} \end{array} \tag{2.14}$$

It appears now naturally that when a monad arises from an adjunction related to an algebraic theory, the algebras for this monad are algebras for this theory.

Lemma 2.58 ([25] VI.8. Theorem 1)**.** *For any adjunction of the form* (2.12)*,* $\mathcal{EM}(\mathsf{UF})$ *and* $\mathbf{Alg}(\Sigma, E)$ *are equivalent categories.*

Example 2.59. By using the presentation given above for $\mathcal{P}_f$, we know that $\mathcal{EM}(\mathcal{P}_f)$ is equivalent to the category of bounded join-semilattices, which we denote by **JSL**. △

The functors connecting $\mathcal{EM}(\mathsf{UF})$ and $\mathbf{Alg}(\Sigma, E)$ are traditionally called *comparison functors*, and we will denote them by $M \colon \mathcal{EM}(\mathsf{UF}) \to \mathbf{Alg}(\Sigma, E)$ and $K \colon \mathbf{Alg}(\Sigma, E) \to \mathcal{EM}(\mathsf{UF})$. Consider first the free monad F_Σ for a signature Σ (i.e. the monad generated by the adjunction (2.11)). The comparison functor $M \colon \mathbf{Alg}(\Sigma) \to \mathcal{EM}(\mathsf{F}_\Sigma)$ maps the free F_Σ-algebra over X, that is $\mu_X^{\mathsf{F}_\Sigma} \colon \mathsf{F}_\Sigma^2 X \to \mathsf{F}_\Sigma X$ to the free H_Σ-algebra over X which we shall denote by $\alpha_X \colon \mathsf{H}_\Sigma \mathsf{F}_\Sigma X \to \mathsf{F}_\Sigma X$. It is well-known that α_X is an isomorphism.

For the monad $T = \mathsf{UF}$, the adjunction between $\mathcal{EM}(T)$ and **Set** results in a universal property. This becomes crucial in the study of algebras, as it intuitively means that any map from a set to the carrier of a T-algebra factors uniquely through the free T-algebra.

Lemma 2.60. *Let* $\mathcal{A} \in \mathcal{EM}(T)$ *be a* T*-algebra with carrier* A*. For every set* X*, for every map* $f \colon X \to A$*, there exists a unique algebra homomorphism* $h \colon (TX, \mu_X) \to \mathcal{A}$ *such that* $h \circ \eta_X = f$*.*

A similar universal property applies to F_Σ:

Lemma 2.61. *Let $\mathcal{A}$ be a Σ-algebra with carrier A. For every map $f\colon X \to A$, there exists a unique map $f^\#\colon \mathsf{F}_\Sigma X \to A$ such that $f^\# \circ \eta_X^{\mathsf{F}_\Sigma} = f$.*

This allows for an alternative formulation of Definition 2.53. For a Σ-algebra $\mathcal{A}$ with carrier A and a map $f\colon X \to A$, we write $f^\#$ the unique map $\mathsf{F}_\Sigma X \to A$ obtained via the universal property of Lemma 2.61, which intuitively extends f to free Σ-terms.

Definition 2.62. Let $\mathcal{A}$ be a Σ-algebra with carrier A. We say that an equation $s = t$ between terms of $\mathsf{F}_\Sigma X$ is valid in $\mathcal{A}$, denoted $\mathcal{A} \models s = t$, if for every map $f\colon X \to A$ we have $f^\#(s) = f^\#(t)$.

Kleisli Categories

Monads also give rise to another type of categories, which we briefly define. The *Kleisli category* $\mathcal{K}l(T)$ has the same objects as **Set**, but morphisms $X \to Y$ in $\mathcal{K}l(T)$ are maps $X \to TY$ in **Set**. The identity map $X \to X$ in $\mathcal{K}l(T)$ is the unit $\eta_X\colon X \to T(X)$. Composition $g \odot f$ in $\mathcal{K}l(T)$ uses T's multiplication: $g \odot f = \mu \circ T(g) \circ f$. There is a forgetful functor $U_K\colon \mathcal{K}l(T) \to \mathbf{Set}$, sending X to $T(X)$ and f to $\mu \circ T(f)$, which has a left-adjoint $F_K\colon \mathbf{Set} \to \mathcal{K}l(T)$ leaving objects unchanged and postcomposing maps with the unit. Again the monad associated with this adjunction is T itself.

Definition 2.63 (Lifting to $\mathcal{K}l(T)$)**.** A lifting of a functor F to $\mathcal{K}l(T)$ is a functor $\hat{T}$ on $\mathcal{K}l(T)$ such that the $\hat{F} \circ F_K = F_K \circ F$.

$$\begin{array}{ccc} \mathcal{K}l(T) & \xrightarrow{\hat{F}} & \mathcal{K}l(T) \\ {\scriptstyle F_K}\uparrow & & \uparrow{\scriptstyle F_K} \\ \mathbf{Set} & \xrightarrow{F} & \mathbf{Set} \end{array} \tag{2.15}$$

2.3 Composing Monads

We are now familiar with the concept of monad, which is the central object of this work. We have explored a series of examples and established a correspondence between monads and algebraic theories. The question of composition arises naturally: can we combine monads? Can the features of two different monads be captured by a single composite monad? Or, in a similar manner as functors, can we sequentially compose monads and expect the result to also have a monadic structure? In this section, we examine several methods used to combine monads.

2.3.1 Monad Transformers

Moggi's seminal paper [31] lays the foundation for the study of monads, as the author relates categorical constructs with features of computation. In [30], Moggi pursues his study and defines a first procedure to combine the features of different monads. He constructs objects named 'monad transformers', which augment one monad with the algebraic features of another. Given a monad T, Moggi exhibits a series of constructions:

- For E a set of exceptions, $T(\_ + E)$ forms a new monad modelling *T-computations with exceptions.*
- For a set S representing states, $T_{\text{seff}}(\_) = (T(\_ \times S))^S$ defines the monad of *T-computations with side-effects.*

His work is groundbreaking in many ways. Even though his Kleisli-like model differs from ours, we are inspired by his approach: Moggi gives a toolbox for combining monadic features, and we aim to replicate this modus operandi in the scope of our study.

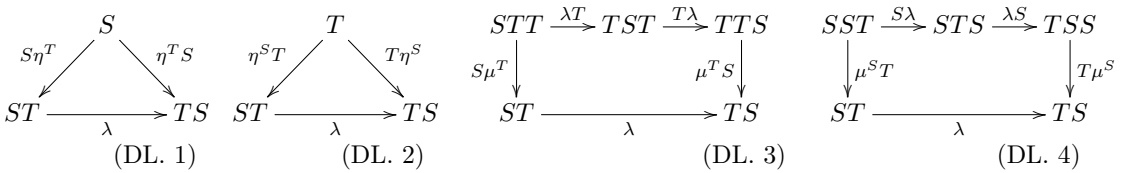
2.3.2 Sum, Tensor

Hyland, Plotkin and Power pursue this search for moand combination and extend Moggi's work in [13] and [14]. They generalise Moggi's transformers to binary operations: can the transformation of T into $T(\_ + E)$ incorporating exceptions into T-computations be seen as an operation $T_E \circ T$ on monads? They define the *sum* of two monads, combining them without any algebraic interaction between the theories presenting them. Moggi's exception transformer $T(\_ + E)$ is a special case of sum. On the contrary, the *tensor*, or commutative combination, yields a monad where operations of one theory always commute with operations of the other. Moggi's side-effects transformer $T_{seff}(\_) = (T(\_ \times S))^S$ realises in fact the tensor of T and $(\_ \times S)^S$.

2.3.3 Distributive Law

We focus our efforts on a third tool for monad combination, the one that follows from functorial composition.

Definition 2.64 (Distributive Law)**.** Let (S, η^S, μ^S) and (T, η^T, μ^T) be monads. A *distributive law of S over T* (see [2]) is a natural transformation $\lambda\colon ST \to TS$ satisfying:



If λ only satisfies (DL. 2) and (DL. 4), we will say that λ is a distributive law of the the monad (S, η^S, μ^S) over *functor* T, or in the terminology of [17], an $\mathcal{EM}$-law of S over T. Dually, if λ only satisfies (DL. 1) and (DL. 3), λ is known as a distributive law of the *functor* S over the monad (T, η^T, μ^T), or $\mathcal{K}l$-law of S over T [17]. This notation will become clear later with Theorem 2.66 which relates distributive laws with Eilenberg-Moore and Kleisli categories.

A distributive law is therefore at first glance a transformation 'commuting' monads S and T. Let us consider an example and define a distributive law of the list monad over the finite powerset monad. We require a natural transformation $\mathcal{LP} \to \mathcal{PL}$, in other words a operation mapping *lists of sets* to *sets of lists*. It must be defined regardless of the the object we apply it to, in order to be natural. An instinctive choice is, when considering a list of sets, to pick successively an element in each set to form a new list, then to take the set of all such lists:

$$\begin{aligned} &\lambda_X^{\mathcal{PL}} \colon \mathcal{LP}X \to \mathcal{PL}X \\ &[U_1, \ldots, U_n] \mapsto \{[u_1, \ldots, u_n], u_i \in U_i, 1 \leq i \leq n\} \end{aligned} \tag{2.16}$$

Checking commutativity of the naturality diagram, as well as (DL. 1), (DL. 2) (DL. 3) and (DL. 4) proves that $\lambda^{\mathcal{PL}}$ forms indeed a distributive law.

We now show how distributive laws define a monadic structure for the functorial composition ST.

Theorem 2.65. *If there exists a distributive law $\lambda \colon TS \to ST$ of the monad T over the monad S, then the composition of S and T also forms a monad (ST, u, m), whose unit u and multiplication m are given by:*

$$X \xrightarrow{\eta_X^T} TX \xrightarrow{\eta_{TX}^S} STX \qquad\qquad STSTX \xrightarrow{S\lambda_{TX}} SSTTX \xrightarrow{\mu_{TTX}^S} STTX \xrightarrow{S\mu_X^T} STX$$

u_X and m_X

Finally, we show that distributive laws are related to monad algebras. The following well-known result will be crucial for our work, as it describes the correspondence between liftings to Eilenberg-Moore or Kleisli categories and distributive laws.

Theorem 2.66. *[2, 18]*

- *$\mathcal{EM}$-laws $\lambda \colon SF \to FS$ and liftings of F to $\mathcal{EM}(S)$ are in one-to-one correspondence.*
- *$\mathcal{K}l$-laws $\lambda \colon FT \to TF$ and liftings of F to $\mathcal{K}l(T)$ are in one-to-one correspondence.*

As this construction of the correspondence between $\mathcal{EM}$-laws and liftings of F to $\mathcal{EM}(S)$ will be used later in our work, we recall here some explicit details. This correspondance is based on Beck's seminal paper [2] where he shows the canonical equivalence

between monad liftings and distributive laws ; we make use of this classical construction and refer the reader to his work for more details.

From a distributive law to a lifting

Assuming we have a distributive law $\lambda\colon SF \to FS$, we lift F to the category of S-algebras in the following manner. Let (X, α) be such an algebra with $\alpha\colon SX \to X$; we map it to the algebra (FX, α) with $\alpha' = (F\alpha \circ \lambda)\colon STX \to FX$.

From a lifting to a distributive law

Let $\widehat{F}$ be a lifting of F to $\mathcal{EM}(S)$. We consider the free S-algebra $\mathcal{A} = (SX, \mu^S_X)$ and compute $\mathcal{A}' = \widehat{F}(\mathcal{A}) = (TSX, \alpha')$. Let us now consider the set FX and the map $F\eta^S_X\colon TX \to TSX$: by the universal property of S-algebras (Lemma 2.60), there exists a unique algebra homomorphism $h\colon \mathcal{A} \to \mathcal{A}'$ such that $h \circ \eta^S_X = F\eta^S_X$. The distributive law is then constructed by defining $\lambda_X = h$. The explicit definition of λ is given by $\lambda_X = \alpha' \circ SF\eta^S_X$.

Using distributive laws to compose monads will be our method of choice, but we will see that their construction is a complicated task. The next chapter focuses on the method we have developed to tackle this problem.

Chapter 3

A Method for Composing Monoidal Monads

Composing two given monads is known to be a difficult process. The existence of a distributive law is not guaranteed, and the literature features no general method to address this question.

The scope of this method (and of most of this thesis) is monoidal monads on **Set**. In this chapter we describe a method for their composition with a given finitary monad. When our procedure succeeds, we obtain the desired distributive law and may build a composite monad.

This construction is the central object of our work and relies on the following idea. Given monads (T, η^T, μ^T) and (S, η^S, μ^S), we want to form the composition TS. As seen in Section 2.2.3, if S is a finitary monad, it admits a presentation in the form of a signature Σ and a set of equations E. Then we consider an algebra $\mathcal{A}$ for S, described as a (Σ, E)-algebra. We show that our composition problem amounts to a very natural question about this structure: if we apply the monadic transformation T to $\mathcal{A}$, do we obtain another (Σ, E)-algebra? In other words, are the algebraic features of S preserved by the application of T?

Consider for instance the powerset monad $\mathcal{P}$ and the ring of integers $\mathbb{Z}$. is $\mathcal{P}\mathbb{Z}$, the set of all subsets of $\mathbb{Z}$, still a ring? To answer this question, we must first examine how the algebraic symbols of a ring, namely $\times$, $+$, $-$, 1 and 0, can be understood on sets rather than integers. For $U, V \in \mathcal{P}\mathbb{Z}$, what is the meaning of $U + V$? A natural choice is the *pointwise* lifting of the interpretations we had on $\mathbb{Z}$, which yields for instance $\{k, l\} + \{m, n\} = \{k + m, k + n, l + m, l + n\}$, $-\{m, n\} = \{-m, -n\}$ and $0 = \{0\}$. Once we have redefined the operations in this natural way, we may verify the validity of the ring equations on $\mathcal{P}\mathbb{Z}$. We see for instance that the equality $x + (-x) = 0$, which held

on $\mathbb{Z}$, does not hold on subsets:

$$\begin{aligned}\{2,3\} + (-\{2,3\}) &= \{2,3\} + \{-2,-3\} \\ &= \{2+(-2), 2+(-3), 3+(-2), 3+(-3)\} \\ &= \{-1,1\} \neq \{0\}\end{aligned}$$

On the contrary, the law of commutativity is preserved by $\mathcal{P}$ as for $U, V \subseteq \mathbb{Z}$ we have:

$$\begin{aligned}U + V &= \{u + v \mid u \in U, v \in V\} \\ &= \{v + u \mid v \in V, u \in U\} \\ &= V + U\end{aligned}$$

The example of $\mathcal{P}$ and $\mathbb{Z}$ clearly shows that a monadic transformation does not automatically preserve algebraic constructs. In this chapter, we first recall some results from the literature; then we focus on monoidal monads and show that their structure allows for a canonical approach to this problem. Finally, we show that the preservation by T of the algebraic features of S actually results in a distributive law, therefore permitting to compose T and S into a monad TS.

3.1 Preservation of Algebraic Features

3.1.1 Preservation in the case of $\mathcal{P}$

We discussed above the example of the powerset monad and a few cases of (non-) preservation. A more general study of this instance has been done by Gautam in [11], a paper much prior to any category theory. The author defines the construction of a *complex algebra*, which describes exactly the application of $\mathcal{P}$ to an arbitrary algebra $\mathcal{A}$. For any n-ary operation $\bullet$ of $\mathcal{A}$ (with carrier A), Gautam defines a new version of it, this time on subsets of A. We will write $\widehat{\bullet}$ this *complex* version of $\bullet$. As we did above, the author characterises it as the pointwise application of $\bullet$ on elements of the subsets $U_1, \ldots, U_n \subseteq A$.

$$\begin{aligned}A^n &\to A \\ \widehat{\bullet}\colon (U_1, \ldots, U_n) &\mapsto \{\bullet(u_1, \ldots, u_n) \mid u_1 \in U_1, \ldots, u_n \in U_n\}\end{aligned}$$

This construction turns all operations on A into operations on $\mathcal{P}A$, thus constructing a Σ-algebra structure on the latter. Gautam calls this new algebra the *complex* version of $\mathcal{A}$. By combining the complex versions of all operation symbols, Σ-terms can now be interpreted on subsets rather than on elements of A, and we can then compare them to verify if equalities of $\mathcal{A}$ still hold. This defines a first notion of preservation for the powerset monad: given an equation $u = v$, we say that it is *preserved by* $\mathcal{P}$ if for every algebra $\mathcal{A}$ where $u = v$ holds, $u = v$ also holds on the complex version of $\mathcal{A}$.

Gautam then proceeds to categorise equations depending on their preservation by $\mathcal{P}$. His criteria for this classification will turn out to be crucial in our work, as they place the focus on the layout of the variables in the equation. He shows that for an equality to be preserved by $\mathcal{P}$, only one thing matters: how many times each variable appears on each side of the equation.

Theorem 3.1 ([11])**.** *Let u and v be distinct terms. The equation $u = v$ is preserved by $\mathcal{P}$ if and only if each variable appears exactly once in u and once in v.*

We have seen above that the law of *inverse* was not preserved by $\mathcal{P}$, because it is not preserved in the case of $\mathbb{Z}$. This actually follows from Gautam's theorem, as the variable layout in the equation $x + (-x) = 0$ is incompatible with $\mathcal{P}$: x appears twice on the left side, and does not appear at all on the right side. Other laws that cannot be preserved by $\mathcal{P}$ include idempotence ($x * x = x$) because it contains a duplicated variable, and absorption ($x \times 0 = 0$) because x is missing on the right side. In contrast, the law of commutativity $x + y = y + x$ verifies Gautam's conditions, confirming our example above: it is preserved by $\mathcal{P}$ on any algebra. The question we want to address in this chapter is: can we adapt this approach to monads other than the powerset?

In the rest of this chapter we formalise Gautam's lifting of algebraic features in categorical terms and for an arbitrary monoidal monad; the following two chapters will focus on generalising the ideas of Theorem 3.1 to classify diverse classes of monads and equations.

3.2 Lifting the Signature

Let us now consider the general case of a monad T, and let Σ be a signature. In this section, we define a way to reinterpret algebraic symbols after the application of T. In categorical terms, we must *lift* the structure of Σ-algebra. We show here that if T is monoidal (2.25), there exists a canonical approach to this question: for any Σ-algebra $\mathcal{A}$ with carrier A, we can use the map $\psi\colon TA \times TA \to T(A \times A)$ to provide the set TA with the same algebraic structure as $\mathcal{A}$. Ultimately, we proceed to lift T to the category $\mathbf{Alg}(\Sigma)$. This canonical process is the main reason for which we focus our work on monoidal monads.

To illustrate the general idea, suppose Σ contains a binary operation $\bullet$. Let $\mathcal{A}$ be a Σ-algebra with carrier A; it is provided with a morphism interpreting $\bullet$ as $\bullet_{\mathcal{A}}\colon A \times A \to A$. Since T is monoidal, using the associated natural transformation ψ we can define a binary operation $\bullet_{\widehat{T}\mathcal{A}}$ on TA as follows.

$$\bullet_{\widehat{T}\mathcal{A}} \;\equiv\; \left(TA \times TA \xrightarrow{\psi_{A,A}} T(A \times A) \xrightarrow{T\bullet_{\mathcal{A}}} TA \right) \tag{3.1}$$

The meaning of our notation $\bullet_{\widehat{T}\mathcal{A}}$ will become clear in the following sections of this chapter, as our method will result in the lifting $\widehat{T}$ on the category $\mathbf{Alg}(\Sigma)$.

Example 3.2. Gautam's construction happens to be an instance of our monoidal approach. Let $\mathcal{A}$ be an algebra with carrier A and signature Σ. In the case of $\mathcal{P}$ and for $U, V \in \mathcal{P}A$, recall that $\psi(U, V) = U \times V$, and that $\mathcal{P}$ acts pointwise on morphisms. Thus any binary operation $+ \in \Sigma$ can be lifted in the following manner.

$$\begin{aligned}(U +_{\widehat{\mathcal{P}}\mathcal{A}} V) &= \mathcal{P}(+_{\mathcal{A}}) \circ \psi_{A,A}(U, V) \\ &= \mathcal{P}(+_{\mathcal{A}})(U \times V) \\ &= \{u +_{\mathcal{A}} v \mid u \in U, v \in V\}\end{aligned}$$

The lifting of operations by the multiset monad $\mathcal{M}$ can be written in the same way, keeping in mind that elements may have duplicates. We now show what happens in the cases of the distribution monad and the $\mathcal{R}_C$ monad. △

Example 3.3. Consider the distribution monad $\mathcal{D}$ and an algebra $\mathcal{A}$ with carrier A and a binary operation $*$. Recall that in the case of $\mathcal{D}$, the monoidal map is given by the measure product.

$$\begin{aligned}\mathcal{D}A \times \mathcal{D}B &\to \mathcal{D}(A \times B) \\ \psi_{A,B} \colon (\delta_1, \delta_2) &\mapsto \delta_1 \times \delta_2\end{aligned}$$

This allows to lift $*_{\mathcal{A}}$ to a binary operation on measures: for $\delta_1, \delta_2 \in \mathcal{D}A$, we have for $a \in A$:

$$\begin{aligned}(\delta_1 *_{\widehat{\mathcal{D}}\mathcal{A}} \delta_2)(a) &= (\mathcal{D}(*_{\mathcal{A}}) \circ \psi_{A,A}(\delta_1, \delta_2))(a) \\ &= \delta_1 \times \delta_2\{(u, v) \mid u, v \in A, u *_{\mathcal{A}} v = a\}\end{aligned}$$

△

Example 3.4. Consider the monad $\mathcal{R}_C$. The function $\psi_{A,B}$ maps the pair of $f \in A^C, g \in B^C$ to $x \mapsto (f(x), g(x))$. Again, we consider a Σ-algebra $\mathcal{A}$ with carrier A that features a binary operation $+$ and define its lifting in the following manner. Let $f, g \in \mathcal{R}_C(A) = A^C$

and let c be an element of c.

$$\begin{aligned}(f +_{\widehat{\mathcal{R}_C}\mathcal{A}} g)(c) &= (\mathcal{R}_C(+_{\mathcal{A}}) \circ \psi_{A,A}(f,g))(c) \\ &= f(c) +_{\mathcal{A}} g(c)\end{aligned}$$

Note that $\psi^1 = \mathrm{id}$ and $\psi^0 = \eta_1$, which allows us to define a lifting for constants and unary operations. Let 0 and $-$ respectively be such symbols of Σ. Then we have:

$$\begin{aligned}0_{\widehat{\mathcal{R}_C}\mathcal{A}}(c) &= (\mathcal{R}_C(0_{\mathcal{A}}) \circ \eta_1)(c) \\ &= 0_{\mathcal{A}}\end{aligned}$$

As $\mathcal{R}_C(0_{\mathcal{A}})$ constructs the constant function $0_{\mathcal{A}}$. Finally, as $\psi^1 = \mathrm{id}$, unary operations have a pointwise lifting. For instance the unary operator $-$ is lifted as follows:

$$(-_{\widehat{\mathcal{R}_C}\mathcal{A}} f)(c) = -_{\mathcal{A}} f(c)$$

△

This conversion of algebraic interpretations can be done in a similar fashion for any n-ary operation $\sigma \in \Sigma$, therefore providing the set TA with interpretations of each symbol of the signature. This can be formalised in categorical terms using a distributive law $\lambda^{\Sigma} \colon \mathsf{H}_{\Sigma} T \to T\mathsf{H}_{\Sigma}$, as in [36]. To define it, first note that for any $\sigma \in \Sigma$ we have the map

$$(TX)^{|\sigma|} \xrightarrow{\psi^{|\sigma|}} TX^{|\sigma|} \xrightarrow{T\theta_\sigma} T \coprod_{\sigma \in \Sigma} X^{|\sigma|} \ ,$$

where θ_σ is the coproduct injection. The distributive law λ^{Σ} is obtained as the cotupling of these maps:

$$\lambda^{\Sigma}_X \ \equiv \ \left(\mathsf{H}_{\Sigma} TX = \coprod_{\sigma \in \Sigma} (TX)^{|\sigma|} \xrightarrow{\left[T\theta_\sigma \circ \psi^{|\sigma|}\right]_{\sigma \in \Sigma}} T\mathsf{H}_{\Sigma} X \right). \tag{3.2}$$

This construction results in the following well-known Theorem.

Theorem 3.5 ([36])**.** *Let $T \colon \mathbf{Set} \to \mathbf{Set}$ be a monoidal monad, then for any finitary signature Σ, there exists a distributive law $\lambda^{\Sigma} \colon \mathsf{H}_{\Sigma} T \to T\mathsf{H}_{\Sigma}$ of the polynomial functor associated with Σ over the monad T.*

The distributive law (3.2) forms the core of our construction. First, we use it to construct a lifting of the functor T.

Definition 3.6. We denote by $\widehat{T}$ the functorial lifting of T to $\mathbf{Alg}(\mathsf{H}_\Sigma) = \mathbf{Alg}(\Sigma)$ obtained from λ^Σ via Theorem 2.66.

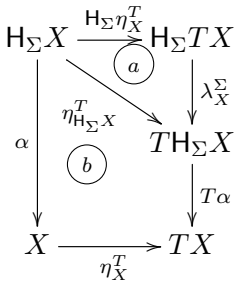
This lifting operates the following transformation on H_Σ-algebras:

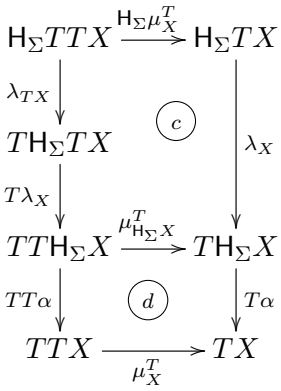
$$\widehat{T}\colon \quad \left(\mathsf{H}_\Sigma X \xrightarrow[\alpha]{} X \right) \quad \mapsto \quad \left(\mathsf{H}_\Sigma TX \xrightarrow[\lambda^\Sigma]{} T\mathsf{H}_\Sigma X \xrightarrow[T\alpha]{} TX \right)$$

From the monadic structure of (T, η^T, μ^T), we construct the liftings $\widehat{\eta}, \widehat{\mu}$ of respective type $\widehat{\eta}\colon I \to \widehat{T}$ and $\widehat{\mu}\colon \widehat{T}\widehat{T} \to \widehat{T}$ and show that we obtain a monad on the category of Σ-algebras.

Theorem 3.7. *$(\widehat{T}, \widehat{\eta}, \widehat{\mu})$ is a monad on $\mathbf{Alg}(\Sigma)$.*

Proof. We show that for any algebra (X, α) of $\mathbf{Alg}(\Sigma)$, η^T_X and μ^T_X are algebra homomorphisms. Their naturality follows immediately from the naturality of η^T, μ^T. Let (X, α) be an arbitrary algebra with $\alpha\colon \mathsf{H}_\Sigma X \to X$.





(3.3)

(a) and (c) commute because they correspond to the distributive law properties (DL. 1) and (DL. 3) of λ^Σ; (b) and (d) commute by naturality of η^T and μ^T.

The monad laws for η^T and μ^T immediately imply the same properties for their liftings $\widehat{\eta}$ and $\widehat{\mu}$. □

3.3 Preserving Equations

Now that we can lift T to $\widehat{T}$, we can also interpret equations after the application of the monad: if $t_1 = t_2$ holds on a Σ-algebra $\mathcal{A}$, we can now study the validity of this equation

on $\widehat{T}\mathcal{A}$. This leads naturally to the central notion of *preservation* of equations, that we define here in a general setting.

Definition 3.8. Let Σ be a signature, V a set of variables, and $t_1, t_2 \in \mathsf{F}_\Sigma V$. We say that the monad T preserves the equation $t_1 = t_2$ if for every Σ-algebra $\mathcal{A}$ we have:

$$\mathcal{A} \models t_1 = t_2 \quad \Rightarrow \quad \widehat{T}\mathcal{A} \models t_1 = t_2\,. \tag{3.4}$$

A set of equations is preserved if each one of them is.

Remark 3.9. In the case of $\mathcal{P}$, this notion of preservation corresponds exactly to Gautam's concept of 'validity of complex equations' presented at the beginning of this chapter. Because Gautam's lifting is a special case of our construction $\widehat{T}$ (see Example 3.2), the definition employed for Theorem 3.1 turns out to be an instance of Definition 3.8.

Example 3.10. Does the multiset monad preserve the commutativity of an operation? Let us consider an algebra $\mathcal{A}$ with carrier A featuring a commutative operation $\bullet$.

we can verify that for $U, V \in \mathcal{M}A$ we have:

$$\begin{aligned} U \bullet_{\widehat{\mathcal{M}}\mathcal{A}} V &= \{u \bullet_{\mathcal{A}} v \mid u \in U, v \in V\} \\ &= \{v \bullet_{\mathcal{A}} u, v \in V, u \in U\} && \text{by commutativity of } \bullet_A \\ &= V \bullet_{\widehat{\mathcal{M}}\mathcal{A}} U \end{aligned}$$

Therefore $\mathcal{M}$ preserves commutativity. △

Example 3.11. Consider an algebra $\mathcal{A}$ with carrier A and an idempotent operation $\bullet$ and the monad $\mathcal{D}$ of probability distributions. Let $a, b \in A$, such that $a \bullet b \neq a$, $a \bullet b \neq b$ and $a \bullet b \neq b \bullet a$. Let $\mathbb{P} \in \mathcal{D}\mathcal{A}$ be the distribution on A such that $\mathbb{P}(a) = \mathbb{P}(b) = 0.5$. Note that, for all $u, v \in A$, we have $\nu(u) \bullet \nu(v) \neq 0$ iff $u, v \in \{a, b\}$.

$$\begin{aligned} (\mathbb{P} \bullet_{\widehat{\mathcal{D}}\mathcal{A}} \mathbb{P})(a \bullet b) &= \sum\{(\nu \times \nu)(u, v) \mid u, v \in A \text{ s.t. } u \bullet v = a \bullet b\} \\ &= \nu(a) \bullet \nu(b) \;=\; 0.25 \;\neq\; 0 \;=\; \nu(a \bullet b) \end{aligned}$$

Thus we have $\nu \neq \nu \bullet_{\widehat{\mathcal{D}}\mathcal{A}} \nu$, which means that idempotence is not preserved by $\mathcal{D}$. △

Example 3.12. For a set C, consider the monad $\mathcal{R}_C$ mapping X to X^C and recall the liftings defined in Example 3.4. Does $\mathcal{R}_C$ preserve the law of inverse $x + (-x) = 0$? We consider an algebra $\mathcal{A}$ featuring this law, an element f of $\widehat{\mathcal{R}_C}\mathcal{A}$ and $c \in C$.

$$\begin{aligned}(f +_{\widehat{\mathcal{R}_C}\mathcal{A}} (-_{\widehat{\mathcal{R}_C}\mathcal{A}} f))(c) &= f(c) + (-f(c)) \\ &= 0_{\mathcal{A}}\end{aligned}$$

$(f +_{\widehat{\mathcal{R}_C}\mathcal{A}} (-_{\widehat{\mathcal{R}_C}\mathcal{A}} f))$ is a function that always maps to $0_{\mathcal{A}}$ regardless of its input, in other words it is the constant function $0_{\widehat{\mathcal{R}_C}\mathcal{A}}$. Hence $\mathcal{R}_C$ preserves the equation of inverse. △

It clearly appears from this series of examples that equations may or may not be preserved depending on the monad and on the equality itself. The literature only offers a few results on the matter; one is due to Gautam (Theorem 3.1), and a second one presented below generalises one direction of Gautam's result. Recall that we call an equation *linear* when every variable appears exactly once on each side (Definition 2.42). Theorem 3.1 shows that the class of linear equations is exactly what $\mathcal{P}$ preserves, and Manes and Mulry show with the more general result that all monoidal monads are able to lift such equalities.

Theorem 3.13 ([26])**.** *Let $T\colon \mathbf{Set} \to \mathbf{Set}$ be a monoidal monad. Then T preserves linear equations.*

Note that if an equation is *not* linear, then either there is a variable which occurs on one side but not the other (which we will later refer to as a *drop* equation, Definition 4.18) or there is a variable that occurs twice (referred to as a *dup* equation, Definition 4.19). Different questions naturally arise from this result: which monads preserve a dup equation such as idempotence? Which monads preserve the inverse law? We have seen in this section that the respective answers for $\mathcal{D}$ and $\mathcal{P}$ are negative, but that $\mathcal{R}_C$ does preserve inverse. These problems will be precisely addressed in the following chapters. But first, we focus on the case where a monad T successfully preserves all equations presenting S, and we show that our construction results then in a distributive law $ST \to TS$.

3.4 Completing the distributive law

Theorem 3.14. *Suppose $T, S\colon \mathbf{Set} \to \mathbf{Set}$ are monads, that T is monoidal and that $\mathcal{EM}(S) \simeq \mathbf{Alg}(\Sigma, E)$, and let $\widehat{T}\colon \mathbf{Alg}(\Sigma) \to \mathbf{Alg}(\Sigma)$ be the unique lifting of T defined via Theorem 3.5. If $\widehat{T}$ preserves the equations of E, then there exists a distributive law $\lambda\colon ST \to TS$ of (S, η^S, μ^S) over (T, η^T, μ^T).*

Proof. We assume that the equations in E are preserved by $\widehat{T}$, hence the monad $(\widehat{T}, \widehat{\eta}, \widehat{\mu})$ restricts to $\mathbf{Alg}(\Sigma, E) = \mathcal{EM}(S)$. By Theorem 2.66, this results in a distributive law λ of (S, η^S, μ^S) over T. We show that λ is actually a distributive law of (S, η^S, μ^S) over (T, η, μ). We must then show that (DL. 1) and (DL. 3) commute.

First, note that $\mu^T_S \circ T\lambda \circ \lambda_T$ and $\lambda \circ S\mu^T$ are algebra morphisms of type $STT \to TS$; moreover η^T_S and $\lambda \circ S\eta^T$ are algebra morphisms of type $S \to TS$. Finally, we remark that for a set X, both SX and TSX have a structure of S-algebra, moreover SX is the free S-algebra. Hence we can make use of Lemma 2.60 to show that η^T_S and $\lambda \circ S\eta^T$ must be equal by uniqueness of the algebra morphism of type $S \to TS$. For this purpose, we precompose $\mu^T_S \circ T\lambda \circ \lambda_T$ and $\lambda \circ S\mu^T$ with η^S and show that they are equalised:

$$\begin{aligned}
\mu^T_S \circ T\lambda \circ \lambda \circ \eta^S_{TT} &= \mu^T_S \circ T\lambda \circ T\eta^S_T && \text{by (DL. 2)} \\
&= \mu^T_S \circ TT\eta^S && \text{by (DL. 2)} \\
&= T\eta^S \circ \mu^T && \text{by naturality of } \mu^T \\
&= \lambda \circ \eta^S_T \circ \mu^T && \text{by (DL. 2)} \\
&= \lambda \circ S\mu^T \circ \eta^S_{TT} && \text{by naturality of } \eta_S
\end{aligned}$$

We operate the same reasoning on η^T_S and $\lambda \circ S\eta^T$:

$$\begin{aligned}
\eta^T_S \circ \eta^S &= T\eta^S \circ \eta^T && \text{by naturality of } \eta^T \\
&= \lambda \circ \eta^S_T \circ \eta^T && \text{by (DL. 2)} \\
&= \lambda \circ S\eta^T \circ \eta^S && \text{by naturality of } \eta^S
\end{aligned}$$

Therefore by the property of free algebras expressed in Lemma 2.60, we obtain the

following equalities:

$$\mu_S^T \circ T\lambda \circ \lambda_T = \lambda \circ S\mu^T \tag{3.5}$$

$$\eta_S^T = \lambda \circ S\eta^T \tag{3.6}$$

Hence the diagrams (DL. 3) and (DL. 1) commute, concluding our proof that λ is a distributive law of (S, η^S, μ^S) over (T, η^T, μ^T). □

This theorem is the final piece of our construction, which we briefly recall: first, the structure of monoidal monads allowed to canonically lift a set monad T to the category of Σ-algebras (3.2). We then applied this lifting $\widehat{T}$ to any algebra $\mathcal{A}$ and verified if the equalities between Σ-terms that are valid in $\mathcal{A}$ are preserved by this transformation (3.3). If they are preserved for all algebra $\mathcal{A}$, and if (Σ, E) presents the monad S, then we now obtain a distributive law of S over T (3.4). Searching for distributive laws between monads is a notoriously difficult task; We now have a general tool to address the problem when our conditions are met, allowing to give a positive answer in many cases.

Example 3.15. Consider the monads $\mathcal{P}$ and $\mathcal{L}$. Recall that $\mathcal{L}$ is presented by the theory of monoids; we have seen at the beginning of this chapter how $\mathcal{P}$ lifts binary operations and constants. Furthermore, Theorem 3.1 shows that the equations of monoids (2.45) are all preserved by $\mathcal{P}$ because they are linear. Therefore by Theorem 3.14, we obtain a distributive law $\mathcal{L}\mathcal{P} \to \mathcal{P}\mathcal{L}$ and we may construct the composite monad $\mathcal{P}\mathcal{L}$. Note that the distributive law that we obtain is a well-known transformation, namely the one presented in 2.3.3. △

Example 3.16. We have seen in example 3.12 that $\mathcal{R}_C$ preserves the law of inverse. By Theorem 3.13, it also preserves the equations of unit, commutativity and associativity because it is monoidal (2.28). Therefore $\mathcal{R}_C$ preserves the theory of Abelian groups (described in example 2.46), hence by Theorem 3.14 there exists a distributive law $\mathcal{A}\mathcal{R}_C \to \mathcal{R}_C\mathcal{A}$ between $\mathcal{R}_C$ and the monad $\mathcal{A}$ of free Abelian groups. △

Note that we have not yet explicitly described the action of the distributive law constructed in Theorem 3.14. This will be the object of chapter 7, where we examine precisely the algebraic interactions produced by this law.

We now have a procedure to compose monads when their features are compatible. However, the general question of whether a given monad preserves an equation remains unclear. Examples 3.10 to 3.12 show that the outcome depends on the monad and the

equality considered. Linear equations do not constitute a problem (Theorem 3.13), but if variables are duplicated or dropped from one side to the other, preservation is not guaranteed. In the next two chapters, we carry out a precise study of this question and respectively provide sufficient and necessary conditions for preserving equations.

3.5 Related Work

The general question of building a distributive law between two monads S and T has proven to be difficult. The mere existence of such a law is already a tricky subject, as shows the history of this area of category theory. Consider for example the aforementioned case of composing the powerset monad $\mathcal{P}$ with itself. An attempt is made in 2007 in [26], where Manes and Mulry offer the following candidate for a distributive law (we write $\mathcal{P}'$ to differentiate one instance of $\mathcal{P}$):

$$\begin{aligned} &\mathcal{P}\mathcal{P}'X \to \mathcal{P}'\mathcal{P}X \\ &\lambda_X : \mathcal{A} \mapsto \{\{a_A : A \in \mathcal{A}\} : (a_A) \in \prod_{A\in\mathcal{A}} A\} \end{aligned}$$

In a nutshell, this transformation is similar to the distributive law we defined in 2.3.3 it acts on a set of sets $\mathcal{A}$ by selecting an element in each set A. However, this construct does not satisfy axiom (DL. 3), therefore λ is not a distributive law. For $X = \{a, b, c\}$, $\mathcal{A} = \{\{\{a\}, \{b, c\}\}\} \in \mathcal{P}\mathcal{P}'\mathcal{P}'X$ gives a counterexample as follows:

$$\begin{aligned} \mu^{\mathcal{P}'}_{\mathcal{P}X} \circ \mathcal{P}'\lambda_X \circ \lambda_{\mathcal{P}'X}(\mathcal{A}) &= \{\{a, b\}, \{a, c\}\} \\ \lambda_X \circ \mathcal{P}\mu^{\mathcal{P}'}_X(\mathcal{A}) &= \{\{a\}, \{b\}, \{c\}\} \end{aligned}$$

Manes and Mulry themselves acknowledged this mistake in their follow-up paper. In 2015, Klin and Rot made a similar claim [20], but recently Klin and Salamanca have in fact shown that there is no distributive law of $\mathcal{P}$ over itself. They explain carefully why mistakes in the previous results were so subtle and hard to spot [21]. More recently, Zwart and Marsden generalised the ideas behind [21] to prove the non-existence of distributive laws in a wide variety of cases, settling many questions left open until now.

As this problem seems to be very technical and sometimes counter-intuitive, our contribution is a general method clearing up many cases with positive results, and exploiting the structure of monoidal monads. A few different approaches for constructing distributive laws have been discussed in the literature: first, King and Wadler examine how to construct one in some particular cases in [19]. Later, Manes and Mulry present a first a general theorem in [26]: commutative monads compose with monads presented by a linear theory via a distributive law λ. This result comes complete with the constructive definition of λ, using a method very similar to ours. However, our work extends beyond the linear case to place the focus on the idea of preservation of algebraic features. As we will see in the next chapter, we generalise Manes and Mulry's result to several classes of

non-linear theories and the monads preserving them. The work of Eugenia Cheng was also a source of inspiration, particularly [8] where she studies how several distributive laws may be consistently combined to form a the composite of multiple monads.

Our work connects distributive laws with algebraic presentations, in a very similar manner to the study undertaken by Bonsague, Hansen, Kurz and Rot in [5]. In a nutshell, their paper establishes a method to 'quotient' a distributive law in order to accomodate the equations presenting a monad, a process which relates closely to our preservation of equations. As we mentioned before, a pioneering work on algebraic preservation was done by Gautam in [11], and extended by Grätzer in [12]. Although Gautam's work was done in a purely mathematical context and prior to category theory, his construction corresponds precisely to the case of the powerset monad in our monoidal setting. Our work then generalises his notion of 'complex algebras' to a lifting of algebraic structures by an arbitrary monoidal monad. The next chapter explores more similarities with Gautam's work with our classification of equations.

Finally, it is worth mentioning that Manes and Mulry's second paper on monad composition [27] follows an approach comparable to ours. Making use of the monadic concept of 'Kleisli strength', very similar to a monoidal structure, the authors build a step-by-step lifting of algebraic structures. Zwart and Marsden later established that Manes and Mulry's attempt at distributing the list monad over itself results in a faulty transformation (as in fact there exists no distributive law $\mathcal{L}\mathcal{L} \to \mathcal{L}\mathcal{L}$), confirming once again the difficulty of this topic.

Chapter 4

Preservation of Equations

In the current chapter, we establish sufficient conditions for equations to be preserved by a monoidal monad T. For this purpose, we must precisely study the interactions between the monad and the algebraic terms of the equation. We have seen in Chapter 3 (Theorem 3.1) the importance of counting the variable occurences. Although this result focuses on $\mathcal{P}$, we will show here that Gautam's sufficient conditions can be generalised to different classes of monoidal monads. First, we examine the mechanisms underlying the preservation of an equation, and we define a decomposition of algebraic terms focusing on variable rearrangements. Finally, we will show how this framework leads to a series of sufficient conditions for preservation for an arbitrary monoidal monad T.

4.1 New Representations for Equations

We start with some notation used in this chapter. We fix a finite set V of variables and a bijection $b\colon V \to \{1, \dots, n\}$, where $|V| = n$ is the size of V. We will refer to b as the *enumeration* of our variables, allowing us to order them as $x_1, x_2, \dots, x_n$ when $b(x_i) = i$ for every i. We will sometimes write variables x, y, z when studying an example. If t is an algebraic term, we will denote by $\mathrm{Var}(t)$ the *set of variables* in t and by $\mathrm{Arg}(t)$ the *list of arguments* used in t ordered as they appear in t. For example, the list of arguments of $t = f(x_1, g(x_3, x_2), x_1)$ is $\mathrm{Arg}(t) = [x_1, x_3, x_2, x_1]$. We consider an equation $t_1 = t_2$ between two Σ-terms with variables in V, and a Σ-algebra $\mathcal{A}$ with carrier A.

We interpret algebraic terms by decomposing them into two transformations: consider for instance the term $x \times x$. Its interpretation on $\mathcal{A}$ can be intuitively described in two steps. First, we pick an element a of the algebra to substitute to the variable x, and we duplicate it; then we apply the operation $\times_{\mathcal{A}}$ to both copies.

In the general case, the first part of the interpretation is a morphism with domain $A^{|V|}$, the Cartesian product of $|V|$-times A; it picks an element of A for each variable

(we will call these elements the *inputs* of t). Then we rearrange, copy and duplicate the inputs to match the layout of the arguments of t.

Definition 4.1. For a term t with $k = |\mathrm{Arg}(t)|$, we define the morphism $\delta^V_{\mathcal{A}}(t)\colon A^{|V|} \to A^k$ as the following pairing of projections:

$$\text{if } \mathrm{Arg}(t) = [x_{i_1}, x_{i_2}, x_{i_3}, \dots x_{i_k}] \text{ then } \delta^V_{\mathcal{A}}(t) = \langle \pi_{i_1}, \pi_{i_2}, \pi_{i_3}, \dots \pi_{i_k} \rangle$$

If $k = 0$, $\delta^V_{\mathcal{A}}(t)$ is the unique map ! to the final object 1. Note that this definition depends on our enumeration of variables b. We will see later that this dependency does not affect our results on preservation of equations.

Example 4.2. Consider the term $t = (x_1 + (x_2 \bullet x_1)) + x_3$, with $V = \{x_1, x_2, x_3\}$. Then we have $\delta^V_{\mathcal{A}}(t) = \langle \pi_1, \pi_2, \pi_1, \pi_3 \rangle\colon A^3 \to A^4$. △

Once the inputs have been correctly rearanged, we can focus on the algebraic operations of t. For each symbol of the signature, we refer to its interpretation in $\mathcal{A}$ and apply it in the natural way.

Definition 4.3. For a term t with $k = |\mathrm{Arg}(t)|$, $\gamma^V_{\mathcal{A}}(t)\colon A^k \to A$ is defined inductively:

$$\gamma^V_{\mathcal{A}}(x) = \mathrm{id}_A$$
$$\gamma^V_{\mathcal{A}}(\sigma(t_1, \dots, t_n)) = A^k \xrightarrow{\gamma^V_{\mathcal{A}}(t_1) \times \dots \times \gamma^V_{\mathcal{A}}(t_n)} A^i \xrightarrow{\sigma_{\mathcal{A}}} A$$

We write $\sigma_{\mathcal{A}}$ the interpretation of $\sigma \in \Sigma$ in $\mathcal{A}$.

The product $\gamma^V_{\mathcal{A}}(t_1) \times \dots \times \gamma^V_{\mathcal{A}}(t_i)$ has the desired type, because we have

$$|\mathrm{Arg}(t_1)| + \dots + |\mathrm{Arg}(t_n)| = k = |\mathrm{Arg}(t)|$$

Note that the case of a constant term $t = c$ also follows from the definition above, namely $\gamma^V_{\mathcal{A}}(c) = 1 \xrightarrow{c_{\mathcal{A}}} A$.

Example 4.4. Again for the term $t = (x_1 + (x_2 \bullet x_1)) + x_3$, we have $\gamma^V_{\mathcal{A}}(t) = +_{\mathcal{A}} \circ (+_{\mathcal{A}} \times \mathrm{id}) \circ (\mathrm{id} \times \bullet_{\mathcal{A}} \times \mathrm{id})\colon A^4 \to A$. △

Finally we combine the two transformations and define $[\![t]\!]^V_{\mathcal{A}}$ as $\gamma^V_{\mathcal{A}}(t) \circ \delta^V_{\mathcal{A}}(t)$. The following lemma follows easily from the definitions.

Lemma 4.5. *For any $t \in \mathsf{F}_\Sigma V$, $\delta^V_{\mathcal{A}}(t), \gamma^V_{\mathcal{A}}(t)$, and thus $[\![t]\!]^V_{\mathcal{A}}$, are natural in $\mathcal{A}$.*

We are now provided with a natural transformation $[\![t]\!]^V : (-)^{(|V|)}\mathsf{U}_\Sigma \to \mathsf{U}_\Sigma$ interpreting $t \in \mathsf{F}_\Sigma V$. We aim to use this interpretation to reason on equations, therefore we must show that the natural soundness and completeness property holds. First, we show that $\gamma^V_{\mathcal{A}}(t)$ is connected with our previous valuation-based interpretation $[\![t]\!]_v$ (defined in 2.53).

Lemma 4.6. *Let t be a term with variables in V, for which we write $\mathrm{Arg}(t) = [x_{i_1} \dots x_{i_k}]$. Let v be a valuation $V \to A$. We write $a_i = v(x_i)$ for all $i \leq |V|$, with $a_1, \dots, a_{|V|} \in A$.*

If $|\mathrm{Arg}(t)| > 0$, then we have:

$$\gamma^V_{\mathcal{A}}(t)(a_{i_1}, \dots, a_{i_k}) = [\![t]\!]_v$$

Similarly, if $|\mathrm{Arg}(t)| = 0$ we have $\gamma^V_{\mathcal{A}}(t)(*) = [\![t]\!]_v$ where $*$ represents the unique element of 1.

Proof. The case $|\mathrm{Arg}(t)| = 0$ is treated easily: since t contains no variable, $[\![t]\!]_v$ does not depend of v and is simply the combined interpretation in the algebra $\mathcal{A}$ of all symbols used in t. By definition of $\gamma^V_{\mathcal{A}}$, this corresponds to $\gamma^V_{\mathcal{A}}(t)(*)$.

If $|\mathrm{Arg}(t)| = k > 1$, we proceed by induction on t.

- if $t = x_{i_1}$, then $\gamma^V_{\mathcal{A}}(t) = \mathrm{id}$, hence we have

$$\gamma^V_{\mathcal{A}}(t)(a_{i_1}) = a_{i_1} = v(x_{i_1}) = [\![t]\!]_v$$

- For the case $t = \sigma(t_1, \dots, t_n)$, we examine closely the list of arguments. Because $\mathrm{Arg}(t)$ sequentially creates a list of all the variables appearing in the term, we have $\mathrm{Arg}(t) = \mathrm{Arg}(t_1) \colon \dots \colon \mathrm{Arg}(t_n)$, where the sign $:$ denotes the concatenation of lists. More precisely, there exist integers $p_1, p_2 \dots p_n$ and we have:

$$[x_{i_1} \dots x_{i_k}] = [x_{i_1} \dots x_{p_1}] \colon [x_{(p_1+1)} \dots x_{p_2}] \colon \dots \colon [x_{(p_{n-1}+1)} \dots x_{p_n}]$$

  Where $p_n = i_k$. Moreover, for all $m \leq n$ we have $[x_{(p_{m-1}+1)} \dots x_{p_m}] = \mathrm{Arg}(t_m)$. Note that the case of a constant subterm, in other words the case $\mathrm{Arg}(t_i)$ being

an empty list, is covered by taking $p_i = p_{i-1}$. With this subdivision in mind, the definition of $\gamma^V_{\mathcal{A}}$ yields

$$\gamma^V_{\mathcal{A}}(t)(a_{i_1}, \ldots, a_{i_k}) = \sigma_{\mathcal{A}}(\gamma^V_{\mathcal{A}}(t_1)(a_{i_1}, \ldots, a_{p_1}), \ldots, \gamma^V_{\mathcal{A}}(t_n)(a_{(p_{n-1}+1)}, \ldots, a_{p_n}))$$

Then we apply the induction hypothesis and obtain:

$$\begin{aligned}\gamma^V_{\mathcal{A}}(t)(a_{i_1}, \ldots, a_{i_k}) &= \sigma_{\mathcal{A}}([\![t_1]\!]_v, \ldots, [\![t_n]\!]_v) \\ &= [\![t]\!]_v\end{aligned}$$

□

With the help of this technical result, we can show soundness and completeness of our new interpretation by combining $\delta^V_{\mathcal{A}}$ and $\gamma^V_{\mathcal{A}}$.

Theorem 4.7. *For $\mathcal{A}$ a Σ-algebra and $t_1, t_2 \in \mathsf{F}_\Sigma V$, $[\![t_1]\!]^V_{\mathcal{A}} = [\![t_2]\!]^V_{\mathcal{A}}$ iff $\mathcal{A} \models t_1 = t_2$.*

Proof. Let A be a Σ-algebra and $t_1, t_2 \in \mathsf{F}_\Sigma V$, let $a_1, \ldots\ a_{|V|} \in A$.

We first consider t_1 and write $\mathrm{Arg}(t_1) = [x_{i_1}, \ldots x_{i_1}]$ as previously. By definition of $\delta^V_{\mathcal{A}}$, we have $\delta^V_{\mathcal{A}}(u)(a_1, \ldots, a_{|V|}) = (a_{i_1}, \ldots, a_{i_k})$; this function rearranges the inputs to match the arguments. We combine this information with the previous lemma and obtain that for any valuation v such that $v(x_i) = a_i$ for $1 \leq i \leq |V|$, we have

$$\gamma^V_{\mathcal{A}} \circ \delta^V_{\mathcal{A}}(t_1)(a_1, \ldots, a_{|V|}) = \gamma^V_{\mathcal{A}}(t_1)(a_{i_1}, \ldots, a_{i_k}) = [\![t_1]\!]_v$$

The same observation applies to t_2, hence we have:

$$\begin{aligned}[\![t_1]\!]^V_{\mathcal{A}} = [\![t_2]\!]^V_{\mathcal{A}} &\Leftrightarrow \gamma^V_{\mathcal{A}} \circ \delta^V_{\mathcal{A}}(t_1) = \gamma^V_{\mathcal{A}} \circ \delta^V_{\mathcal{A}}(t_2) \\ &\Leftrightarrow \forall (a_1, \ldots, a_{|V|}) \in A^{|V|}, \\ &\qquad \gamma^V_{\mathcal{A}} \circ \delta^V_{\mathcal{A}}(t_1)(a_1, \ldots, a_{|V|}) = \gamma^V_{\mathcal{A}} \circ \delta^V_{\mathcal{A}}(t_2)(a_1, \ldots, a_{|V|}) \\ &\Leftrightarrow \forall v : V \to A, [\![t_1]\!]_v = [\![t_2]\!]_v \\ &\Leftrightarrow \mathcal{A} \models t_1 = t_2\end{aligned}$$

□

We now have a new method to interpret algebraic terms which exactly translates algebraic equalities, but also focuses on the layout of variables in terms. In order to study the preservation of an equation, we will compare two different interpretations of each term: the one on an algebra $\mathcal{A}$, and the one on the lifted algebra $\widehat{T}\mathcal{A}$.

4.2 Residual Diagrams

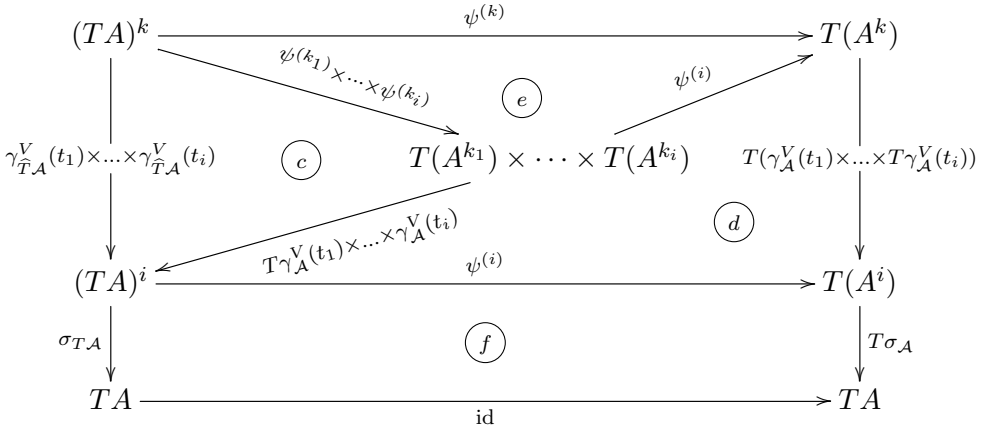
In this section, we define diagrams that we call *residual*; we construct them from an equation and they provide sufficient conditions for its preservation. Recall from Definition 3.8 that an equation $t_1 = t_2$ is preserved if $\mathcal{A} \models t_1 = t_2 \Rightarrow \widehat{T}\mathcal{A} \models t_1 = t_2$, or equivalently as we know now, if $[\![t_1]\!]^V_{\mathcal{A}} = [\![t_2]\!]^V_{\mathcal{A}} \Rightarrow [\![t_1]\!]^V_{\widehat{T}\mathcal{A}} = [\![t_2]\!]^V_{\widehat{T}\mathcal{A}}$. In this section, we show that our construction of $[\![t_1]\!]^V_{\widehat{T}\mathcal{A}}$ allows to reduce the question of preserving $t_1 = t_2$ to simple properties of the monad. As a first step, we prove a technical property relating $\gamma^V_{\widehat{T}\mathcal{A}}$ and $\gamma^V_{\mathcal{A}}$.

Lemma 4.8. *For any $\mathcal{A} \in \mathbf{Alg}(\Sigma)$, $t \in \mathsf{F}_\Sigma V$, $k = |\mathrm{Arg}(t)|$ we have:*

$$T\gamma^V_{\mathcal{A}}(t) \circ \psi^{(k)} = \gamma^V_{\widehat{T}\mathcal{A}}(t)$$

Proof. Recall that $\gamma^V_{\mathcal{A}}$ is defined inductively; therefore we proceed by induction on the term t.

- if t is a variable x, then $k = 1$, $\gamma^V_{\mathcal{A}}(t) = \gamma^V_{\widehat{T}\mathcal{A}}(t) = \psi^{(k)} = \mathrm{id}$ and the equality trivially holds.
- If $t = \sigma(t_1, \dots t_i)$, let $k_j = |\mathrm{Arg}(t_j)|$. We have $k = k_1 + \cdots + k_i$. We show the commutativity of the following diagram, which represents $T\gamma^V_{\mathcal{A}}(t) \circ \psi^{(k)} = \gamma^V_{\widehat{T}\mathcal{A}}(t)$.



(e) commutes by monoidality of ψ; (c) commutes by induction hypothesis for each term t_j; (d) commutes by naturality of ψ_i; (f) commutes by definition of our monoidal lifting of algebraic operations: $\sigma_{T\mathcal{A}} = T\sigma_{\mathcal{A}} \circ \psi_i$.

□

We now define an important original construct: the *residual diagrams*. Their commutativity will lead to a tailor-made sufficient condition for the preservation of an equation. First, we introduce a rather complex diagram representing the operation of our lifting $\widehat{T}$ on an algebraic term; we will then show that we only need to consider a simpler subdiagram.

Definition 4.9. For a term t, consider the diagram below on the category $\mathbf{Alg}(\Sigma)$. We call $\mathrm{Pres}(T, t, V)$ the outer square of Diagram (4.1); it is subdivided into two inner squares, and we call the top one (r) the *residual diagram* $\mathcal{R}(T, t, V)$.

$$\begin{array}{ccc}
(-)^{(|V|)}\mathsf{U}_\Sigma\widehat{T} & \xrightarrow{\psi^{|V|}_{\mathsf{U}_\Sigma}} & T(-)^{|V|}\mathsf{U}_\Sigma \\
\downarrow{\scriptstyle \delta^V_{\widehat{T}}(t)} & (r) & \downarrow{\scriptstyle T\delta^V(t)} \\
(-)^k\mathsf{U}_\Sigma\widehat{T} & \xrightarrow[\psi^{(k)}_{\mathsf{U}_\Sigma}]{} & T(-)^k\mathsf{U}_\Sigma \\
\downarrow{\scriptstyle \gamma^V_{\widehat{T}}(t)} & & \downarrow{\scriptstyle T\gamma^V(t)} \\
\mathsf{U}_\Sigma\widehat{T} & \xrightarrow[\mathsf{U}_\Sigma \mathrm{id}_{\widehat{T}}]{} & \mathsf{U}_\Sigma\widehat{T}
\end{array}
\qquad
\begin{array}{l}
\llbracket t \rrbracket^V_{\widehat{T}} \\ \\ \\ \\ \\
T\llbracket t \rrbracket^V
\end{array}
\tag{4.1}$$

Since $\mathsf{U}_\Sigma \circ \widehat{T} = T \circ \mathsf{U}_\Sigma$ by definition of $\widehat{T}$ being a lifting, it is clear that the horizontal arrows $\psi^{(|V|)}_{\mathsf{U}_\Sigma}$ and $\psi^{(k)}_{\mathsf{U}_\Sigma}$ are well-typed. Intuitively, $\mathrm{Pres}(T, t, V)$ represents the 'preservation' of the term t on its own by the monad T. Note that both $\mathcal{R}(T, t, V)$ and $\mathrm{Pres}(T, t, V)$ apply to objects on the category $\mathbf{Alg}(\Sigma)$. Throughout this section we will always study their commutativity by evaluating them on an arbitrary algebra $\mathcal{A}$ with carrier A, thus obtaining a diagram on **Set** instead. The following lemma shows that $\mathcal{R}(T, t, V)$ is the crucial part of diagram (4.1): the bottom square always commutes, whereas the top square (r) is not necessarily commuting.

Lemma 4.10. *If* $\mathcal{R}(T, t, V)$ *commutes, then* $\mathrm{Pres}(T, t, V)$ *commutes.*

Proof. We evaluate Diagram (4.1) on any arbitrary algebra $\mathcal{A}$. Then Lemma 4.8 immediately shows that the bottom square commutes. Therefore the outer diagram commutes

if $\mathcal{R}(T, t, V)$ commutes. □

Thus we do not need to consider the entirety of diagram (4.1): the residual part is enough. Now we can combine residual diagrams for t_1 and t_2 to obtain a condition under which $t_1 = t_2$ is preserved. The following soundness theorem follows easily from Lemma 4.10.

Theorem 4.11. *If $t_1, t_2 \in \mathsf{F}_\Sigma V$ are such that $\mathcal{R}(T, t_1, V)$ and $\mathcal{R}(T, t_2, V)$ commute, then T preserves $t_1 = t_2$.*

Proof. Again, we consider an arbitrary algebra $\mathcal{A}$. If $\mathcal{A} \models t_1 = t_2$, then $[\![t_1]\!]^V_{\mathcal{A}} = [\![t_2]\!]^V_{\mathcal{A}}$ by Theorem 4.7 and thus $T[\![t_1]\!]^V_{\mathcal{A}} \circ \psi_A^{(|V|)} = T[\![t_2]\!]^V_{\mathcal{A}} \circ \psi_A^{(|V|)}$. Since $\mathcal{R}(T, t_1, V)$ and $\mathcal{R}(T, t_2, V)$ commute, so do $\mathrm{Pres}(T, t_1, V)$ and $\mathrm{Pres}(T, t_2, V)$ by Lemma 4.10, which is to say by evaluating them on $\mathcal{A}$ that $T[\![t_1]\!]^V_{\mathcal{A}} \circ \psi_A^{(|V|)} = [\![t_2]\!]^V_{\widehat{T}\mathcal{A}}$ (and similarly for t_2). Therefore we have $[\![t_1]\!]^V_{\widehat{T}\mathcal{A}} = [\![t_2]\!]^V_{\widehat{T}\mathcal{A}}$, which means by Lemma 4.7 that $\widehat{T}\mathcal{A} \models t_1 = t_2$. □

We now see that residual diagrams act as sufficient conditions for equation preservation: we only have to study the commutativity of $\mathcal{R}(T, t_1, V)$ and $\mathcal{R}(T, t_2, V)$ to prove that $t_1 = t_2$ is preserved. Let us now present a few examples and show the practical consequences of our result.

Example 4.12. Consider the equation of commutativity $x+y = y+x$. By Theorem 4.11, it is preserved by T if the following diagrams commute.

$\mathcal{R}(T, x + y, \{x, y\})$ given by:

$$\begin{array}{ccc} (TA)^2 & \xrightarrow{\psi} & T(A^2) \\ {\scriptstyle \langle \pi_1, \pi_2 \rangle}\downarrow & & \downarrow{\scriptstyle T\langle \pi_1, \pi_2 \rangle} \\ (TA)^2 & \xrightarrow[\psi]{} & T(A^2) \end{array}$$

$\mathcal{R}(T, y + x, \{x, y\})$ given by:

$$\begin{array}{ccc} (TA)^2 & \xrightarrow{\psi} & T(A^2) \\ {\scriptstyle \langle \pi_2, \pi_1 \rangle}\downarrow & & \downarrow{\scriptstyle T\langle \pi_2, \pi_1 \rangle} \\ (TA)^2 & \xrightarrow[\psi]{} & T(A^2) \end{array}$$

(4.2)

We remark that in $\mathcal{R}(T, x + y, \{x, y\})$, the morphism $\langle \pi_1, \pi_2 \rangle$ actually is the identity. Therefore this diagram trivially commutes. On the other hand, $\mathcal{R}(T, x + y, \{x, y\})$ amounts to a non-trivial property. Because $\langle \pi_2, \pi_1 \rangle = \mathsf{swap}$, this diagram is identical to (SYM), which commutes if T is monoidal. Therefore both residual diagrams commute, and commutativity is always preserved by monoidal monads. △

Example 4.13. Consider the law of associativity of a binary operation $x \bullet (y \bullet z) = (x \bullet y) \bullet z$. It is preserved if the residual diagrams presented below commute.

$\mathcal{R}(T, x \bullet (y \bullet z), \{x, y, z\})$ given by:

$$\begin{array}{ccc} (TA)^3 & \xrightarrow{\psi^{(3)}} & T(A^3) \\ {\scriptstyle \langle \pi_1, \pi_2, \pi_3 \rangle}\downarrow & & \downarrow{\scriptstyle T\langle \pi_1, \pi_2, \pi_3 \rangle} \\ (TA)^3 & \xrightarrow[\psi^{(3)}]{} & T(A^3) \end{array}$$

$\mathcal{R}(T, (x \bullet y) \bullet z, \{x, y, z\})$ given by:

$$\begin{array}{ccc} (TA)^3 & \xrightarrow{\psi^{(3)}} & T(A^3) \\ {\scriptstyle \langle \pi_1, \pi_2, \pi_3 \rangle}\downarrow & & \downarrow{\scriptstyle T\langle \pi_1, \pi_2, \pi_3 \rangle} \\ (TA)^3 & \xrightarrow[\psi^{(3)}]{} & T(A^3) \end{array}$$

(4.3)

Note that both diagrams are identical: that is because both sides of the equation have the same layout of variables. Moreover, note that the morphisms $\langle \pi_1, \pi_2, \pi_3 \rangle \colon (TA)^3 \to (TA)^3$ and $\langle \pi_1, \pi_2, \pi_3 \rangle \colon A^3 \to A^3$ are actually both equal to id. Therefore our diagrams trivially commute, thus associativity is always preserved. △

The same reasoning apply to both left and right unit laws, with residual diagrams encoding trivial equations id = id. It follows that:

Theorem 4.14. *Monoidal monads preserve associativity, unit and commutativity.*

We have seen in the previous section, for instance in Example 3.12, that some equations are not systematically preserved by commutative monads; we now have precise conditions under which preservation succeeds.

Example 4.15. Consider the laws of idempotence $x \bullet x = x$ and absorption $x \bullet 0 = 0$. Once again, some of their residual diagrams are trivial: $\mathcal{R}(T, x, \{x\})$ and $\mathcal{R}(T, 0, \{x\})$ both amount to id = id. We are left with one diagram for each equality, displayed below.

Idempotency: $x \bullet x = x$

$\mathcal{R}(T, x \bullet x, \{x\})$ given by:

$$\begin{array}{ccc} TA & \xrightarrow{\mathrm{id}} & TA \\ {\scriptstyle \langle \pi_1, \pi_1 \rangle}\downarrow & & \downarrow{\scriptstyle T\langle \pi_1, \pi_1 \rangle} \\ (TA)^2 & \xrightarrow[\psi]{} & T(A^2) \end{array}$$

Absorption: $x \bullet 0 = 0$

$\mathcal{R}(T, x \bullet 0, \{x\})$ given by:

$$\begin{array}{ccc} TA & \xrightarrow{\mathrm{id}} & TA \\ {\scriptstyle !}\downarrow & & \downarrow{\scriptstyle T!} \\ 1 & \xrightarrow[\eta_1]{} & T1 \end{array}$$

(4.4)

Unlike the diagrams studied before, $\mathcal{R}(T, x \bullet x, \{x\})$ and $\mathcal{R}(T, x \bullet 0, \{x\})$ are not trivially commuting. In fact, we can show that they sometimes fail to commute.

Consider for instance the powerset $\mathcal{P}$, and an algebra $\mathcal{A}$ such that its carrier A has at least two elements. Now consider two distinct elements $a, b \in A$ and the set $\{a, b\} \in \mathcal{P}A$. The left branch of $\mathcal{R}(T, x \bullet x, \{x\})$ maps this set to $\{(a, a), (b, b)\}$ whereas the right branch maps it to $\{(a, a), (a, b), (b, a), (b, b)\}$. For the case of $\mathcal{R}(T, x \bullet 0, \{x\})$, consider the empty set $\emptyset \in \mathcal{P}A$. We have $T!(\emptyset) = \emptyset$, but on the contrary $\eta_1 \circ !(\emptyset) = \{*\} \neq \emptyset$, hence the diagram does not commute. △

When residual diagrams fail to commute, can we say that the considered equation is not preserved? In general we cannot, because the converse of Theorem 4.11 does not hold. We explain this below with the example of $\mathcal{P}$.

Example 4.16. Consider $\mathcal{P}$ and a trivial equation that holds in any algebra: $x \bullet x = x \bullet x$. It is clear that $\widehat{\mathcal{P}}\mathcal{A} \models x \bullet x = x \bullet x$ whenever $\mathcal{A} \models x \bullet x = x \bullet x$. By definition, it is preserved by $\mathcal{P}$. However $\mathcal{R}(\mathcal{P}, x \bullet x, \{x\})$ does not commute, because it is the same diagram as $\mathcal{R}(\mathcal{P}, x + x, \{x\})$ studied in Example 4.15. △

We remark that residual diagrams only involve ψ, projections and the monad T, sometimes inside pairings. It is meaningful to note that the actual operations of Σ appearing in an equation $t_1 = t_2$ have no impact on its preservation. What matters is the variable rearrangement transformations $\delta^V(t_1)$ and $\delta^V(t_2)$, and how they interact with the monoidal maps ψ.

The two equalities considered in Example 4.15 present particular features: either a duplicated variable ($x \bullet x = 0$) or a variable missing from one side ($x \bullet 0 = 0$). In the next section, we define equation classes corresponding to such features; then we will show that in each case, a simple residual diagram captures the preservation of the entire class.

4.3 Classes of Equations

For the purpose of preservation by a monad, it makes sense to classify equations depending on their variable layout. We have already mentioned *linear* equations, where each variable appears exactly once on each side (see Definition 2.42); we now recall two other classes found in the literature.

Definition 4.17.

- An equation is called *affine* if each variable appears at most once on each side.
- An equation is called *relevant* if each variable appears at least once on each side.

We will see in the following chapters that the names of these classes correspond to categorical properties of monads preserving them. Note that both affine and relevant equations include linear equations; in order to obtain later an accurate correspondence between monads and equations we must define more precise classes of non-linear equations. Intuitively, an equation is non-linear if either a variable fails to appear on one side, or if a variable appears more than once on one side. These two situations outline the classes of *drop* and *dup* equations, which we define below together with some of their subclasses.

Note that by definition, an equation is non-linear if and only if it features a duplicated variable or a deleted variable. Therefore the classes of linear, dup and drop equations cover all possible algebraic equations.

Definition 4.18 (Drop equations)**.** An equation $t_1 = t_2$ is

1. *drop* when at least one variable appears in t_1 but not in t_2 (or conversely);
2. *one-drop* when it features a variable that appears once in t_1 and does not appear in t_2 (or conversely);
3. *strict-drop* when at least one variable appears in t_1 but not in t_2 (or conversely) and no variable appears twice in t_1 or t_2.

The set of strict-drop equations is included in the set of one-drop equations, and the latter in the set of drop equations. Both inclusions are strict. Strict-drop equations can equivalently be characterised as non-linear equations in which variables appear at most once on each side.

Definition 4.19 (Dup equations)**.** An equation $t_1 = t_2$ is

1. *dup* when at least one variable appears more than once in t_1 or in t_2;
2. *2-dup* when it is dup and each variable appears at most twice in t_1 or t_2;
3. *strict-dup* when at least one variable appears more than once in t_1 or in t_2 and $\mathrm{Var}(t_1) = \mathrm{Var}(t_2)$

Equivalently, an equation is strict-dup when it is not linear and each variable of $\mathrm{Var}(t_1) \cup \mathrm{Var}(t_2)$ appears at least once in t_1 and in t_2. Every strict-dup equation is a dup equation. Figure 4.1 depicts an overview of dup and drop equations as well as their subclasses. As shown on this diagram, the classes of dup and drop equations overlap.

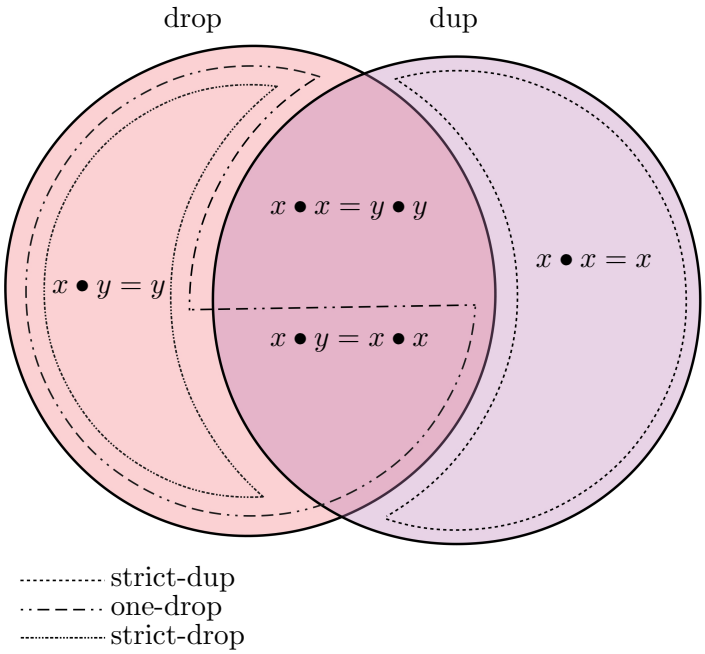


Figure 4.1: Classes of equations

Example 4.20. The law of absorption $x \bullet 0 = 0$ is a drop equation because x does not appear on the right side. More precisely, it is strict-drop because it contains no duplication. The equation $x \bullet (y \bullet y) = y \bullet y$ shows one occurrence of x on the left side and none on the right side, therefore it is a one-drop equation. Because y is duplicated, it is also a dup equation and not strict-drop. The equation $x \bullet x = y \bullet y$ is drop but not one-drop. △

Example 4.21. The law of idempotence $x = x \bullet x$ is a strict-dup equation. Distributivity of $\bullet$ over $+$, written $x \bullet (y + z) = x \bullet y + x \bullet z$ is strict-dup as well. The equation $x \bullet (y \bullet y) = y \bullet y$ is dup, because y is duplicated, but it is not strict-dup. △

Example 4.22. The law of inverse $x + (-x) = x$ is both a dup and a drop equation: the variable x is duplicated and appears on one side only. △

4.4 Classes of Monads and What They Preserve

The various residual diagrams presented in Section 4.2 offer sufficient conditions for the preservation of the equations they represent. We have seen that only the layout of variables matters: for example, the residual diagrams of $(x + y) * z = x * z$ will also permit to preserve $x * (y + z) = x + z$. In this section, we bring this remark even further by showing that the diagrams seen in Example 4.15 actually allow to preserve entire classes of equations.

4.4.1 Monoidal Monads

First we show that, without requiring any additional property, monoidal monads preserve any linear equation. First, we prove with this rather technical lemma that such monads are well-behaved regarding permutations.

Lemma 4.23. *Let $n \in \mathbb{N}$ and let T be a monoidal monad. Let α be an isomorphic natural transformation $(-)^n \to (-)^n$ realising a permutation of the elements of the product; in other words $\alpha = \langle \pi_{p(1)}, \ldots, \pi_{p(n)} \rangle$ for p an element of the* symmetric group *S_n. Then we have:*

$$\psi^{(n)} \circ \alpha = T\alpha \circ \psi^{(n)}$$

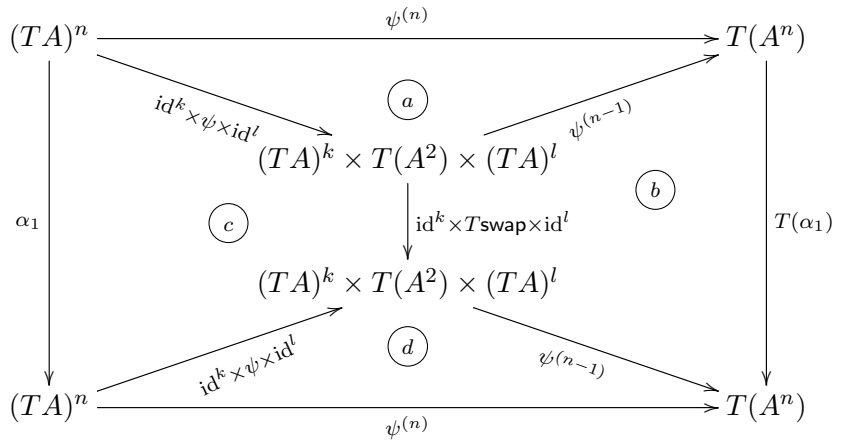
Proof. Let p be a permutation. If p is the identity, our result holds immediately. If not, p can be decomposed as a composition of *adjacent transpositions*. Hence α can be decomposed as $\alpha_m \circ \cdots \circ \alpha_1$ with each α_i realising an adjacent transposition, that is with α_i of the form:

$$\langle \pi_1, \ldots, \pi_{j-1}, \pi_{j+1}, \pi_j, \pi_{j+2}, \ldots, \pi_n \rangle = \mathrm{id} \times \cdots \times \mathrm{id} \times \mathsf{swap} \times \mathrm{id} \times \cdots \times \mathrm{id}$$

We prove that $\psi^{(n)} \circ (\alpha_m \circ \cdots \circ \alpha_1) = T(\alpha_m \circ \cdots \circ \alpha_1) \circ \psi^{(n)}$ by induction on the number m of adjacent transpositions. Note that $m > 1$ as there is at least one transposition in the decomposition of p.

- If $m = 1$, $\delta^V_{\mathcal{A}}(t_1) = \alpha_1$ that we write $\alpha_1 = \mathrm{id}^k \times \mathsf{swap} \times \mathrm{id}^l$ with $k + l + 2 = n$. Then

our property is represented as the following diagram:



(a) and (d) commute by monoidality, (b) commutes by naturality of $\psi^{(n-1)}$, (c) commutes because T is symmetric (it corresponds to Diagram (SYM)), which proves the base case of our induction.

- We assume that the property holds for m. Then we have:

$$T(\alpha_m \circ \cdots \circ \alpha_1) \circ \psi^{(n)} = \psi^{(n)} \circ (\alpha_m \circ \cdots \circ \alpha_1)$$

$$T\alpha_{m+1} \circ T(\alpha_m \circ \cdots \circ \alpha_1) \circ \psi^{(n)} = T\alpha_{m+1} \circ \psi^{(n)} \circ (\alpha_m \circ \cdots \circ \alpha_1)$$

$$T(\alpha_{m+1} \circ \cdots \circ \alpha_1) \circ \psi^{(n)} = \psi^{(n)} \circ (\alpha_{m+1} \circ \cdots \circ \alpha_1) \qquad \text{by our base case.}$$

□

In a linear equation, each variable appears only once on each side. In terms of rearrangement, this means that the arguments of both terms are formed by permutation: no duplication nor deletion is allowed, only a shuffling of our inputs. By Lemma 4.23, monoidal monads are compatible with this process, which results in the following preservation result.

Theorem 4.24. *Let T be a commutative monad. If $t_1 = t_2$ is a linear equation, then it is preserved by T.*

Proof. Since $t_1 = t_2$ is linear, each variable appears exactly once in each term. Therefore, for any algebra $\mathcal{A}$ with carrier A, $\delta^V_{\mathcal{A}}(t_1)$ and $\delta^V_{\mathcal{A}}(t_2)$ operate permutations of the product

$(TA)^{|V|}$. By Lemma 4.23 we have $\psi^{(|V|)} \circ T\delta^V_{\mathcal{A}}(t_1) = \delta^V_{\widehat{T}\mathcal{A}}(t_1) \circ \psi^{(|V|)}$, in other words $\mathcal{R}(T, t_1, V)$ commutes; and the same applies to t_2. Thus T preserves $t_1 = t_2$. □

Monoidal monads flawlessly preserve this type of equation, and therefore can be composed with with monads presented by linear signatures (Theorem 3.14), which confirms Manes and Mulry's result in [26].

The proof of Lemma (4.23) shows the structural importance of the symmetry property (Diagram (SYM)) for monoidal monads. Without this property, it would be impossible to preserve an equation when variables do not appear in the same order on both sides (for instance $x \times y = y \times x$). The very definition of linear equations legitimizes our requirement of symmetry, as it only involves the occurrence of variables and not their order. It is of course very convenient that monoidal monads on **Set** are automatically symmetric. In order to generalise our results outside **Set** one would have to require this property, or to use the conditions of Theorem 2.30 to ensure that monoidal monads are necessarily symmetric in the considered category.

Lemma (4.23) offers another valuable consequence: for monoidal monads on **Set**, the preservation of an equation does not depend on possible permutations of the variables. In other words, if T is monoidal, reordering the variables in an equation does not affect its preservation by T. This is perhaps reassuring, as our definition of $\delta^V_{\mathcal{A}}$ depends on the ordering of variables in V.

4.4.2 Affine Monads

In drop equations, for instance $x \bullet 0 = 0$, a variable is missing from one side. Theorem 4.24 does not apply there, and as we have seen in Example 4.15, not all monoidal monads preserve such equations. Intuitively, our monads need to be well-behaved regarding variable deletions. We show in this section that *affine monads* have the ability to preserve strict-drop equations.

Definition 4.25 ([22, 15])**.** A monoidal monad T on a Cartesian monoidal category $\mathbf{C}$ is called *affine* if it has one of the following equivalent properties.

- $T1$ is final.
- The following diagram commutes:

$$T1 \xrightarrow{!} 1 \xrightarrow{\eta_1} T1 \qquad \text{(id: } T1 \to T1\text{)} \tag{4.5}$$

- The following diagram commutes:

$$\begin{array}{ccccc} TA & \xrightarrow{!} & 1 & \xrightarrow{\eta_1} & T1 \\ & \searrow^{T!} & & \nearrow & \end{array} \tag{4.6}$$

- The following diagram commutes for all objects A, B:

$$\begin{array}{ccc} TA \times TB & \xrightarrow{\psi} & T(A \times B) \\ & \searrow^{\mathrm{id}} & \downarrow \chi \\ & & TA \times TB \end{array} \tag{4.7}$$

Example 4.26. Note that the first property can be conveniently written as

$$T1 = 1 \tag{4.8}$$

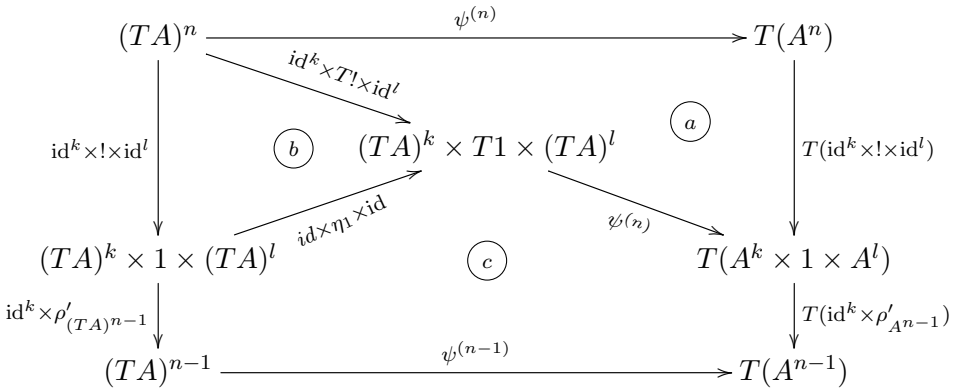
Both the distribution and non-empty powerset monad are affine (as $T1$ is easily seen to be final in both cases). On the other hand, the powerset monad is not affine because $\mathcal{P}1 = 2$. △

Lemma 4.27. *Let $m < n \in \mathbb{N}$, T an affine monoidal monad. Let ϵ be a natural transformation defined as $\epsilon_m \circ \cdots \circ \epsilon_1$, with each ϵ_i of the form $(\mathrm{id}^k \times \rho'_l) \circ (\mathrm{id}^k \times ! \times \mathrm{id}^l)$ for some $k, l \in \mathbb{N}$. Then we have:*

$$T\epsilon \circ \psi^{(n)} = \psi^{(n-m)} \circ \epsilon$$

Proof. We proceed by induction on m, the number of deletions realised by ϵ.

- If $m = 1$, then $\epsilon = (\mathrm{id}^k \times \rho'_l) \circ (\mathrm{id}^k \times ! \times \mathrm{id}^l)$ for some $k, l \in \mathbb{N}$. We prove below that $\psi^{(j)} \circ \epsilon = T\epsilon \circ \psi^{(|V|)}$. (c) commutes by Lemma 2.35, (a) commutes by naturality of $\psi^{(|V|)}$, (b) commutes because T is affine.



- Again we assume that the property holds for m. Then we have:

$$\begin{aligned} T(\epsilon_m \circ \cdots \circ \epsilon_1) \circ \psi^{(n)} &= \psi^{(n-m)} \circ (\epsilon_m \circ \cdots \circ \epsilon_1) \\ T\epsilon_{m+1} \circ T(\epsilon_m \circ \cdots \circ \epsilon_1) \circ \psi^{(n)} &= T\epsilon_{m+1} \circ \psi^{(n-m)} \circ (\epsilon_m \circ \cdots \circ \epsilon_1) \\ T(\epsilon_{m+1} \circ \cdots \circ \epsilon_1) \circ \psi^{(n)} &= \psi^{(n-m-1)} \circ (\epsilon_{m+1} \circ \cdots \circ \epsilon_1) \qquad \text{by our base case.} \end{aligned}$$

□

Theorem 4.28. *Let T be an affine monoidal monad. If $t_1 = t_2$ is a* strict-drop *equation, then it is preserved by T.*

Proof. If $t_1 = t_2$ is strict-drop, then $\mathrm{Arg}(t_1)$ and $\mathrm{Arg}(t_2)$ are obtained from V by deleting some variables and rearranging them. Again we write $\delta^V_{\mathcal{A}}(t_1)$ as $\alpha \circ \epsilon$, where α operates a permutation, and $\epsilon = \epsilon_m \circ \cdots \circ \epsilon_1$ with each ϵ_i deleting exactly one input. Thus ϵ_i is of the form $(\mathrm{id}^k \times \rho'_l) \circ (\mathrm{id}^k \times ! \times \mathrm{id}^l)$ for some $k, l \in \mathbb{N}$. We write $|\mathrm{Arg}(t_1)| = k$; note that since m inputs are deleted by ϵ we have $|V| - m = k$. Then we obtain:

$$\begin{aligned} T\delta^V_{\mathcal{A}}(t_1) \circ \psi^{(|V|)} &= T\alpha \circ T\epsilon \circ \psi^{(|V|)} \\ &= T\alpha \circ \psi^{(|V|-m)} \circ \epsilon && \text{by Lemma 4.27} \\ &= \psi^{(|V|-m)} \circ \alpha \circ \epsilon && \text{by Lemma 4.23} \\ &= \psi^{(k)} \circ \delta^V_{\widehat{T}\mathcal{A}}(t_1) \end{aligned}$$

Therefore $\mathcal{R}(T, t_1, V)$ commutes. The same reasoning applies to $\mathcal{R}(T, t_2, V)$, thus $t_1 = t_2$ is preserved. □

Example 4.29. We have seen in example 4.15 that $\mathcal{P}$ does not preserve the law of absorption $x \bullet 0 = 0$. On the contrary, the non-empty powerset monad $\mathcal{P}^+$ is affine, therefore it does preserve this equation. △

Note that the residual diagram for absorption, presented in Example 4.15, exactly corresponds to Diagram (4.6) which defines affineness. In a sense, $x \bullet 0 = 0$ is the simplest case of strict-drop equation. Affine monads clearly preserve it, along with all the other equations from this class.

4.4.3 Relevant Monads

Definition 4.30 ([22, 15])**.** A monoidal monad $T\colon \mathbf{C} \to \mathbf{C}$ on a Cartesian monoidal category $\mathbf{C}$ is *relevant* if one of the following equivalent conditions holds.

- The following diagram commutes for all object A:

$$\begin{array}{ccc} TA & \xrightarrow{\Delta} & TA \times TA \\ & \searrow^{T\Delta} & \downarrow{\scriptstyle \psi} \\ & & T(A \times A) \end{array} \tag{4.9}$$

- The following diagram commutes for all objects A, B:

$$\begin{array}{ccc} T(A \times B) & \xrightarrow{\chi} & TA \times TB \\ & \searrow^{\mathrm{id}} & \downarrow{\scriptstyle \psi} \\ & & T(A \times B) \end{array} \tag{4.10}$$

Example 4.31. The *Maybe* monad $\bot$ is relevant. Consider a set A and let $x \in \bot A = A + 1$. if $x = a \in A$ we have $(\bot\Delta)(a) = (a, a) = (\psi \circ \Delta)(a)$. Similarly, if $x = * \in 1$ we have $(\bot\Delta)(*) = * = (\psi \circ \Delta)(*)$. Hence Diagram 4.9 commutes and the monad is relevant. △

Example 4.32. Consider the writer monad $\mathcal{W}_M$ with $(M, \bullet, 1)$ a monoid. Let A be a set and $a \in A, m \in M$. Again we compute $(\mathcal{W}_M\Delta)(a, m) = ((a, a), m)$ whereas $(\psi \circ \Delta)(a, m) = ((a, a), m \bullet m)$. Therefore $\mathcal{W}_M$ is relevant if and only if $m \bullet m = m$ for

Monad	Affine	Relevant
Powerset $\mathcal{P}$	$\times$	$\times$
Non-empty Powerset $\mathcal{P}^+$	✓	$\times$
Distributions $\mathcal{D}$	✓	$\times$
Maybe $X + 1$	$\times$	✓
Reader $\mathcal{R}_C$	✓	✓
Writer $\mathcal{W}_M$	✓ iff M is trivial	✓ iff M is idempotent
Multiset $\mathbf{M}_{\mathbb{S}}$	✓ iff $\mathbb{S}$ is trivial	✓ iff $\mathbb{S}$ is trivial

Table 4.1: Affineness and relevance of well-known monads

every $m \in M$, in other words if M is an idempotent commutative monoid, also known as a bounded join-semilattice. △

In a similar way as we did for affine monads, we show that this categorical property models the ability for a monad to handle variable duplications.

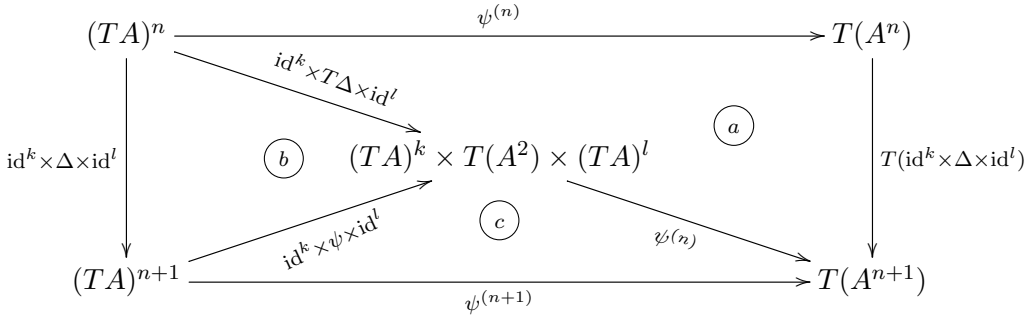
Lemma 4.33. *Let $n \in \mathbb{N}$, T a relevant monoidal monad. Let β be a natural transformation $(-)^n \to (-)^{n+m}$ defined as $\beta_m \circ \cdots \circ \beta_1$, with each β_i of the form $\mathrm{id} \times \cdots \times \mathrm{id} \times \Delta \times \mathrm{id} \times \cdots \times \mathrm{id}$. Then we have:*

$$T\beta \circ \psi^{(n)} = \psi^{(n+m)} \circ \beta$$

Proof. We proceed by induction on m, the number of duplications realised by β.

- For the case $m = 1$, we write $\beta_1 = \mathrm{id}^k \times \Delta \times \mathrm{id}^l$. We show that the diagram below

commutes, which represents our property.



(c) commutes by monoidality, (a) commutes by naturality of $\psi^{(n)}$, (b) commutes because T is relevant.

- We assume that the property holds for m. Then we have:

$$T(\beta_m \circ \cdots \circ \beta_1) \circ \psi^{(n)} = \psi^{(n+m)} \circ (\beta_m \circ \cdots \circ \beta_1)$$
$$T\beta_{m+1} \circ T(\beta_m \circ \cdots \circ \beta_1) \circ \psi^{(n)} = T\beta_{m+1} \circ \psi^{(n+m)} \circ (\beta_m \circ \cdots \circ \beta_1)$$
$$T(\beta_{m+1} \circ \cdots \circ \beta_1) \circ \psi^{(n)} = \psi^{(n+m+1)} \circ (\beta_{m+1} \circ \cdots \circ \beta_1) \qquad \text{by our base case.}$$

□

Theorem 4.34. *Let T be a relevant monoidal monad. If $t_1 = t_2$ is a* strict-dup *equation, then it is preserved by T.*

Proof. For a strict-dup equation $t_1 = t_2$, $\mathrm{Arg}(t_1)$ and $\mathrm{Arg}(t_1)$ are obtained from V by duplicating some variables and rearranging them. In a similar fashion as in the proof of Theorem 4.24, we write $\delta^V_{\mathcal{A}}(t_1)$ as $\alpha \circ \beta$, where α operates a permutation, and $\beta = \beta_m \circ \cdots \circ \beta_1$ with β_i of the form $\mathrm{id} \times \cdots \times \mathrm{id} \times \Delta \times \mathrm{id} \times \cdots \times \mathrm{id}$. Intuitively, we first duplicate using Δ all the inputs that need to appear several times, then we rearrange them all with α. If we write $|\mathrm{Arg}(t_1)| = k$, because there are exactly m duplications we

have $|V| + m = k$. Thus we have:

$$\begin{aligned} T\delta^V_{\mathcal{A}}(t_1) \circ \psi^{(|V|)} &= T\alpha \circ T\beta \circ \psi^{(|V|)} \\ &= T\alpha \circ \psi^{(|V|+m)} \circ \beta && \text{by Lemma 4.33} \\ &= \psi^{(|V|+m)} \circ \alpha \circ \beta && \text{by Lemma 4.23} \\ &= \psi^{(k)} \circ \delta^V_{\widehat{T}\mathcal{A}}(t_1) \end{aligned}$$

Therefore $\mathcal{R}(T, t_1, V)$ commutes. The same reasoning applies to $\mathcal{R}(T, t_2, V)$, thus $t_1 = t_2$ is preserved. □

Example 4.35. The equation of idempotency $x \bullet x = x$ is a typical *dup* equation. We have seen before that $\mathcal{P}$ cannot preserve it. We know now that the Maybe monad $\perp$ does preserve it, by its property of relevance. △

Again we remark that the residual diagram for idempotency, visible in Example 4.15, is identical to the definition of relevance (4.9).

4.4.4 Cartesian monads

Definition 4.36 ([15])**.** A monad T is called *Cartesian* if it is both affine and relevant.

Example 4.37. The monad $\mathcal{R}_C$ is Cartesian. It is easy to see that because $(X \times Y)^C = X^C \times Y^C$, χ and ψ are isomorphisms and inverse of each other, therefore diagrams 4.7 and 4.10 both commute. △

Cartesian monads combine the powers of relevant and affine monads, yielding the following preservation result.

Theorem 4.38. *Let T be a commutative, relevant and affine monad. For all t_1 and t_2, T preserves $t_1 = t_2$.*

Proof. We decompose the morphism $\delta^V_{\mathcal{A}}(t_1)$ as $\alpha \circ \beta \circ \epsilon$, where ϵ drops the unnecessary variables, β duplicates the ones that need copying, then α rearranges them to match the arguments of t_1. It is clear that all variable combinations can be obtained this way. For

the term t_1, let us assume that there are l deletions and m duplications, hence we have $k = |\mathrm{Arg}(t_1)| = |V| - l + m$

$$\begin{aligned} T\delta^V_{\mathcal{A}}(t_1) \circ \psi^{(|V|)} &= T\alpha \circ T\beta \circ T\epsilon \circ \psi^{(|V|)} \\ &= T\alpha \circ T\beta \circ \psi^{(|V|-l)} \circ \epsilon && \text{by Lemma 4.27} \\ &= T\alpha \circ \psi^{(|V|-l+m)} \circ \beta \circ \epsilon && \text{by Lemma 4.33} \\ &= \psi^{(|V|-l+m)} \circ \alpha \circ \beta \circ \epsilon && \text{by Lemma 4.23} \\ &= \psi^{(k)} \circ \delta^V_{\widehat{T}\mathcal{A}}(t_1) \end{aligned}$$

Therefore $\mathcal{R}(T, t_1, V)$ commutes, and again the same reasoning applies to $\mathcal{R}(T, t_2, V)$, thus $t_1 = t_2$ is preserved. □

Example 4.39. We have seen before the case of the inverse law $x + (-x) = 0$. This equation is both dup and drop, therefore neither Theorem 4.28 nor Theorem 4.34 apply. However, as seen at the beginning of this chapter, this equation is preserved by $\mathcal{R}_C$ because this monad is both affine and relevant. △

This series of preservation properties can now be connected with the results of Chapter 3 to obtain distributive laws.

Example 4.40. The monad $\mathcal{R}_C$ preserves all the laws of Abelian groups because it is Cartesian, therefore by Theorem 3.5 there exists a distributive law $\mathcal{A}\mathcal{R}_C \to \mathcal{R}_C\mathcal{A}$. As a matter of fact, $\mathcal{R}_C$ can be composed with any finitary monad as it preserves all equations. △

Example 4.41. The non-empty finite powerset monad ${\mathcal{P}_f}^+$ is presented by the theory of bounded semilattices (Example 2.48). It is not a linear theory as it features an equation of idempotence, but by Theorem 4.34 this law is preserved by a relevant monad such as $\bot$. Therefore we can compose $\bot$ and ${\mathcal{P}_f}^+$ via a distributive law. Note that in this case we retrieve the distributive law defined by Jacobs in [15], and that the composite monad $\bot{\mathcal{P}_f}^+$ is equal to the finite powerset monad $\mathcal{P}_f$. △

4.5 Discussion and Related Work

In this chapter we have discussed a series of classes of monads and the equations they preserve by making use of their monoidal structure. Our findings can be seen as an extension of some existing works in the literature. As mentioned in the previous chapter, Gautam's study of complex algebras already features the main ideas behind our work. The author studies the equations that are preserved by the powerset monad and classifies them in terms of variable rearrangements, already outlining the classes of linear, dup and drop equations. A first generalisation of part of Gautam's work appears in Manes and Mulry's paper [26], where it is shown that all monoidal monads (and not only $\mathcal{P}$) preserve linear theories. In this chapter, we have confirmed this result with Theorem 4.38, then further generalised it to the classes of monoidal, affine and relevant monads together with the equations they preserve with Theorems 4.28, 4.34 and 4.38.

Gautam has shown with Theorem 3.1 that $\mathcal{P}$ does not preserve dup nor drop equations. In the light of the current chapter, it is tempting to explain this non-preservation by the properties of $\mathcal{P}$: the monad lacks affineness and relevance, which would allow non-linear equations to be preserved. However, can we affirm that a non-relevant (resp. non-affine) monad never preserves dup (resp. drop) equations? Can we obtain converse results to Theorems 4.24, 4.28, 4.34 and 4.38? The next chapter will explore this question and build a series of necessary conditions corresponding to various cases of equation preservation.

Chapter 5

Necessary Conditions for Preservation

The previous chapters have explored the notion of equation preservation by a monoidal monad, and presented a series of sufficient conditions. We have seen that the property of affineness of a monad is sufficient to ensure that a strict-drop equation is preserved. Similarly, if a monad is relevant, then it preserves equations that feature duplicated variables.

In this chapter, we investigate the converse of these results. If a monad T preserves a drop equation, is it necessarily affine? Or, by contraposition: if T is not affine (resp. relevant), can we say that it does not preserve strict-drop equations (resp. strict-dup equations) in general? Once more, Gautam's results in [11] shed some light on the question. The author studies the case of the powerset monad: $\mathcal{P}$ is neither affine nor relevant, and indeed does not preserve any strict-drop or strict-dup equation. This chapter focuses on generalising Gautam's result. First, we focus on drop equations and show that our conjecture is verified: if a monoidal monad T preserves a strict-drop equation, then it is affine. Then we examine the case of dup equations, which turns out to be more complicated. The equation of idempotence is strict-dup, and we show that its preservation does imply relevance of the monad. However we also uncover certain cases of strict-dup equations that are preserved by non-relevant monads. Generalising the case of idempotence, we outline a subclass of strict-dup equations whose preservation is equivalent to relevance. Finally we show that preserving some equations does not imply relevance but a weaker property, which we call n-relevance.

5.1 Affine Monads and Drop Equations

A sufficient condition for the preservation of certain equations is expressed in Theorem 4.28: if a monoidal monad T preserves a strict-drop equation, then it is affine. In this section, we focus on the converse and, in fact, prove a stronger result: preservation of *one-drop* equations implies affineness. This covers strict-drop equations, but also other equalities outside the scope of Theorem 4.28, such as $x \bullet (y \times y) = y$.

In this section, we study the preservation of a one-drop equation by a monoidal monad T on a particular algebra, and we show that it implies affineness. We need to choose a convenient algebra to carry out our proof: for the case of affine monads and drop equations, the trivial algebra 1 is a sensible choice for several reasons. First, because it is an algebra for every theory: we do not need to prove that our drop equations hold on it. Recall also that affineness of a monad T can be characterised by the equality $T1 = 1$ (4.8). Intuitively, this means that the set 1 is a good candidate to capture the property of affineness of a monad T.

We now proceed to the main lemma, which despite being quite technical expresses a simple idea. Recall the conditions given by the residual diagrams. In the case of a drop equation, they may or may not commute, but if we precompose them with a certain morphism α, we can ensure that one of them commutes and that the other characterises the affineness of T.

Lemma 5.1. *Let $t_1 = t_2$ be a one-drop equation with $t_1, t_2 \in \mathsf{F}_\Sigma V$ and $\mathrm{Var}(t_1) \cup \mathrm{Var}(t_2) = V$. Let $T\colon \mathbf{Set} \to \mathbf{Set}$ be a monoidal monad. We define $B = T1 \times 1^n$ and $\alpha = \mathrm{id} \times (\eta_1)^n\colon T1 \times 1^n \to (T1)^n$; then:*

1. *The following diagram commutes:*

$$\begin{array}{ccccc} B & \xrightarrow{\alpha} & (T1)^{|V|} & \xrightarrow{\delta^V_{T1}(t_2)} & (T1)^{k_2} \\ {\scriptstyle\alpha}\downarrow & & & & \downarrow{\scriptstyle\psi} \\ (T1)^{|V|} & \xrightarrow[\psi]{} & T(1^{|V|}) & \xrightarrow[T\delta^V_1(t_2)]{} & T(1^{k_2}) \end{array} \tag{5.1}$$

2. *If the following diagram commutes, then T is affine:*

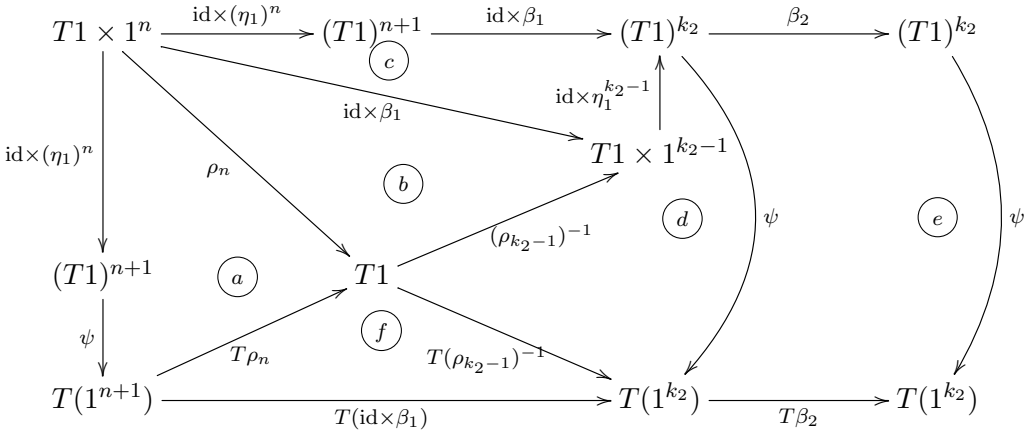
$$\begin{array}{ccccc} B & \xrightarrow{\alpha} & (T1)^{|V|} & \xrightarrow{\delta^V_{T1}(t_1)} & (T1)^{k_1} \\ {\scriptstyle\alpha}\downarrow & & & & \downarrow{\scriptstyle\psi} \\ (T1)^{|V|} & \xrightarrow[\psi]{} & T(1^{|V|}) & \xrightarrow[T\delta^V_1(t_1)]{} & T(1^{k_1}) \end{array} \tag{5.2}$$

Or the other way around, by substituting t_1 and t_2.

Proof. Let $t_1 = t_2$ be a one-drop equation. For $n \in \mathbb{N}$, we denote $|V|$ by $n+1$, as by definition a one-drop equation contains at least one variable. Without loss of generality, we can say that this variable x appears once in t_2 but not in t_1. Again without loss of generality, we can reorder our variables and assume that x is the first variable in our enumeration of V. Let $|V| = n+1$. Recall that we define $B = T1 \times 1^n$ and $\alpha = \mathrm{id} \times (\eta_1)^n \colon T1 \times 1^n \to (T1)^n$.

Note that because the first variable is dropped in t_1, $\delta^V_{\mathcal{A}}(t_1)$ can be decomposed as $\delta^V_{\mathcal{A}}(t_1) = \rho \circ (! \times \beta)$ where $\beta \colon (-)^n \to (-)^{k_1}$ is a natural transformation carrying out the rearrangement of the remaining inputs. Similarly, because x appears only once in t_2, $\delta^V_{\mathcal{A}}(t_2)$ can be decomposed as $\delta^V_{\mathcal{A}}(t_1) = (\beta_2) \circ (\mathrm{id} \times \beta_1)$. The first input is not modified, $\beta_1 \colon (-)^n \to (-)^{k_2 - 1}$ is the natural transformation that may drop or duplicate some of the other inputs, and finally $\beta_2 \colon (-)^{k_2} \to (-)^{k_2}$ realises a permutation of all its inputs.

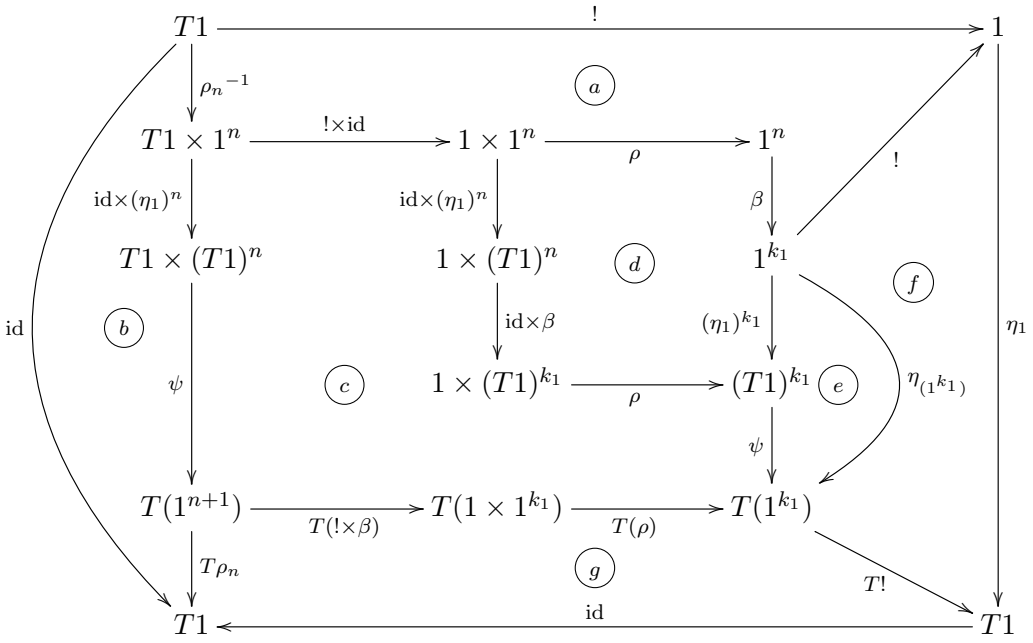
We start with point 1. Note that $k_2 = \mathrm{Arg}(t_2) > 0$ as t_2 features at least one variable by assumption. Let us subdivide the diagram as follows:



(a) and (d) commute by Lemma 2.36. (b) and (f) commute by Lemma 2.38. (c) commutes by naturality of β_1. Finally, (e) commutes by Lemma 4.23 since β_2 encodes a permutation. We've proven point 1.

Next up is point 2. We assume commutation of the diagram (5.2) and show the commutation of the following diagram. Note that the outer diagram amounts to $\eta_1 \circ ! = \mathrm{id}$,

which characterises the affineness of T (4.5).



(a) and (g) commute by finality. (b) commutes by property (2.8). (c) corresponds to our assumption (5.2). (d) commutes by naturality of ρ and (f) by naturality of η, whereas (e) commutes by monoidal property (MM.1). □

We are ready to prove the following result. It is crucial to our purpose, because it shows that studying preservation only on the trivial algebra 1 is sufficient to derive affineness.

Lemma 5.2. *Let $t_1 = t_2$ be a one-drop equation as in Lemma 5.1, and T: **Set** $\to$ **Set** a monoidal monad. If $t_1 = t_2$ holds on $\widehat{T}1$, then T is affine.*

Proof. $t_1 = t_2$ trivially holds on 1, therefore we have:

$$\gamma_1^V(t_1) \circ \delta_1^V(t_1) = \gamma_1^V(t_2) \circ \delta_1^V(t_2) \tag{5.3}$$

Let α be the morphism given by Lemma 5.1. The equation is preserved by T, hence it holds on $\widehat{T}1$. Then we have:

$$
\begin{aligned}
& T\gamma_1^V(t_1) \circ \psi \circ \delta^V_{\widehat{T}1}(t_1) \circ \alpha \\
&\quad = \gamma^V_{\widehat{T}1}(t_1) \circ \delta^V_{\widehat{T}1}(t_1) \circ \alpha && \text{by Lemma 4.8} \\
&\quad = \gamma^V_{\widehat{T}1}(t_2) \circ \delta^V_{\widehat{T}1}(t_2) \circ \alpha && t_1 = t_2 \text{ holds on } \widehat{T}1 \\
&\quad = T\gamma_1^V(t_2) \circ \psi \circ \delta^V_{\widehat{T}1}(t_2) \circ \alpha && \text{by Lemma 4.8} \\
&\quad = T\gamma_1^V(t_2) \circ T\delta_1^V(t_2) \circ \psi \circ \alpha && (5.1) \\
&\quad = T\gamma_1^V(t_1) \circ T\delta_1^V(t_1) \circ \psi \circ \alpha && (5.3)
\end{aligned}
$$

The map $\gamma_1^V(t_1)$ is an isomorphism, therefore we can precompose the previous equality with $T\gamma_1^V(t_1)^{-1}$ and obtain (5.2). Hence T is affine by Lemma 5.1. □

The above lemma leads naturally to the main result of this section.

Theorem 5.3. *Let $t_1 = t_2$ be a one-drop equation with $t_1, t_2 \in \mathsf{F}_\Sigma V$ and $\mathrm{Var}(t_1) \cup \mathrm{Var}(t_2) = V$. Let $T\colon \mathbf{Set} \to \mathbf{Set}$ be a monoidal monad. If T preserves $t_1 = t_2$, then T is affine.*

Proof. Any equation $t_1 = t_2$ trivially holds on 1. If T preserves $t_1 = t_2$, then it holds on $T1$. Thus by Lemma 5.2, T is affine. □

The above theorem is particularly useful in contrapositive form: if a monad is *not* affine, then we know that it does not preserve any drop equations. In particular, it generalises Gautam's result: that $\mathcal{P}$ does not preserve any one-drop equation [11].

Example 5.4. The maybe monad $X + 1$, the writer monad $X \times M$ for M a non-trivial monoid, and the generalised multiset monad $\mathbf{M}_\mathbb{S}$ for $\mathbb{S}$ a non-trivial semiring (see Table 4.1) are not affine. Hence by Theorem 5.3 they do not preserve one-drop equations, for instance $x \times 0 = 0$. △

Note that Theorem 5.3 treats preservation of single equations. Another consequence of Lemma 5.2 is that, if a monoidal monad $T\colon \mathbf{Set} \to \mathbf{Set}$ preserves a non-empty *set* of equations E that includes a one-drop equation, then it is affine. To see this, note that any equation holds on the algebra 1; therefore also on $T1$, hence T is affine by Lemma 5.2.

Because one-drop equations include strict-drop equations, we can combine Theorems 5.3 and 4.28 to obtain the following comprehensive result.

Theorem 5.5. *Let $t_1 = t_2$ be a strict-drop equation, T a monoidal monad.*

T is affine if and only if T preserves $t_1 = t_2$.

This theorem highlights the importance of the strict-drop class ; such an equivalence could not be obtained with the class of affine equations (see Definition 4.17) as they also include linear equations.

5.1.1 Decidability

The previous section establishes the equivalence between affineness of a monad T and preservation of one-drop equations. We now use this result to analyse a more algorithmic question: is it decidable whether a monad T (presented by finitely many operations and equations) preserves a given equation $t_1 = t_2$? Unfortunately the answer is negative, which we prove by showing that the question whether a given monad is affine is undecidable.

We use an encoding of the following decision problem, which is known to be undecidable [6].

Theorem 5.6. *The following problem is undecidable:*

- *Instance: a finite presentation (G, R) of a monoid $\mathcal{M}$;*
- *Question: Is $\mathcal{M}$ trivial?*

This allows us to conclude on the decidability of affineness.

Theorem 5.7. *The following problem is undecidable:*

- *Instance: a finite signature Σ, a finite set E of equations;*
- *Question: Is the monad T presented by (Σ, E) affine?*

Proof. Let (G, R) be a finite presentation. We construct the following theory:

$$\begin{aligned} \Sigma = &\quad \text{unary operations } f_g, \text{ for } g \in G \\ E = &\quad f_{g_1}(f_{g_2}(\dots f_{g_n}(x)\dots)) = f_{h_1}(f_{h_2}(\dots f_{h_k}(x)\dots)) \\ &\quad \text{for each pair } (g_1 \dots g_n, h_1 \dots h_k) \in R \end{aligned}$$

Let T be the monad presented by (Σ, E). We show that $T1$ is isomorphic to $\mathcal{M}$. We write e for the unit of $\mathcal{M}$ and $*$ for the element of 1. Each element of $T1$ can be seen as a (Σ, E)-term on the generator $*$, through the map $F\colon T1 \to \mathcal{M}$ defined by $F(x) = e$ and $F(f_{g_1}(f_{g_2}(\dots f_{g_n}(*)\dots))) = g_1 g_2 \dots g_n$. This map is an isomorphism: by construction, $F(u) = F(v) \Leftrightarrow u = v$ for u, v terms of $T1$, and all $m \in \mathcal{M}$ can be written as some $F(t)$ for $t \in T1$. Hence $T1 = 1$ iff $\mathcal{M}$ is trivial. By this reduction we conclude that affineness is undecidable. □

Because we have established the equivalence between preserving a class of equations and affineness, and because the latter is undecidable, we obtain a general result on the decidability of equation preservation.

Corollary 5.8. *The following problem is undecidable:*

- *Instance: a finite theory (Σ, E), an equation $t_1 = t_2$*
- *Question: does the monad T presented by (Σ, E) preserve $t_1 = t_2$?*

5.2 Relevant Monads and Dup Equations

Having treated affine monads, we now turn to the other main class of interest: relevant monads. Theorem 4.34 shows that if a monad is relevant, it preserves strict-dup equations; in this section we explore the converse.

The question turns out to be more complicated than the case of affine monads. Unlike affineness, the property of relevance is not related to the final object 1; instead, its definition (4.30) refers to arbitrary objects of the category. This time, we cannot hope to prove relevance by considering only the trivial algebra. In this section we present a rather complex strategy to tackle the problem. It is built around the same idea as the previous case: for a dup equation, we first find a convenient algebra where the equation holds, then assuming its preservation we derive the property of relevance.

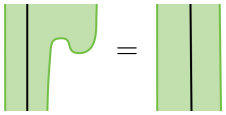
Our proof method for dup equations and relevant monads is quite technical; for more clarity we choose to illustrate it with a new type of diagram. First, we get familiar with those 'Sleeve' diagrams, then we show how they allow us to understand that preservation of dup equations leads to relevance.

5.2.1 A Diagrammatic Approach

Our diagrams are inspired from several well-known graphical representations for monoidal categories. The concept of *functorial boxes* is introduced by Cockett and Seely in [9] then further examined by Melliès in [29], where functors are represented as boxes surrounding morphisms and objects. Monoidal properties then allow to gather several objects inside a single box. More details on this representation are given by McCurdy [28], although he focuses on monoidal functors verifying some particular properties. Most of these assumptions, which include the *Frobenius property*, cannot be made in our case as they would entail affineness and relevance and defeat the purpose of this section. For this reason, we do not benefit from the same soundness and completeness properties as McCurdy for our diagrammatic calculus. We will therefore only use such diagrams as an inspiration and an illustration of our method and still provide a categorical proof for our results. Let us first summarise a few central ideas of this type of diagrams.

An arbitrary object X of our category is now represented by a thread (or 'wire'), and the application of T on this object results in a 'sleeve' covering it. We read a diagram from bottom to top and represent products implicitly as horizontal adjacency. The morphism $\chi\colon T(X \times Y) \to TX \times TY$ is modelled by a cup-like shape where one sleeve containing two objects splits into two sleeved objects, whereas ψ is modelled in the opposite way and merges two sleeved objects into a single sleeve.

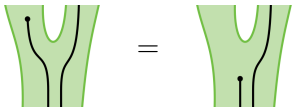
Note that the object 1 is not represented in our diagrams. By the isomorphism $X \times 1 \simeq 1$, we can imagine the presence of 1 as a vertical thread anywhere on the diagram without affecting the calculations. Some deformations of the outline of sleeves are allowed: for instance, the following diagrammatic equality corresponds to the right unitality of a monoidal functor (MF.2). Note that the neither the product nor the unitor ρ is explicitly represented in the diagram.



(5.4)

We can 'delete' an object by mapping it to the final object 1. We represent this process with an unfinished vertical thread. Naturality of χ and ψ allow to 'pull' these

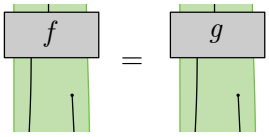
threads to the bottom of the diagram:



$$(T! \times \mathrm{id}) \circ \chi \quad = \quad \chi \circ T(! \times \mathrm{id}) \tag{5.5}$$

Finally, unfinished threads may be ignored:

Lemma 5.9. *For morphisms $f, g \colon T(X \times 1) \to TY$, we have $f = g$ if and only if the following diagram equality holds for all X, Y non-empty sets.*



Proof. In **Set**, the map $! \colon X \to 1$ is an epimorphism when X is a non-empty set, therefore $T(\mathrm{id} \times !)$ as well as $\mathrm{id} \times T!$ are epimorphisms. Thus we have

$$f \circ T(\mathrm{id} \times !) = g \circ T(\mathrm{id} \times !)$$
$$f = g$$

□

In the rest of this chapter, we will sometimes make use of these properties to ignore certain threads by mapping them to the object 1. Intuitively, we pull the thread to the bottom of the diagram then apply Lemma 5.9.

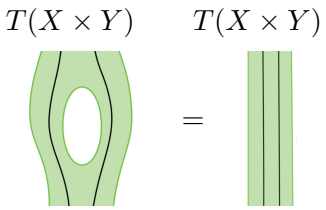
We finally recall a specific property that will become useful in some of our proof. The following holds trivially for a monoidal monad on **Set** and a set A (see for instance [15]).

$$\Delta_{TA} = \chi_{A,A} \circ T\Delta_A \colon TA \to TA \times TA \tag{5.6}$$

In terms of our diagrams, it implies that duplicating a sleeved object amounts to duplicating the object inside the sleeve, then separating both copies using χ.

The composition $\psi \circ \chi$ splits a sleeved product, then reunites both components into one sleeve. We recall from (4.10) that in the case of relevant monads, this yields the

identity.



$$\psi \circ \chi \quad = \quad \mathrm{id} \tag{5.7}$$

In [28], this diagrammatic equality is presented as a property of connectivity of functorial regions. We like to describe the graphical aspect of this property as follows: applying χ followed by ψ results in a 'bubble' in our sleeve, surrounded by two threads representing arbitrary objects, and relevance allows to pop this bubble. In the rest of this section, we develop a method to reduce complex equational problems to this 'bubble popping' property.

First, we show a slightly different characterisation of relevance which will become more convenient in the rest of this chapter. The property expressed in diagram (5.8) is similar to our definition of relevance (4.10) (in particular, it is easy to show that it is verified by relevant monads). However, it only involves a single set A.

Lemma 5.10. *A monoidal monad T on* **Set** *is relevant if and only if the following diagram commutes for all $A \in$* **Set***.*

$$\begin{array}{ccccccc} T(A \times A) & \xrightarrow{T\Delta} & T(A^2 \times A^2) & \xrightarrow{\chi} & T(A^2) \times T(A^2) & \xrightarrow{\psi} & T(A^2 \times A^2) \\ & \searrow & & & & & \downarrow{\scriptstyle T(\pi_1 \times \pi_2)} \\ & & & & & & T(A \times A) \end{array} \tag{5.8}$$

Proof.

- ($\Rightarrow$) We assume that T is relevant, hence $\psi \circ \chi = \mathrm{id}$ and (5.8) commutes immediately.

- ($\Leftarrow$) Let C, D be non-empty sets, and let $A = C \times D$. Assume that Diagram (5.8) commutes for this choice of A. We precompose it with

$$T\Delta \colon T(C \times D) \to T((C \times D) \times (C \times D))$$

and postcompose it with

$$T(\pi_1 \times \pi_2) \colon T((C \times D) \times (C \times D)) \to T(C \times D)$$

Then we obtain the following equalities between morphisms of type $T(C \times D) \to T(C \times D)$:

$$\begin{aligned}
\mathrm{id} &= T(\pi_1 \times \pi_2) \circ \mathrm{id} \circ T\Delta \\
&= T(\pi_1 \times \pi_2) \circ T(\pi_1 \times \pi_2) \circ \psi \circ \chi \circ T\Delta \circ T\Delta \\
&= T(\pi_1 \times \pi_4) \circ \psi \circ \chi \circ T\Delta \circ T\Delta \\
&= \psi \circ T(\pi_1) \times T(\pi_4) \circ \chi \circ T\Delta \circ T\Delta && \text{by naturality of } \psi \\
&= \psi \circ \chi \circ T(\pi_1 \times \pi_4) \circ T\Delta \circ T\Delta && \text{by naturality of } \chi \\
&= \psi \circ \chi
\end{aligned}$$

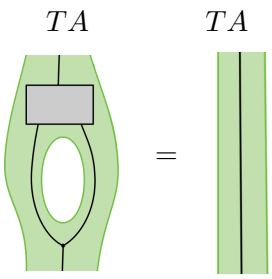
Which proves the relevance of T.

□

We will now present our proof method and illustrate it with sleeve diagrams. We start with a simple case, then study a more complex example, and finally we define a general strategy for a large subclass of dup equations.

5.2.2 The Case of Idempotence

Now we focus on the simplest dup equation: idempotence of a binary operation $x \bullet x = x$. Assuming that T preserves it, our strategy is to define an algebra $\mathcal{A}$ and an operation $\bullet_{\mathcal{A}}$ such that $x \bullet_{\mathcal{A}} x = x$, and then to derive the relevance of T from the preservation of the idempotence of $\bullet_{\mathcal{A}}$. For such an algebra (whose carrier is denoted by A), we draw the following diagrams. The grey box represents our binary idempotent operation $\bullet_{\mathcal{A}}$, and the left diagram represents the term $x \bullet_{\mathcal{A}} x$ on the lifted algebra $\widehat{T}\mathcal{A}$. By preservation of idempotence, it must be equal to the identity, modelled by the right diagram.

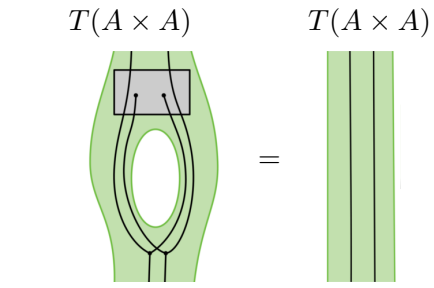


$$T(\bullet_{\mathcal{A}}) \circ \psi \circ \Delta \quad = \quad \mathrm{id} \tag{5.9}$$

In order to derive the property of relevance from this equality, we conveniently choose $\mathcal{A}$ and $\bullet_{\mathcal{A}}$. Let A be a non-empty set, we define on $A \times A$ the following binary operation:

$$\begin{aligned} \bullet_{\mathcal{A}} \colon (A \times A) \times (A \times A) &\to (A \times A) \\ ((a,b),(c,d)) &\mapsto (a,d) \end{aligned}$$

The operation $\bullet_{\mathcal{A}}$ may be defined categorically as $(\pi_1 \times \pi_2)$ and is idempotent: $\bullet_{\mathcal{A}}((a,b),(a,b)) = (a,b)$. For this algebra, the previous diagrams become:



$$T(\bullet_{\mathcal{A}}) \circ \psi \circ \Delta = \mathrm{id} \tag{5.10}$$

We are now very close to (5.7).

Theorem 5.11. *Let T be a monoidal monad. If T preserves $x \bullet x = x$, then T is relevant.*

Proof. Consider the algebra $\mathcal{A}$ we have constructed; $x \bullet x = x$ holds on $\mathcal{A}$ and $\bullet_{\mathcal{A}} = (\pi_1 \times \pi_2)$. By preservation of idempotence, $x \bullet x = x$ also holds on $\widehat{T}\mathcal{A}$. Therefore, as shown in the diagrams above, we have the following equality on $T(A \times A)$:

$$T(\bullet_{\mathcal{A}}) \circ \psi \circ \Delta = \mathrm{id}$$

$$T(\pi_1 \times \pi_2) \circ \psi \circ \Delta = \mathrm{id}$$

Hence immediately by Lemma 5.10, T is relevant.

□

The preservation of idempotence therefore implies relevance. As we did for drop equations, can we generalise this result to the whole class of strict-dup equations? the answer is no: consider for instance the equation

$$x \bullet (y \bullet y) = y \bullet x \tag{5.11}$$

and the generalised multiset monad $\mathcal{M}_{\mathbb{Z}_2}$ (counting multipicity in the ring $\mathbb{Z}_2$). We show that $\mathcal{M}_{\mathbb{Z}_2}$ preserves the equation (5.11) despite not being relevant(see Table 4.1). Before we prove this, we remark that (5.11) implies $x \bullet y = y \bullet x$. Indeed:

$$\begin{aligned} x \bullet y &= y \bullet (x \bullet x) = (x \bullet x) \bullet (y \bullet y) = (x \bullet x) \bullet (y \bullet (y \bullet y)) \\ &= (x \bullet x) \bullet ((y \bullet y) \bullet (y \bullet y)) = (y \bullet y) \bullet (x \bullet x) = x \bullet (y \bullet y) = y \bullet x. \end{aligned}$$

To show that $\mathcal{M}_{\mathbb{Z}_2}$ preserves (5.11), let us assume that we are given an algebra $\mathcal{A}$ with carrier A and a binary operation $\bullet_{\mathcal{A}} \colon A^2 \to A$. Moreover we assume that$(x \bullet_{\mathcal{A}} (y \bullet_{\mathcal{A}} y)) = (y \bullet_{\mathcal{A}} x)$ for all $x, y \in A$. Note that $\xi \bullet_{\widehat{\mathcal{M}_{\mathbb{Z}_2}}\mathcal{A}} \xi' = \sum_{x,x'} \xi(x)\xi'(x')(x \bullet_{\mathcal{A}} x')$ for any $\xi, \xi' \in \mathcal{M}_{\mathbb{Z}_2} X$, thus

$$(\xi \bullet_{\widehat{\mathcal{M}_{\mathbb{Z}_2}}\mathcal{A}} \xi) = \sum_x \xi(x)^2 (x \bullet_{\mathcal{A}} x) = \sum_x \xi(x)(x \bullet_{\mathcal{A}} x)$$

as the off-diagonals terms cancel each other due to $(x \bullet_{\mathcal{A}} y) + (y \bullet_{\mathcal{A}} x) = 2(x \bullet_{\mathcal{A}} y) = 0$. Hence

$$\begin{aligned} (\xi \bullet_{\widehat{\mathcal{M}_{\mathbb{Z}_2}}\mathcal{A}} (\xi' \bullet_{\widehat{\mathcal{M}_{\mathbb{Z}_2}}\mathcal{A}} \xi')) &= \sum_{x,y \in X} \xi(x)\xi'(y)(x \bullet_{\mathcal{A}} (y \bullet_{\mathcal{A}} y)) \\ &= \sum_{x,y \in X} \xi(x)\xi'(y)(y \bullet_{\mathcal{A}} x) \\ &= (\xi' \bullet_{\widehat{\mathcal{M}_{\mathbb{Z}_2}}\mathcal{A}} \xi). \end{aligned}$$

If we want to prove that preservation of an equation implies relevance of a monad, we must restrict our scope to a subclass of dup equations. The next section defines such a class and a corresponding proof strategy inspired by our approach of relevance.

5.2.3 Sketching the Method for $t[x]$-Equations

The method we applied to idempotence can be generalised to a more general class of equations. Let us now consider a linear Σ-term t and a binary operation $\bullet \in \Sigma$; we focus here on a particular case of dup equation: equalities of the form $t[x \bullet x] = t[x]$. In other words the equation features only one variable duplication and it is the only difference between the two sides; therefore it is always a strict-dup equation. We call such an equality a $t[x]$-equation, and we will show that preserving such an equation implies relevance of the monad.

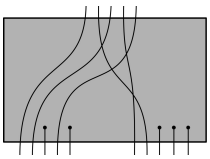
First, we consider a relatively simple case of $t[x]$-equation: when $t[x \bullet x] = t[x]$ only uses one symbol of the signature, the binary operation $\bullet$. Such an equation can be for example $z \bullet (y \bullet (x \bullet x)) = z \bullet (y \bullet x)$, or the law of idempotence $x \bullet x = x$. This is sufficient to treat many dup equations, but some others such as the law of distributivity $(x \times y) + (x \times z) = x \times (y + z)$ are not covered. First, we sketch the method by treating a concrete example of dup equation:

$$(y \bullet (x \bullet x)) \bullet z = z \bullet (y \bullet x). \tag{5.12}$$

As we have done with idempotence, we proceed in two steps. First, we define a convenient algebra satisfying the equality. Then we derive relevance from the preservation of the equation.

Step 1: Constructing the Algebra

Note that this equality is of the desired form with $t[x] = z \bullet (y \bullet x)$. Assume T preserves (5.12) and let A be a non-empty set. Once again, we define a convenient algebra $\mathcal{A}$. For idempotence we chose A^2 as a carrier; this time we take the product A^5. We will see later that this choice gives us enough instances of A for our construction. We define the interpretation $\bullet_{\mathcal{A}} \colon A^5 \times A^5 \to A^5$ as follows. It is simply a tupling or projections, therefore it is convenient to represent it as a series of connections between inputs (the 10 copies of A in $A^5 \times A^5$) and outputs (A^5). We illustrate it graphically below.



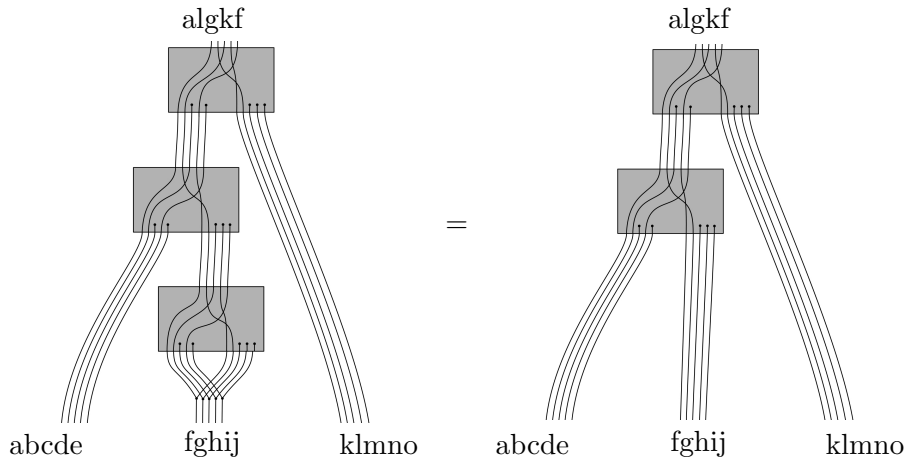
(5.13)

The map $\bullet_{\mathcal{A}}$ can be categorically defined as $\langle \pi_1, \pi_7, \pi_2, \pi_6, \pi_4 \rangle$, but we will rather describe it as the connections (seen as wires) between its inputs and outputs.

Lemma 5.12. *The binary operation m on X^5 satisfies the equation* (5.12).

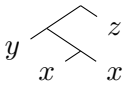
Proof. We show that the evaluations of $(y \bullet x) \bullet z$ and $(y \bullet (x \bullet x)) \bullet z$ are equal when variables x, y, z are substituted with elements of X^5 and $\bullet$ is interpreted with $\bullet_{\mathcal{A}}$. In this framework, this means that on both sides of the diagram equality below, the wires reaching the top of the diagrams are the same (we say that the outputs of both diagrams are matching). For clarity, we label the elements of A that are fed into the diagram with

letters from a to o.



(5.14)

We exploit the similarity between the diagrams above and the syntax trees of $(y \bullet (x \bullet x)) \bullet z$ and $(y \bullet x) \bullet z$. To make our illustration more explicit, we represent the syntax tree of $(y \bullet (x \bullet x)) \bullet z$ (which is almost identical to the syntax tree of $(y \bullet x) \bullet z$, up to the variable duplication).



(5.15)

Let us now shed some light on our construction. We encode positions in binary trees with words in $\{0,1\}^*$. To trace down each thread, we will label each connection of the box (5.13) with a language on the alphabet $\{0,1\}$. To understand this, we picture a binary tree where all nodes feature an instance of $\bullet_{\mathcal{A}}$ (as the diagrams (5.14), above the variable duplication), this language will represent the set of locations in the tree that are connected to the output (the root of the tree).

In our example, the wires coming out of the box (5.13) are respectively labelled $L^*, R^*, LR^*, RL^*, LRL^*$. The first wire is labelled L^* because it connects the first component of the output to the first component of the left input. Therefore, no matter how many occurrences of our box are connected in whichever way, this thread always leads to the wire on the far left. On both sides of (5.14), it connects to the input labelled a. Similarly the R^* wire goes through every occurrence of m on the far right (in our

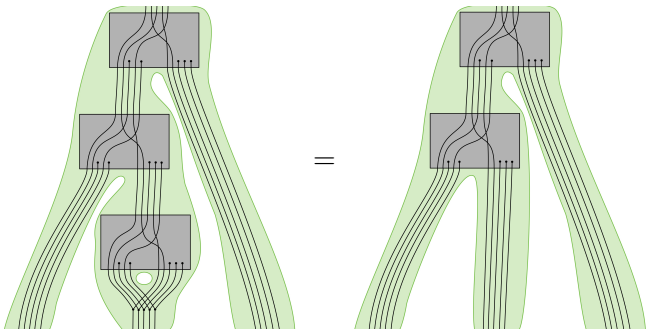
case ending up in l). The LR^* wire turns left, then connects to the R^* output of the next box: from that point, just like the previous thread, it always goes right. In the case of the left hand side term of (5.12), it connects directly to x on its second component. On the left side, it goes through one more box and connects to the input g. The next wire RL^* is connected to the L^* thread of the right input, which for both sides of the equations connects to the input k. The final wire LRL^* takes it a step further, going left then right and connecting to a L^* output. On the left hand side of the equation, it connects to f. On the right hand side, it goes through one more instance of m but also connects to f. Therefore the outputs of both sides of the equation match, in other words $\bullet_{\mathcal{A}}$ satisfies (5.11). □

As we will see in the next section, our construction of $\bullet_{\mathcal{A}}$ does not only verify the desired equation, but also provides us with two special wires in the diagrams (5.14), which will allow us to deduce the property of relevance.

Step 2: Deriving Relevance

Now that we have an algebra $\mathcal{A}$ on which our equality holds, let us consider an arbitrary monoidal monad T. We show that the preservation of our equation implies the relevance of T.

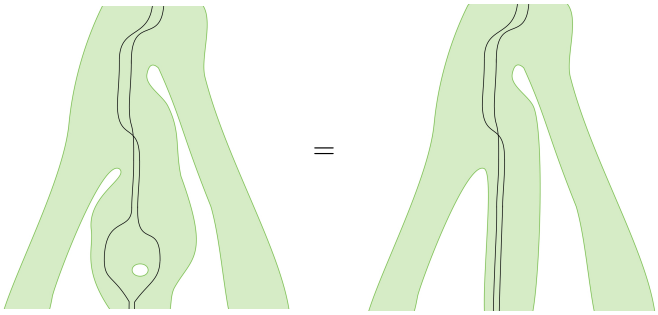
1. We assume that $(y \bullet (x \bullet x)) \bullet z = (y \bullet x) \bullet z$ is preserved by T, hence it holds on $\widehat{T}\mathcal{A}$. In other words hence the 'sleeved' version of the diagram equality (5.14) holds.



(5.16)

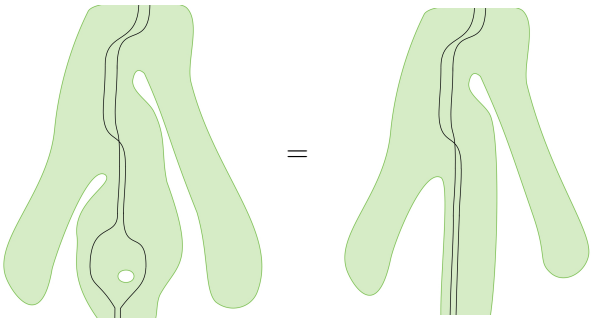
2. Now we discard many threads by pulling them to the bottom of the diagram, then deleting them (as explained in Lemma 5.9 and property (5.5)).We only conserve

two main threads of interest.



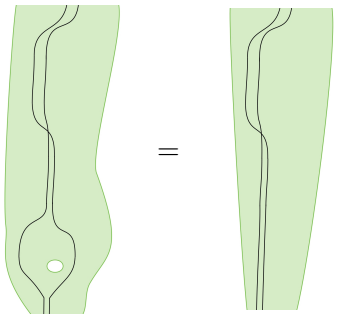
(5.17)

3. Both diagrams now figure some sleeves that only contain the object 1. Each one can now be closed by precomposing with η_1.



(5.18)

4. Each of these sleeves can now be pulled up to the top and absorbed into the main sleeve by using the property of diagram (5.4).



(5.19)

As in the case of idempotence, we now have one one side a sleeve containing two wires left untouched, and on the other side a sleeve containing two wires 'wrapping' the bubble (LR^* and LRL^*). Note that on both sides, the wires go through a swapping, but it suffices to postcompose our equality by $T\mathsf{swap}$ to obtain a very familiar picture: the property of relevance shown in diagram (5.7). The following result follows immediately from these two steps:

Theorem 5.13. *Let T be a monoidal monad. If T preserves $(y \bullet (x \bullet x)) \bullet z = (y \bullet x) \bullet z$, then T is relevant.*

5.2.4 General Case of a $t[x]$-Equation

Having understood the example of $(y \bullet (x \bullet x)) \bullet z = (y \bullet x) \bullet z$, we can now generalise the method to a general $t[x]$-equation. Let t be a linear Σ-term containing only one symbol of the signature, the binary operation $\bullet$.

In this section, we move away from sleeve diagrams and rely on more conventional category theory proof techniques. The reason for this is the lack of soundness and completeness of our diagrams : unlike McCurdy in [28], we do not assume the Frobenius property and therefore cannot guarantee the robustness of a diagrammatic proof. Showing soundness and completeness would be a promising direction for future work, but fortunately the proofs needed in this thesis can be expressed using category theory techniques.

Step 1: Constructing the Algebra

As we did in the example, we start by defining a convenient algebra on which the $t[x \bullet x] = t[x]$ holds. Recall the natural transformations $\delta^V_{\mathcal{A}}$ and $\gamma^V_{\mathcal{A}}$ from Definitions 4.1 and 4.3. First, let us define a few constructs involving the term t.

Note that in $t[x \bullet x]$, only one variable duplication occurs. Without loss of generality, we can reorder our variables to make x the first one. Then we have $\delta^V_{\mathcal{A}}(t[x \bullet x]) = \beta_1 \circ (\Delta \times \mathrm{id} \times \cdots \times \mathrm{id})$ with β_1 a natural isomorphism which operates a permutation (as seen in Lemma 4.23). $t[x]$ shows no duplicated variable, hence $\delta^V_{\mathcal{A}}(t[x])$ is a similar natural isomorphism that we note β_2. We note $\gamma_1 = \gamma^V_{\mathcal{A}}(t[x \bullet x])$ and $\gamma_2 = \gamma^V_{\mathcal{A}}(t[x])$. We denote by n the number of variables in the equation, and therefore we have $\mathrm{Arg}(t[x \bullet x]) = n+1 > 1$ and $\mathrm{Arg}(t[x]) = n > 0$. We can now introduce a lemma that characterises what we require from a 'convenient' algebra.

Lemma 5.14. *For every non-empty set A, there exists a Σ-algebra $\mathcal{A}$, a natural number $k \in \mathbb{N}$ and a natural isomorphism $s \colon (-)^k \to (-)^k$ such that*

(i) *$\mathcal{A}$ has carrier A^k*

(ii) *$t[x \bullet x] = t[x]$ holds in $\mathcal{A}$*

(iii) *$s \circ \gamma_1 \circ \beta_1 = \alpha_1$ and $s \circ \gamma_2 \circ \beta_2 = \alpha_2$, where:*

$$\alpha_1 = \langle \pi_{i_1}, \ldots, \pi_{i_k} \rangle \colon (A^k)^{n+1} \to A^k \quad ; \quad \alpha_2 = \langle \pi_{j_1}, \ldots, \pi_{j_k} \rangle \colon (A^k)^n \to A^k$$

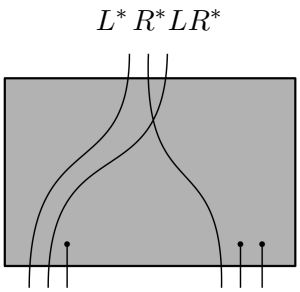
and where $i_1 = 1, i_2 = k{+}2, i_l \neq 1, 2, k{+}1, k{+}2$ for $l > 2$, and $j_1 = 1, j_2 = 2, j_l \neq 1, 2$ for $l > 2$.

Our condition (iii) seems very technical, but it has an intuitive meaning. Recall the previous example and Diagram (5.16): in order to derive later the property of relevance, we must ensure that two specific threads reach the top of the diagram, one of each side of the 'bubble'. The tuplings α_1 and α_2 describe the threads reaching the top, starting above the 'bubble'. On the left side (corresponding to $t[x \bullet x]$), there are $n_1 - 1$ inputs of type A^k. After duplicating of the first input (thus obtaining the $n + 1$ arguments of the term), we want to keep track of two main threads: the first element of the first copy (hence α_1 must include π_1), and the second element of the second copy (corresponding to the projection π_{k+2} in α_1). For the case of $t[x]$, the situation is simpler as there is no duplication: we keep track of the very first element by requiring $j_1 = 1$, and of the second element with $j_2 = 2$. As we saw in Diagram (5.17), our two remaining threads are not necessarily the first two outputs of the diagram; they may even be swapped together. We can then permute all outputs by postcomposing both sides with a natural isomorphism s to ensure that our main threads are now placed first.

Proof. The idea of this lemma is simple: for every $t[x]$-equation, for every set A, we can build a convenient algebra $\mathcal{A}$ whose carrier is a product of several copies of A and where $t[x \bullet x] = t[x]$ holds. To construct this algebra, we build an interpretation $\bullet_{\mathcal{A}} \colon A^k \times A^k \to A^k$ for the unique binary operation used in t. We proceed as in (5.13). Let $w = w_1 \dots w_l \in \{L, R\}^*$ be the word representing the position of the variable duplication in $t[x] = t[x \bullet x]$. Again, we illustrate our interpretation of $\bullet$ as a box with inputs and outputs, which we connect with labelled wires. We use the notation $O_1, O_2 \dots$ for the outputs, $L_1, L_2, \dots$ for the left inputs and $R_1, R_2, \dots$ for the right ones. First, we connect L_1 to O_1, R_2 to O_2, and we respectively label these wires with the languages L^* and R^*. We then make successive connections depending on the letters of w to obtain threads labelled $w_l L^*, w_{l-1} w_l L^*, \dots, w L^*$ and $w_l R^*, w_{l-1} w_l R^*, \dots, w R^*$.

For instance if $w_l = L$, we want one wire labelled LR^* and another one labelled LL^*. The latter is equivalent to L^*, which we already have. To obtain LR^*, we link a 'fresh' output O_3 with L_2. This wire will therefore turn left then connect with the thread

labelled R^* in the next box it encounters; hence the language of this output is LR^*.



We repeat this process until we obtain wires labelled wL^* and wR^*, which will become our two main threads leading to the variable duplication then 'wrapping' the bubble. Let k be the number of wires; by construction we have obtained an algebra $\mathcal{A}$ on A^k on which the equation holds. We now show that the property (iii) is verified. The second part of (iii), requiring that $j_1 = 1, j_2 = 2, j_l \neq 1, 2$ for $l > 2$ immediately follows from our construction. Consider the diagram corresponding to $t[x]$; its output, at the very top, features two threads wL^* and wR^* leading to the first two components of the input substituted with the variable x (which yields $j_1 = 1, j_2 = 2$). For example in (5.14), they correspond to the letters f, g. Because no thread is duplicated, all other outputs are distinct wires, hence $j_l \neq 1, 2$ for $l > 2$. Let us now consider the side of $t[x \bullet x]$: the same applies to show that $i_1 = 1, i_2 = k + 2$, by construction these two wires are the ones that wrap the bubble and make it to the top of the diagram. Let us finally show by contradiction that $i_l \neq 1, 2, k + 1, k + 2$: if this condition was not verified, then there would be at least a thread that reaches the top of the diagram and that is not part of the two main threads, but that comes from the same input as wL^* and wR^*. In this case, by construction, its label would be wL^* or wR^* (depending on which side of the bubble it goes). However we halt our construction when we obtain these two labels for the first time, therefore we will never be in the situation of duplicate labels. In the example (5.14), this appears in the fact that we have outputs f, g but no other output h, i or j. □

Step 2: Deriving Relevance

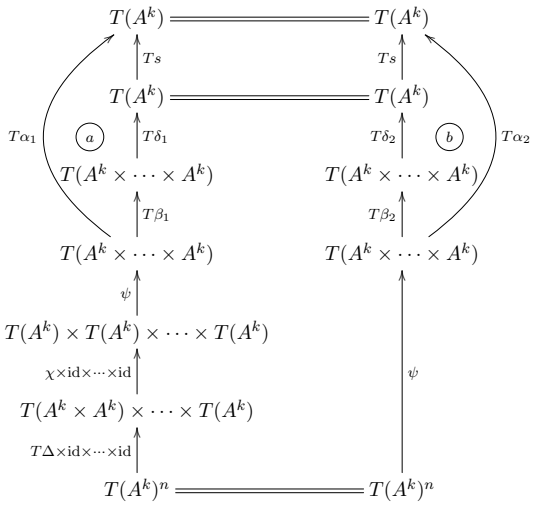
Now that we have constructed a convenient algebra and its specifications, we prove that preserving the equation $t[x \bullet x] = t[x]$ on this particular algebra implies the relevance of

T, which leads to the main result of this section.

Theorem 5.15. *Let t be a linear term with only binary operations. If T preserves $t[x] = t[x \bullet x]$ then T is relevant.*

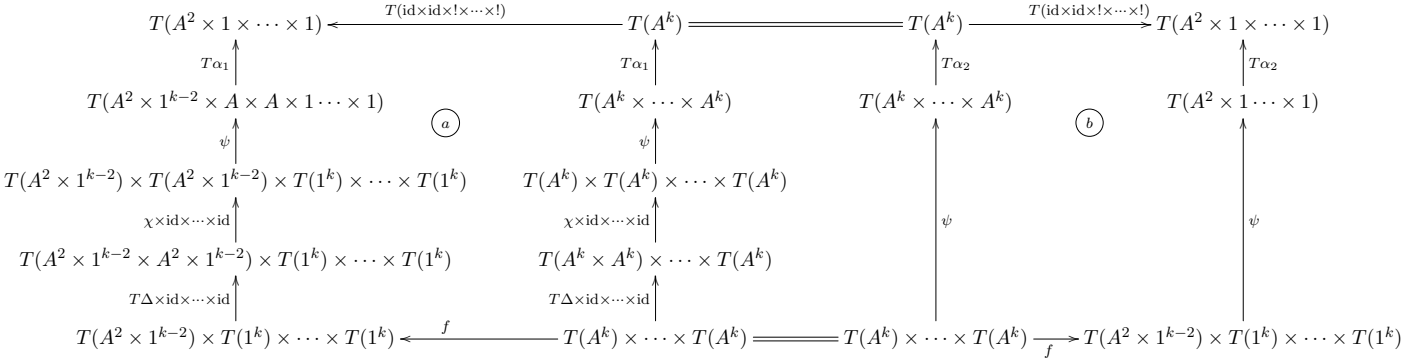
Proof. Although this proof may appear very technical because it treats the question in full generality and only uses category theory, we follow the same strategy as in diagrams (5.17) to (5.17). We consider the algebra $\mathcal{A}$ previously constructed and proceed as follows.

1. First, we state that the equation is preserved, as we did in (5.16). We use Lemma 5.14 to postcompose both sides by Ts, which rearranges threads so that our two 'useful' threads are placed first.



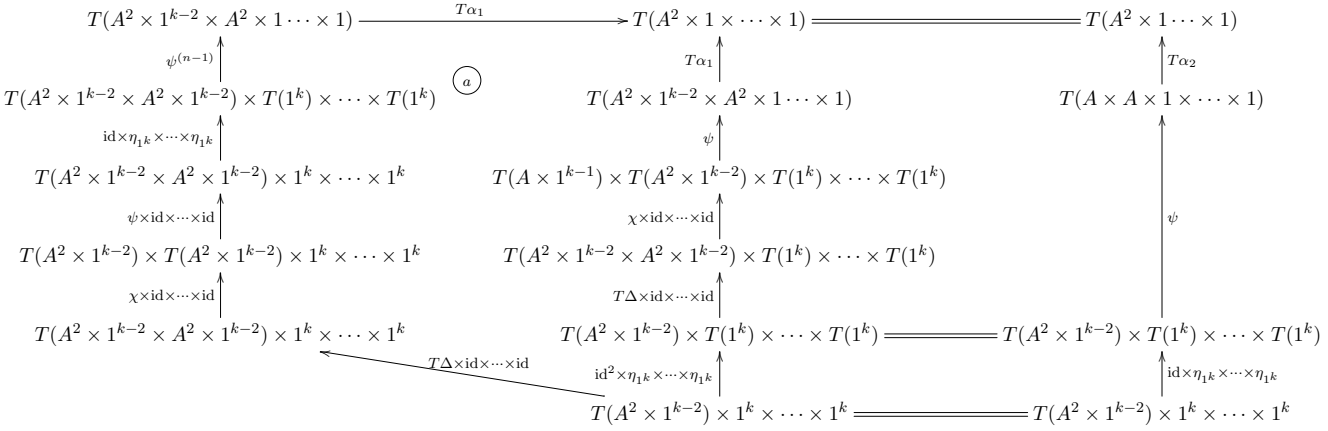
   a and b commute by Lemma 5.14.

2. As a second step, we can now drop all the unnecessary threads. This corresponds to diagrams (5.17) above. This categorical version is more cumbersome as we need to keep track of all the instances of 1 that are made implicit in (5.17).



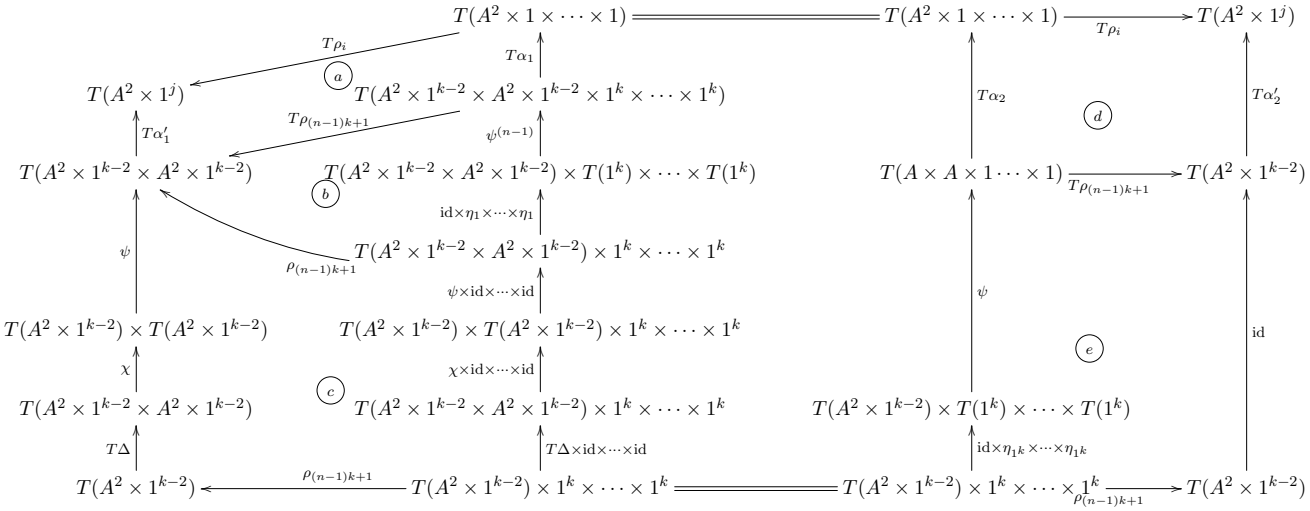
$f = T(\mathrm{id} \times \mathrm{id}\times! \times \cdots \times!) \times T(! \times \cdots \times!) \times \cdots \times T(! \times \cdots \times!)$ note that f is an epimorphism since A is non-empty. (a) and (b) commute respectively by naturality of $T\alpha_1 \circ \psi \circ (\chi \times \mathrm{id} \times \cdots \times \mathrm{id}) \circ (\Delta \times \mathrm{id} \times \cdots \times \mathrm{id})$; and by naturality of $T\alpha_2 \circ \psi$. Therefore we obtain the equality of the two vertical outer sides of the diagram.

3. Our diagram is still very intricate and contains many instances of $T(1^k) = T(1)$. In (5.17), this is represented by the 'empty sleeves'. In order to discard them, we precompose each of them with the arrow η_{1^k}. By using naturality and properties of products, we obtain the following construction, corresponding in our example to (5.18).



The commutation of (a) is immediate; all we do is decompose $\psi^{(n)}$ as $\psi^{(n-1)} \circ \psi \times \mathrm{id} \times \cdots \times \mathrm{id}$ and swap independent elements of morphism products.

4. Now we use the monoidal properties of T illustrated in diagram (5.4) to suppress the copies of $T(1^k)$, which we illustrated above as 'pulling up the empty legs' in (5.19). Because unitors are isomorphisms, we obtain the equality of the two vertical outer sides of the diagram below.

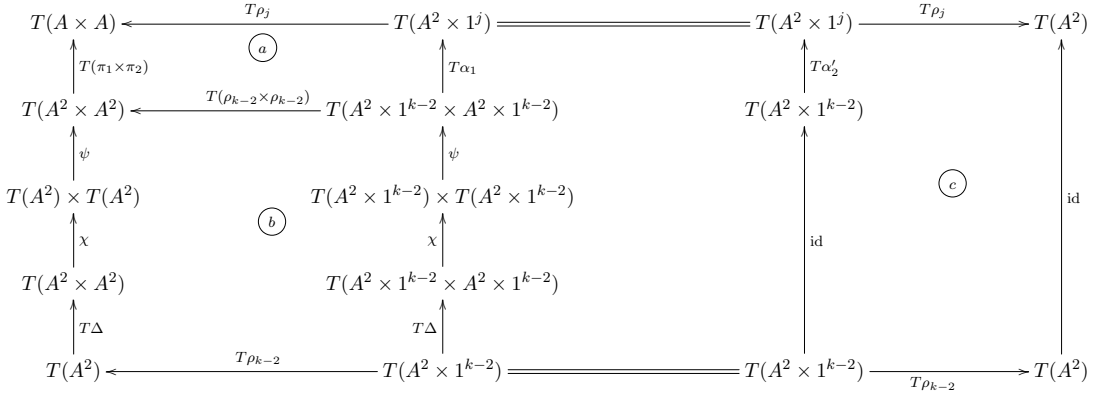


The morphism α_1 is defined in Lemma 5.14 as a tupling of projections selecting k elements in the product $(A^2 \times 1^{k-2} \times A^2 \times 1^{k-2} \times 1^k \times \cdots \times 1^k)$. Note that the condition (iii) of Lemma 5.14 implies that its codomain in this case is $(A^2 \times 1^{k-2})$ (only two elements of type A are selected).

Inside the tupling α_1, let i be the number of projection corresponding to elements from $1^k \times \cdots \times 1^k$(that is, projections π_l for $l \geq 2k$). We define α_1' as the same tupling as α_1 but without such projections. This makes sense as elements from $1^k \times \cdots \times 1^k$ are the ones that are deleted by $\rho_{(n-1)k+1}$. Finally, (a) commutes by composition of projections, as unitors equate projections in a Cartesian category. The same reasoning applies to α_2, and (d) commutes. (b) and (e) commute by Lemma 2.36; (c) commutes by naturality of ρ_n.

5. Once more, the categorical proof must cover some properties left implicit in the illustration (5.18); namely discarding the unnecessary instances of 1 inside the

object $T(A^2 \times 1^{k-2})$. This is easily achieved using naturality and properties of the unitors.



(a), (c) by the same reasoning as above, i.e. composition of unitors and projections. (b) commutes by naturality of $\psi \circ \chi \circ T\Delta$. Again because ρ_{k-2} is an isomorphism we obtain the equality $T(\pi_1 \times \pi_2) \circ \psi \circ \chi \circ T\Delta = \mathrm{id}$

6. Finally, observe that this equality is exactly the one of Diagram (5.8). Hence by Lemma 5.10, T is relevant.

□

This result may seem very specialised since we only consider terms featuring a single operation symbol. However, the process described above actually applies to other equations. Our strategy relies on defining a convenient algebra to derive relevance from the preservation of a particular equation, in particular we define the interpretation $\bullet_{\mathcal{A}}$ of a binary operation. If T preserves $(x + x) \bullet y = x \bullet y$, in particular T preserves it on an algebra where $\bullet$ and $+$ are interpreted as identical. Therefore our method for $(x \bullet x) \bullet y = x \bullet y$ also applies to $(x + x) \bullet y = x \bullet y$. We may generalise this further to treat operations of any arity greater than 2 (in other words, anything but constants): if $f(x, x, z) = x \bullet z$ is preserved by T, then in particular it is on an algebra where $f(x, x, z) = (x \bullet x) \bullet z$ (as long as we can define such an algebra where the considered equation also holds). Since we have treated $(x \bullet x) \bullet z = x \bullet z$ above, our conclusion also applies to $f(x, x, z) = (x \bullet x) \bullet z$. We have obtained the property of relevance from any possible case featuring binary operations, thus we also obtain for free the case of n-ary operations (where $n > 1$).

Theorem 5.16. *Let t be a linear term with no constants. If T preserves $t[x] = t[x \bullet x]$ then T is relevant.*

Our method can therefore be applied regardless of the symbols involved in t, provided that the term is constant-free. For this reason, we can now omit operation symbols in our equations. For instance we refer to our example as $(yx)z = (y(xx))z$.

Again, because $t[x] = t[x \bullet x]$ is a strict-dup equation we can combine Theorems 5.16 and 4.34 to obtain the following equivalence.

Theorem 5.17. *Let t be a linear term with no constants, T a monoidal monad.*

T is relevant if and only if T preserves $t[x] = t[x \bullet x]$.

Let us now outline an even more ambitious generalisation to cover equations outside the strict-dup class. If we use the above approach to treat an equation $t = t[x \bullet x]$, it turns out we can slightly modify the equation without affecting our result. Consider the example $(y \bullet x) \bullet z = (y \bullet (x \bullet x)) \bullet z$ again (which we will write $(yx)z = (y(xx))z$). At the end of our process, the only component of the y that is connected to the output is the first one (through the L^* wire). By construction, adding another iteration of $\bullet_{\mathcal{A}}$ with y on its left input would not change this fact. Let us then substitute y with yv in the equation (for v any new variable). The position of v is coded as the word LLR, which does not belong to the language of any of our outputting wires, therefore v has no influence on the matching of the outputs. In other words, the definition of m allowing to prove relevance from the preservation of $(yx)z = (y(xx))z$ also applies to $((yv)x)z = (y(xx))z$, which is a one-drop equation. One could even substitute v with a more complicated term to obtain another equation, whose preservation would still lead to relevance. This last theorem applies therefore to many equalities outside the case $t[x] = t[x \bullet x]$, even though the exact class described by these modifications is cumbersome to define.

For instance, because Theorem 5.16 applies to $z(xx) = zx$, it also applies to $z(xx) = (zy)x$.

Theorem 5.18. *Let T a monoidal monad. T is relevant if and only if T preserves $z(xx) = (zy)x$.*

5.2.5 n-relevance

We have shown in Theorem 5.11 that the preservation of $x \bullet x = x$ implies relevance. What about the preservation of $f(x, x, x) = x$? We will see that it does not imply relevance, but rather a property of 3-relevance. To define n-relevance in general, we introduce n-ary variations of existing maps: $\Delta^{(n)}$ is the n-times duplication operator $X \to X^n$ and $\chi^{(n)} \equiv \langle T\pi_1, \ldots, T\pi_n \rangle$.

Definition 5.19. Let T be a monoidal monad, T is n-relevant if the following diagram commutes

$$\begin{array}{ccc} TA & \xrightarrow{\Delta^n} & (TA)^n \\ & \searrow^{T\Delta^n} & \downarrow{\scriptstyle \psi^{(n)}} \\ & & T(A^n) \end{array} \tag{5.20}$$

Or equivalently iff $\psi^{(n)} \circ \chi^{(n)} = \mathrm{id}$.

Note that the usual notion of relevance corresponds to 2-relevance.

Theorem 5.20. *Let M be a commutative monoid and consider the corresponding writer monad. $M \times X$ is n-relevant iff $w^n = w$ for all $w \in M$.*

Proof. Let $w \in M$. We have:

$$\begin{aligned} (\psi^{(n)} \circ \chi^{(n)})(w, (x_1, \ldots, x_n)) &= \psi^{(n)}((w, x_1), \ldots, (w, x_n)) \\ &= (w^n, (x_1, \ldots, x_n)) \end{aligned}$$

Hence $M \times X$ is n-relevant iff $w^n = w$. □

Therefore some monads may be n-relevant but not m-relevant for any m with $n > m$; we now show an example of such a monad.

Example 5.21. For all $x \in \mathbb{Z}_2$ we have $x + x + x = x \times 3 = x$ (note that despite the difference in additive and multiplicative notations, this corresponds exactly to the property $w^n = w$ used in the previous theorem). Therefore $X \times \mathbb{Z}_2$ is 3-relevant; however $1 \times 2 = 2 \neq 1$ hence $X \times \mathbb{Z}_2$ is not relevant. △

Both notions of relevance are however related.

Proposition 5.22. *Relevance implies n-relevance for any $n \geq 2$.*

Proof. We proceed by induction on n, the case $n = 2$ is trivial. We assume that T is

relevant and $(n-1)$-relevant, we show that it is n-relevant.

$$\begin{aligned}
&\psi^{(n)} \circ \Delta^{(n)} \\
&= \psi \circ (id \times \psi^{(n-1)}) \circ (id \times \Delta^{(n-1)}) \circ \Delta \\
&= \psi \circ (id \times T\Delta^{(n-1)}) \circ \Delta && (n-1)\text{-relevance} \\
&= (id \times T\Delta^{(n-1)}) \circ \psi \circ \Delta && \text{naturality} \\
&= (id \times T\Delta^{(n-1)}) \circ T\Delta && \text{relevance} \\
&= T\Delta^{(n)} && \square
\end{aligned}$$

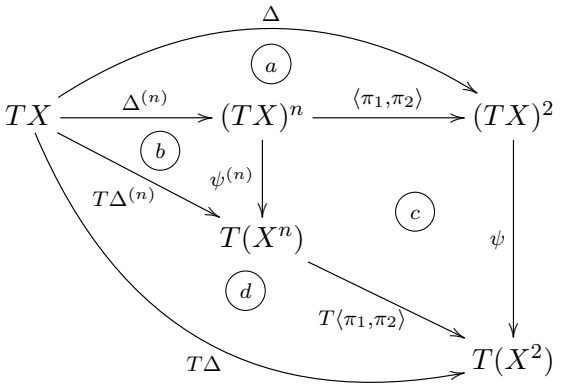
Although n-relevance is weaker than relevance, these two properties are sometimes connected. As it turns out, they are equivalent in the case of affine monads.

Proposition 5.23. *Given any $n > 2$, if T is n-relevant and affine, then T is relevant.*

Proof. Let $n \in \mathbb{N}$. By Lemma 4.27, an affine monad T verifies $\psi \circ \langle \pi_1, \pi_2 \rangle = T\langle \pi_1, \pi_2 \rangle \circ \psi^{(n)}$ because this equality corresponds to the residual diagram of the drop equation $(x_1 \bullet (x_2 \bullet \ldots (x_{n-1} \bullet x_n) \ldots)) = x_1 \bullet x_2$. Then we have:

$$\begin{aligned}
\psi \circ \Delta &= \psi \circ \langle \pi_1, \pi_2 \rangle \circ \Delta^n \\
&= T\langle \pi_1, \pi_2 \rangle \circ \psi^{(n)} \circ \Delta^n && \text{affineness} \\
&= T\langle \pi_1, \pi_2 \rangle \circ T\Delta^n && n\text{-relevance} \\
&= T\Delta && \square
\end{aligned}$$

Therefore T is relevant. For more clarity, we express this proof as a diagram.



(a) and (d) commute trivially ; (b) commutes by n-relevance and (c) commutes by affineness as expressed above. Therefore the outer triangle commutes, which shows n-relevance. Because affine monads are well-behaved towards projections (and pairings or projections), one can reduce the property of n-relevance to the conventional notion of relevance.

Finally, we show that n-relevance is a necessary and sufficient condition to preserve certain equations:

Theorem 5.24. *Assume Σ features an n-ary operation f^n. Then a monoidal monad T preserves $f^n(x,\ldots,x) = x$ if and only if T is n-relevant.*

Proof. ($\Leftarrow$): Consider an algebra $\mathcal{A}$ where $f^n(x,\ldots x) = x$ holds. We make use of Theorem 4.11 and show that the corresponding residual diagrams commute. $\mathrm{id} \circ \delta^V_{\widehat{T}\mathcal{A}}(x) = T\delta^V_{\mathcal{A}}(x) \circ \mathrm{id}$ trivially holds because $\delta^V_{\mathcal{A}}(x) = \delta^V_{\widehat{T}\mathcal{A}}(x) = \mathrm{id}$. Furthermore, note that $\delta^V_{\mathcal{A}}(f^n(x,\ldots x)) = \delta^V_{\widehat{T}\mathcal{A}}(f^n(x,\ldots x)) = \Delta^n$, therefore by n-relevance we have: $\psi^n \circ \delta^V_{\widehat{T}\mathcal{A}}(f^n(x,\ldots x)) = T\delta^V_{\mathcal{A}}(f^n(x,\ldots x)) \circ \mathrm{id}$. By Theorem 4.11, the equation is preserved.

($\Rightarrow$): We assume that $f^n(x,\ldots x) = x$ is preserved on every algebra. Then in particular, it is preserved for the n-ary operation $f^n = \pi_1 \times \cdots \times \pi_n$ defined on a set A^n. The equation holds, in other words we have $(\pi_1 \times \cdots \times \pi_n) \circ \Delta^n = \mathrm{id}$. Then by the same reasoning as in Theorem 5.11, we obtain:

$$\begin{aligned}
\mathrm{id} &= Tf^n \circ \psi^n \circ \Delta^n \\
&= Tf^n \circ \psi^n \circ \chi^n \circ T\Delta^n \\
&= T(\pi_1 \times \cdots \times \pi_n) \circ \psi^n \circ \chi^n \circ T\Delta^n \\
&= \psi^n \circ (T\pi_1 \times \cdots \times T\pi_n) \circ \chi^n \circ T\Delta^n \\
&= \psi^n \circ \chi^n \circ T(\pi_1 \times \cdots \times \pi_n) \circ T\Delta^n \\
&= \psi^n \circ \chi^n \circ T((\pi_1 \times \cdots \times \pi_n) \circ \Delta^n) \\
&= \psi^n \circ \chi^n
\end{aligned}$$

□

5.3 Discussion and Related Work

This chapter was centered on finding necessary conditions for preserving diverse classes of equations. In the case of affine monads, we established the converse of our results from Chapter 4: if a monad T preserves any strict-drop equation, then it must be affine. Our theorem is even stronger, as it also holds for some equations outside the strict-drop class (namely one-drop equations). For the case of dup equations, a variety of cases may occur. When such equalities can be written as $t[x \bullet x] = t[x]$ for t a term containing no constant, we have shown that preservation implies relevance. For other equations such as $f(x, x, \ldots, x) = x$, it turns out that relevance is not necessary; instead it is the notion of n-relevance that characterises monads preserving this equation.

In light of these results, a remark on the topic of equation preservation arises naturally. We have seen that if T preserves the theory Σ, E presenting S, then there exists a distributive law $\lambda\colon ST \to TS$ (Theorem 3.14). The converse does not hold, that is, the existence of such a distributive law does not necessarily mean that T preserves E. The point is that S might have several different presentations by operations and equations, and the notion of preservation makes explicit use of the presentation at hand. Consider for example T to be a non-affine, relevant monad (for instance $X + 1$), and $S = \mathcal{D}$ the distribution monad. Algebras for $\mathcal{D}$ are convex algebras, which are presented by several equivalent theories (see Examples 2.49 and 2.50). We first consider the theory $\mathbb{T}_1$, whose signature contains binary convex combination symbols $\oplus_\lambda$ for $\lambda \in [0, 1]$, and whose equations include the *projection* axioms $\oplus_0(x, y) = y$ and $\oplus_1(x, y) = x$ (see for instance [3] for more details). These two laws are one-drop equations and cannot be preserved by T. But $\mathcal{D}$ is also presented by $\mathbb{T}_2$ (seen in example 2.49), whose signature only contains convex combination operators $\oplus_\lambda$ for $\lambda \in]0, 1[$. Therefore $\mathbb{T}_2$ has no projection axioms; all its equations are linear or dup. By Theorem 4.34, the relevance of T is sufficient to preserve $\mathbb{T}_2$, hence there exists a distributive law $\mathcal{D}T \to T\mathcal{D}$. Indeed, the failure to preserve one presentation of $\mathcal{D}$ does not mean that no distributive law can be found.

Once more, these results confirm Gautam's findings on the powerset monad [11] and allow us to answer the question we asked at the end of Chapter 4. The monad $\mathcal{P}$ does not preserve drop equations because it is not affine, and does not preserve the law of idempotence because it is not relevant. Gautam's original counterexamples showing that drop equations are not preserved by $\mathcal{P}$ are also our inspiration for the definition of the map α in Lemma 5.14. The incompatibility between $\mathcal{P}$ and the law of idempotence is also a central notion of Klin and Salamanca's work in [21]. In this paper, the authors settle a long-standing question of category theory: does the powerset monad distribute over itself? The answer comes after the numerous historical mistakes mentioned in Chapter 3, and is negative. A connection between their work and ours can be observed in this paper: first, recall that the presentation of $\mathcal{P}$ contains an idempotent binary operation, which cannot be preserved by $\mathcal{P}$ as it is not a relevant monad (Theorem 5.11). In an analogous reasoning, Klin and Salamanca define the concept of *nontrivial idempotent term*, a categorical object which prevents any distributivity over the powerset monad. Their proof strategy, which ultimately allowed to prove that $\mathcal{P}$ cannot be composed

with itself, is based on an idea of Gordon Plotkin, and used in a similar manner for the case of $\mathcal{P}$ and $\mathcal{D}$ by Varacca and Winskel in [38]. Bringing this idea further, Zwart and Mardsen show later in [42] that under reasonable conditions, any monad whose presentation includes idempotent operations cannot be distributed over $\mathcal{P}$ (and more generally, over any monad that verifies certain algebraic conditions). Idempotence seems to be a clear obstacle to the composition with the powerset monad; our work generalises this idea to the entire class of non-relevant monads, which cannot preserve this equation (by contraposition of Theorem 5.11). However, one distinction between our work and [42] must be made. In the case of $\mathcal{P}$ and of a monad presented with an idempotent operation, we have shown that preservation fails. But there could exist other distributive laws, not based on a monoidal approach, allowing to compose the considered monads. We will explore some examples of alternative distributive laws in Chapter 6. The work of Zwart and Marsden goes beyond the monoidal setting and establishes clear properties of incompatibilities between the algebraic presentations of some monads.

Chapter 6

Strategies For Non-Composing Monads

The previous chapters have explored diverse aspects of our method for building distributive laws. We have seen in Chapter 3 that monoidal monads provide a structure allowing for a canonical construction of a distributive law by preservation of algebraic features. Then Chapters 4 and 5 respectively establish sufficient and necessary conditions for preservation of equations. We can now precisely pinpoint the cases in which our method succeeds: for instance, composing an affine monad with a monad presented by linear and strict-drop equations. Similarly, we are now certain that some compositions will not be possible in our monoidal framework. Consider a non-relevant monad, such as $\mathcal{P}$, and a theory containing the equation $x + x = x$, for instance the theory of convex algebra presenting $\mathcal{D}$. By the contraposition of Theorem 5.11, we know that $\mathcal{P}$ does not preserve this equation. If we want to compose $\mathcal{P}$ and $\mathcal{D}$, we find ourselves faced with an obstacle.

In the current chapter, we focus on cases where a monad T does not preserve the features of S, and we present a series of methods to overcome this problem by slightly modifying our monads. The first strategy is the simplest, but is worth mentioning: removing the problematic equations from the presentation of S, forming a monad S' instead and allowing to successfully build the composition TS'. The second method operates modifications on T by enforcing relevance or affineness; thus allowing to preserve a whole class of equations. The third method is the most precise: still focusing on T, we can tweak the monad to ensure that it preserves a particular equation on a fixed algebra. This necessitates a rather technical construction but results in a monad preserving the algebraic structure of S. Finally, we present in some cases one last method: constructing an ad-hoc lifting of algebraic features without using the monoidal structure of the considered monad.

6.1 Removing Faulty Equations

Our first method follows a simple idea. Let T and S be two monads that we want to combine to form TS. By examining the properties affineness, relevance and n-relevance of T and the conditions on equations given in Chapters 4 and 5 we can pinpoint any obstacle to monad composition. As signatures are always preserved by monoidal monads, the only possible issues result from equations. Let (Σ, E) be a presentation of S, and let us assume that some equations are not preserved by T. We denote by E' the set obtained from E by removing the faulty equations. We can now define S' as the free model monad of the theory (Σ, E'). By construction, T preserves this theory, therefore by Theorem 3.5, there exists a distributive law $S'T \to TS'$ and the corresponding composite monad TS'.

Example 6.1. Let us try to compose $\mathcal{P}_f$ with itself. $\mathcal{P}_f$ is not relevant, therefore it cannot preserve the equation of idempotence in its presentation. As a matter of fact, we know that $\mathcal{P}_f\mathcal{P}_f$ is not a monad (see [21]). In the monad S, we can discard this equation to obtain a slightly different monad, presented by the theory of commutative monoids: it is the multiset monad $\mathcal{M}$. Having removed the only difficulty, we can now flawlessly compose $\mathcal{P}_f$ and $\mathcal{M}$. △

Note that this method has been put in effect by Varacca and Winskel in [38], where the authors remove the equations of idempotence from the theory presenting $\mathcal{D}$ in order to compose $\mathcal{P}$ with their newly obtained monad of *indexed valuations*.Of course, this strategy requires to operate an important modification on one of the monads; the next sections will present finer methods.

6.2 Enforcing Affineness and Relevance

We know from Chapter 5 that in order to preserve certain equations, a monad may be required to be affine or relevant. When a monad T does not preserve some equations presenting S, we present a new strategy: enforcing composition by restricting T to its affine or relevant part, which we define below.

6.2.1 Affine part of a monad

Definition 6.2 (Affine Part of a Monad)**.** [24] Let (T, η, μ) be a monad on **Set**. There exists a monad (T_a, η^a, μ^a) on **Set**, called the affine part of T, together with a monad morphism $v^a \colon T_a \to T$, determined both uniquely up to isomorphism, such that for every

object X the diagram (6.1) is an equalizer diagram.

$$\begin{array}{ccccc} T_aX & \xrightarrow{v^a_X} & TX & \xrightarrow{T!} & T1 \\ & & \downarrow{\scriptstyle !} & \nearrow{\scriptstyle \eta_1} & \\ & & 1 & & \end{array} \tag{6.1}$$

Note that this construction can be done in any *finitely complete category*, in other words a category admitting all finite limits (which is the case for **Set**).

If a monad S is presented by linear and strict-drop equations and T is a non-affine monad, our method is to restrict T to its affine part T_a. By Theorem 4.28, preservation is ensured and the composition T_aS forms a monad.

Example 6.3. Let us consider the theory (Σ, E) where $\Sigma = \{\bullet, 0\}$ and E only contains the law of absorption $x \bullet 0 = 0$. We denote by S the free model monad of (Σ, E), and consider the powerset monad $\mathcal{P}$. By the contraposition of Theorem 5.3, $\mathcal{P}$ does not preserve absorption because it is not affine. We can then define the affine part of $\mathcal{P}$: for a set X, $\mathcal{P}_a(X)$ is composed of the subsets U of X such that $T!(U) = \eta_1 \circ !(U)$. It is clear that this equality is verified for every $U \neq \emptyset$, therefore the affine part of $\mathcal{P}$ is the non-empty powerset $\mathcal{P}^+$. By Theorem 4.28, $\mathcal{P}^+$ composes with S as it preserves all its features. △

6.2.2 Relevant part of a monad

In a similar fashion as the affine case, we can restrict a monad to its relevant part.

Definition 6.4 (Relevant Part of a Monad)**.** [15] Let (T, η, μ) be a commutative monad on $\mathbf{C}$. There is a monad (T_r, η^r, μ^r) on $\mathbf{C}$, called the relevant part of T, together with a monad morphism $v^r : T' \to T$, determined both uniquely up to isomorphism, such that for every object X the diagram (6.2) is an equalizer diagram.

$$\begin{array}{ccccc} T_rX & \xrightarrow{v^r_X} & TX & \xrightarrow{T\Delta} & T(X \times X) \\ & & \downarrow{\scriptstyle \Delta} & \nearrow{\scriptstyle \psi} & \\ & & TX \times TX & & \end{array} \tag{6.2}$$

Example 6.5. Consider $(M, \bullet, 1)$ monoid and the monad $\mathcal{W}_M(X) = M \times X$. In the same manner as in Example 4.32, we see that the equalizer of the above diagram is the

subset of $\mathcal{W}_M(X)$ made of pairs (x, m) such that $m \bullet m = m$. Therefore the relevant part of T is $M' \times (-)$ where M' is the greatest join-semilattice included in M, in other words the elements m of M such that $m \bullet m = m$. △

If a monad S is presented by linear and strict-dup equations and T is a non-affine monad, we can restrict T to its relevant part T_r. By Theorem 4.34, preservation of all equations presenting S is ensured and the composition $T_r S$ forms a monad.

Example 6.6. $\mathcal{P}$ is not relevant, and its relevant part is the maybe monad $X + 1 = \bot$. The problem of $\mathcal{P}\mathcal{P}_f$ can be addressed in the following way: if the $\mathcal{P}$ was relevant, then composition would succeed. Thus we substitute this $\mathcal{P}$ with $\bot$ to obtain the monad $\bot\mathcal{P}_f$. This is however quite an extreme solution. Although the monads do compose, we have discarded most of the components of one of them: its algebraic theory now only consists of one constant and no axiom. △

Finally, we can combine those techniques to enforce relevance and affine at the same time by restricting a monad to its Cartesian part.

Definition 6.7 (Cartesian Part of a Monad)**.** [24] A monad that is relevant and affine is called *Cartesian*. For T monoidal monad, the Cartesian part T_c of T is both the affine part of T_r and the relevant part of T_a.

When trying to compose T and S, we can therefore always replace T with T_c and obtain a successful composition via Theorem 4.38. However, in most cases the Cartesian part of T becomes too trivial to be useful in our construction, as the following example shows.

Example 6.8. Let S be the free model monad corresponding to the theory (Σ, E) with $\Sigma = \{\bullet, 0\}$ and E the set of equations containing $x \bullet x$ and $x \bullet 0 = 0$. We know from Chapter 4 that being Cartesian is sufficient to preserve this theory. If we want to compose $\mathcal{P}$ with S, we can therefore consider the Cartesian part of $\mathcal{P}$. $\mathcal{P}_c = (\mathcal{P}_r)_a = (\bot)_a = \mathrm{Id}$, hence this construction results in the trivial identity monad. The composition succeeds, but we have lost all the content of $\mathcal{P}$. △

This method offers a general way to restrict monoidal monads to enforce equation preservation, but lacks precision in some cases. Intuitively, it is because taking the affine, relevant or Cartesian part of a monad is already a very strong restriction. The next section presents a finer strategy that focuses on the equations of one precise algebra.

6.3 Restricting to Enforce an Equation

Taking the affine or relevant part of a monad can be considered excessive: consider the case of $\mathcal{P}_f\mathcal{P}_f$ again. The outer monad $\mathcal{P}_f$ lacks the ability to preserve the equation $x + x = x$. Making it relevant enforces that not only this equation is preserved, but a whole class of axioms, namely all equalities with duplicated variables. This is because, as we explained in Chapter 3, our sufficient conditions for equation preservation, such as residual diagrams and properties of relevance or affineness, only relate to variable rearrangements and never to actual operations of the considered algebra. For an algebra $\mathcal{A} \in \mathcal{EM}(\mathcal{P}_f)$, we could therefore build a finer approach where we tweak $\mathcal{P}_f$ to preserve exactly $x +_{\mathcal{A}} x = x$. More precisely, we will build $\widehat{\mathcal{P}_f}$, lifting of $\mathcal{P}_f$ to $\mathbf{Alg}(\Sigma)$ in the same manner as in Chapter 3, then restrict this lifting to a monad on $\mathbf{Alg}(\Sigma)$ that preserves idempotency of $+_{\mathcal{A}}$. First, we define a few technical constructs involved in our method.

6.3.1 Re-interpreting Terms

Recall that in Chapter 4, we constructed an interpretation of terms by separating variable rearrangements (with the natural transformation $\delta^V_{\mathcal{A}}$) and operations (with $\gamma^V_{\mathcal{A}}$). For the purpose of the current method, we do not need to keep track of variable rearrangements anymore. Hence we define a new interpretation of terms using a more conventional inductive approach.

For all objects X and $m, n \in \mathbb{N}$, let $\phi\colon (X^m)^n \to (X^n)^m$ be the natural isomorphism that for all $1 \leq i \leq n$ makes the diagram

$$\begin{array}{ccc} (X^m)^n & \xrightarrow{\phi} & (X^n)^m \\ & \searrow^{\pi_i} & \downarrow{\scriptstyle \pi_i^m} \\ & & X^m \end{array} \tag{6.3}$$

commute, which implies that for all objects Y and $f_1, \ldots, f_n\colon X \to Y$ the diagram below commutes.

$$\begin{array}{ccc} & X^m & \\ {\scriptstyle \langle f_1^m, \ldots, f_n^m \rangle}\downarrow & & \searrow^{\langle f_1, \ldots, f_n \rangle^m} \\ (Y^m)^n & \xrightarrow{\phi} & (Y^n)^m \end{array} \tag{6.4}$$

Finally, note that given a Σ-algebra $\mathcal{A}$ with carrier A and any $n \in \mathbb{N}$, there is a Σ-algebra $\mathcal{A}^n$ with carrier A^n. Its interpretation of an operation $\sigma \in \Sigma$ is given by

$$\sigma_{\mathcal{A}^n} = (A^n)^{|\sigma|} \xrightarrow{\phi} (A^{|\sigma|})^n \xrightarrow{\sigma^n_{\mathcal{A}}} A^n \tag{6.5}$$

For more clarity throughout this section, we will denote $\sigma_{\mathcal{A}}$ by σ when no confusion is possible.

As we did in Chapter 4, we fix a finite set V of variables and a bijection $b\colon V \to \{1, \ldots, |V|\}$ (the enumeration of variables), where $|V|$ is the size of V. A *term* is either a variable $v \in V$ or $\sigma(t_1, \ldots, t_{|\sigma|})$, which is a syntactical application of an operation σ to terms $t_1, \ldots, t_{|\sigma|}$. We can now introduce our new interpretation.

Definition 6.9 (Interpretation of Terms)**.** Given a term t and a Σ-algebra $\mathcal{A}$ with carrier A, define $(\!|t|\!)_{\mathcal{A}}\colon A^{|V|} \to A$ inductively by

$$(\!|x|\!)_{\mathcal{A}} = A^{|V|} \xrightarrow{\pi_{b(x)}} A$$

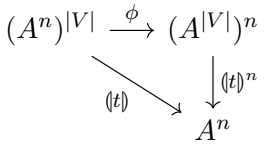
if t is a variable x and

$$(\!|\sigma(t_1, \ldots, t_{|\sigma|})|\!)_{\mathcal{A}} = A^{|V|} \xrightarrow{\langle (\!|t_1|\!)_{\mathcal{A}}, \ldots, (\!|t_{|\sigma|}|\!)_{\mathcal{A}} \rangle} A^{|\sigma|} \xrightarrow{\sigma} A$$

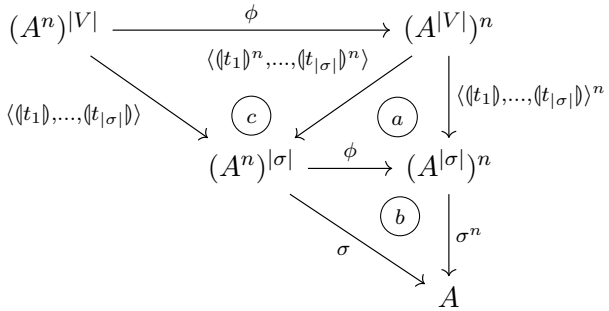
if $t = \sigma(t_1, \ldots, t_{|\sigma|})$.

For a term t, our new construct $(\!|t|\!)$ is very similar to $[\![t]\!]$, except that we do not separate variable rearrangements and algebraic operations anymore. We will show later that both interpretations are actually identical, but the definition of $(\!|t|\!)$ is more convenient in this chapter. First, we prove a series of properties of this interpretation.

Lemma 6.10. *For all Σ-algebras $\mathcal{A}$ with carrier A, terms t, and $n \in \mathbb{N}$, the diagram below commutes.*



Proof. Proof by induction on the structure of t. If $t = x \in V$, we have a special case of (6.3). For the inductive case where $t = \sigma(t_1, \ldots, t_n)$, we conclude by commutativity of



The triangle (a) commutes by the property (6.4), and (b) by (6.5). Finally, (c) commutes by induction hypothesis.

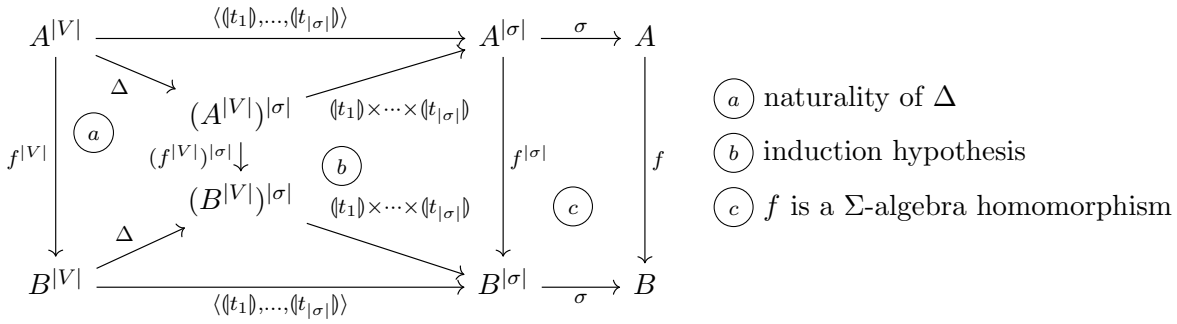
□

Lemma 6.11. *For all terms t, $(\!|t|\!)_{\mathcal{A}}$ is natural in $\mathcal{A}$.*

Proof. Consider a Σ-algebra homomorphism $f\colon \mathcal{A} \to \mathcal{B}$, where $\mathcal{A}$ and $\mathcal{B}$ are Σ-algebras carried by A and B respectively. We will prove by induction on the structure of t that

$$\begin{array}{ccc} A^{|V|} & \xrightarrow{f^{|V|}} & B^{|V|} \\ {\scriptstyle (\!|t|\!)}\downarrow & & \downarrow{\scriptstyle (\!|t|\!)} \\ A & \xrightarrow{f} & B \end{array}$$

commutes. For $t = x \in V$ this follows directly from naturality of $\pi_{b(x)}$. Consider the case of $t = \sigma(t_1, \ldots, t_{|\sigma|})$. Then



commutes, which completes the proof. Note that Δ is the diagonal (pairing of identities).

□

This interpretation is more appropriate for the constructions we will study in this section. It is however equivalent to the one we previously constructed, as $[\![t]\!]$ and $(\!|t|\!)$ are merely made of the same morphism in a different order.

Lemma 6.12. *For an algebra $\mathcal{A}$, a set of variables V, a term t, we have $[\![t]\!]^V_{\mathcal{A}} = (\!|t|\!)^V_{\mathcal{A}}$.*

Proof. We proceed by induction on t.

- If $t = x$ then $(\!|x|\!)^V_{\mathcal{A}} = \pi_{b(x)}$, where b is the bijection enumerating variables in V as before. We have $[\![x]\!]^V_{\mathcal{A}} = \mathrm{id} \circ \pi_{b(x)} = (\!|x|\!)^V_{\mathcal{A}}$.
- If $t = \sigma(t_1, \ldots, t_n)$, we assume that $[\![t_i]\!]^V_{\mathcal{A}} = (\!|t_i|\!)^V_{\mathcal{A}}$ for $1 \leq i \leq n$. Recall that for term t_i, $[\![t_i]\!]^V_{\mathcal{A}} = \gamma^V_{\mathcal{A}}(t_i) \circ \delta^V_{\mathcal{A}}(t_i)$. For clarity we write here $\gamma^V_{\mathcal{A}}(t_i) = \gamma_i$ and $\delta^V_{\mathcal{A}}(t_i) = \delta_i$.

First, recall that $\mathrm{Arg}(t) = \mathrm{Arg}(t_1) : \ldots : \mathrm{Arg}(t_n)$, where $:$ denotes concatenation. Moreover, we recall the following property of tuplings: for all f, g, h we have $\langle f, \langle g, h\rangle\rangle = \langle\langle f, g\rangle, h\rangle = \langle f, g, h\rangle$. Therefore $\delta^V_{\mathcal{A}}(t) = \langle \delta_i, \ldots \delta_n\rangle$. We write $k = |\mathrm{Arg}(t)|, k_i = |\mathrm{Arg}(t_i)|$ and we have $k = k_1 + \cdots + k_n$.

$$
\begin{aligned}
(\!|t|\!)^V_{\mathcal{A}} &= A^{|V|} \xrightarrow{\langle (\!|t_1|\!), \ldots, (\!|t_n|\!)\rangle} A^n \xrightarrow{\sigma} A \\
&= A^{|V|} \xrightarrow{\Delta^n} (A^{|V|})^n \xrightarrow{(\!|t_1|\!) \times \cdots \times (\!|t_n|\!)} A^n \xrightarrow{\sigma} A \\
&= A^{|V|} \xrightarrow{\Delta^n} (A^{|V|})^n \xrightarrow{[\![t_1]\!] \times \cdots \times [\![t_n]\!]} A^n \xrightarrow{\sigma} A \\
&= A^{|V|} \xrightarrow{\Delta^n} (A^{|V|})^n \xrightarrow{\delta_1 \times \cdots \times \delta_n} A^{k_1 + \cdots + k_n} \xrightarrow{\gamma_1 \times \cdots \times \gamma_n} A^n \xrightarrow{\sigma} A \\
&= A^{|V|} \xrightarrow{\langle \delta_1 \times \cdots \times \delta_n\rangle} A^{k_1 + \cdots + k_n} \xrightarrow{\gamma_1 \times \cdots \times \gamma_n} A^n \xrightarrow{\sigma} A \\
&= A^{|V|} \xrightarrow{\delta^V_{\mathcal{A}}(t)} A^k \xrightarrow{\gamma_1 \times \cdots \times \gamma_n} A^n \xrightarrow{\sigma} A \\
&= A^{|V|} \xrightarrow{\delta^V_{\mathcal{A}}(t)} A^k \xrightarrow{\gamma^V_{\mathcal{A}}(t)} A \\
&= [\![t]\!]^V_{\mathcal{A}}
\end{aligned}
$$

□

6.3.2 Restricting to Enforce an Equation

Now that we have defined the interpretation $(\!|t|\!)$, we can present our third method which relies on restricting the lifting of a monad to enforce equation preservation. The idea for this strategy stems from the works of Silva, Bonchi and Sokolova in [3]. In that paper, the authors examine the composition of the powerset monad $\mathcal{P}$ with the distribution monad $\mathcal{D}$ in order to lift $\mathcal{P}$ to $\mathcal{EM}(\mathcal{D})$. It is however well-known that the composition $\mathcal{PD}$ is not a monad (see for instance [38]). In particular, our method highlights the incompatibility between the non-relevant monad $\mathcal{P}$ and the equation of idempotence in the presentation of $\mathcal{D}$. The authors of [3] make use of a clever trick to overcome this issue, that we summarise here. Recall that a $\mathcal{EM}(\mathcal{D})$ is the category of convex algebras: sets provided with a family of binary operations $\oplus_\lambda$ for $\lambda \in [0, 1]$ satisfying the following axioms:

$$
\begin{aligned}
&p \oplus_\lambda p = p \qquad p \oplus_\lambda q = q \oplus_{1-\lambda} p \\
&p \oplus_\lambda (q \oplus_\psi r) = (p \oplus_{\frac{\lambda}{\lambda + (1-\lambda)\psi}} q) \oplus_{\lambda + (1-\lambda)\psi} r
\end{aligned}
\tag{6.6}
$$

Note that this theory includes equations of idempotence, which $\mathcal{P}$ cannot preserve (Theorem 5.11). To understand the situation more precisely, consider a convex algebra $\mathcal{A}$ with carrier A. The set $\mathcal{P}A$ does not in general have the structure of convex algebra. For instance consider two distinct elements $a, b \in A$; then we see that the subset $\{a, b\}$ does not verify the equation of idempotence.

$$\begin{aligned}\{a,b\} \oplus_\lambda \widehat{\mathcal{P}}_\mathcal{A} \{a,b\} &= \{a \oplus_\lambda a, a \oplus_\lambda b, b \oplus_\lambda a, b \oplus_\lambda b\} \\ &= \{a, a \oplus_\lambda b, b\} \neq \{a,b\}\end{aligned}$$

On the other hand, if we consider the subset $\{a, b, a \oplus_\lambda b\}$, the equation of idempotence is verified:

$$\{a,b,a \oplus_\lambda b\} \oplus_\lambda \widehat{\mathcal{P}}_\mathcal{A} \{a,b,a \oplus_\lambda b\} = \{a,b,a \oplus_\lambda b\}$$

The idea is then to restrict $\mathcal{P}A$ and only keep the subsets that verify the equation of idempotence of $\oplus_\lambda$ (for every $\lambda \in]0,1[$). A quick calculation shows that such subsets are the ones that are closed under $\oplus_\lambda$ for every $\lambda \in]0,1[$, in other words convex subsets. As the authors of [3] do, we construct the *convex powerset* monad $\mathcal{P}_C$ to preserve the structure of convex algebra.

Note that this procedure bears resemblance to the construction of the affine or relevant part of a monad: once again, we operate a restriction to a coequalizer-like object to only conserve the elements that verify the desired property. However, contrary to the previous method, the current construction explicitly involves the operation $\oplus_\lambda$ from the considered algebra $\mathcal{A}$. Therefore our resulting monad must be defined on the category of convex algebras rather than **Set**. In the rest of this chapter, we formally define this procedure for an arbitrary monoidal monad T. For a Σ-algebra $\mathcal{A}$ with carrier A and a set of equations E, we assume that we can define a 'restriction' of the set TA that verifies the equations in E, and we show that this restriction has a monadic structure on the category $\mathcal{EM}(S)$.

For the purpose of this section, we highlight the following properties of monoidal monads, for $X \in \mathbf{Set}$ and $m, n \in \mathbb{N}$.

$$\begin{array}{ccc} X^n & \overset{\eta}{\searrow} & \\ {\scriptstyle \eta^n}\downarrow & & \\ (TX)^n & \xrightarrow{\psi} & T(X^n) \end{array} \qquad \begin{array}{ccc} (T^2X)^n & \xrightarrow{\mu^n} & (TX)^n \\ {\scriptstyle \psi}\downarrow & & \downarrow{\scriptstyle \psi} \\ T((TX)^n) & & \\ {\scriptstyle T\psi}\downarrow & & \\ T^2(X^n) & \xrightarrow{\mu} & T(X^n) \end{array} \qquad \begin{array}{ccc} ((TX)^m)^n & \xrightarrow{\phi} & ((TX)^n)^m \\ {\scriptstyle \psi^n}\downarrow & & \downarrow{\scriptstyle \psi^m} \\ T(X^m)^n & & T(X^n)^m \\ {\scriptstyle \psi}\downarrow & & \downarrow{\scriptstyle \psi} \\ T((X^m)^n) & \xrightarrow{\phi} & T((X^n)^m) \end{array} \tag{6.7}$$

Note that these conditions follow easily from (MF.1), (MF.2) and (MM.2). Now we move on to defining a new notion of a morphism 'satisfying equations', with a construction similar to an equalizer.

Definition 6.13. Given a Σ-algebra $\mathcal{A}$ with carrier A, a morphism $j\colon X \to TA$ is said to *satisfy* an equation $t_1 = t_2$ if the diagram below commutes.

$$\begin{array}{ccc} X^{|V|} & \xrightarrow{j^{|V|}} & (TA)^{|V|} \\ {\scriptstyle j^{|V|}}\downarrow & & \downarrow{\scriptstyle (\!|t_1|\!)} \\ (TA)^{|V|} & \xrightarrow{(\!|t_2|\!)} & TA \end{array} \tag{6.8}$$

Moreover we assume that this construct verifies a universal property: whenever there is an object J with a morphism $j\colon J \to TA$ satisfying all equations in E, then there exists a unique morphism $u\colon J \to \overline{T}A$ rendering the diagram below commutative.

$$\begin{array}{ccc} J & & \\ {\scriptstyle u}\vdots & \searrow^{j} & \\ \overline{T}A & \xrightarrow{i} & TA \end{array} \tag{6.9}$$

Now let us fix Σ and E and assume that for each (Σ, E)-algebra $\mathcal{A}$ with carrier A, there exists an object $\overline{T}A$ with a morphism $i_{\mathcal{A}}\colon \overline{T}A \to TA$ satisfying all equations in E. Intuitively, this means that we assume having found, for all algebra $\mathcal{A}$, a 'restriction' of TA where the desired equations hold, and that this restriction is maximal. So far, it is only an object that we denote by $\overline{T}A$, but we will see in the following paragraphs that this construction defines a monad.

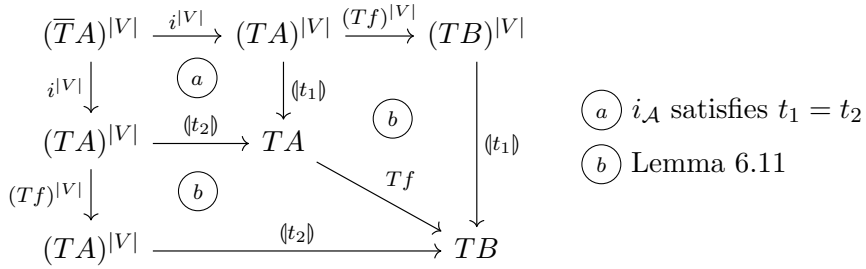
Lemma 6.14. *For any (Σ, E)-algebra $\mathcal{A}$, $i_{\mathcal{A}}$ is monic.*

Proof. Let A be the carrier of $\mathcal{A}$. Consider an object X with morphisms $f, g\colon X \to \overline{T}A$ such that $i_{\mathcal{A}} \circ f = i_{\mathcal{A}} \circ g$. Because $i_{\mathcal{A}}$ satisfies every equation in E, so does $i_{\mathcal{A}} \circ f$. Thus, there exists a unique morphism $h\colon X \to \overline{T}A$ such that $i_{\mathcal{A}} \circ h = i_{\mathcal{A}} \circ f$. Since $i_{\mathcal{A}} \circ h = i_{\mathcal{A}} \circ f = i_{\mathcal{A}} \circ g$, we have $h = f = g$. We conclude that $i_{\mathcal{A}}$ is monic. □

Let $U\colon \mathbf{Alg}(\Sigma, E) \to \mathbf{Alg}(\Sigma)$ be the functor that forgets that the equations in E hold, and let $(\widehat{T}, \widehat{\eta}, \widehat{\mu})$ be the lifting of (T, η, μ) to $\mathbf{Alg}(\Sigma)$ defined in Chapter 3. First, we show that our restriction $\overline{T}$ can be used to define a functor on $\mathbf{Alg}(\Sigma, E)$.

Lemma 6.15. *There exists a functor $\widehat{\overline{T}}\colon \mathbf{Alg}(\Sigma, E) \to \mathbf{Alg}(\Sigma, E)$ such that the collection of morphisms i constitutes a natural transformation $U\widehat{\overline{T}} \to TU$.*

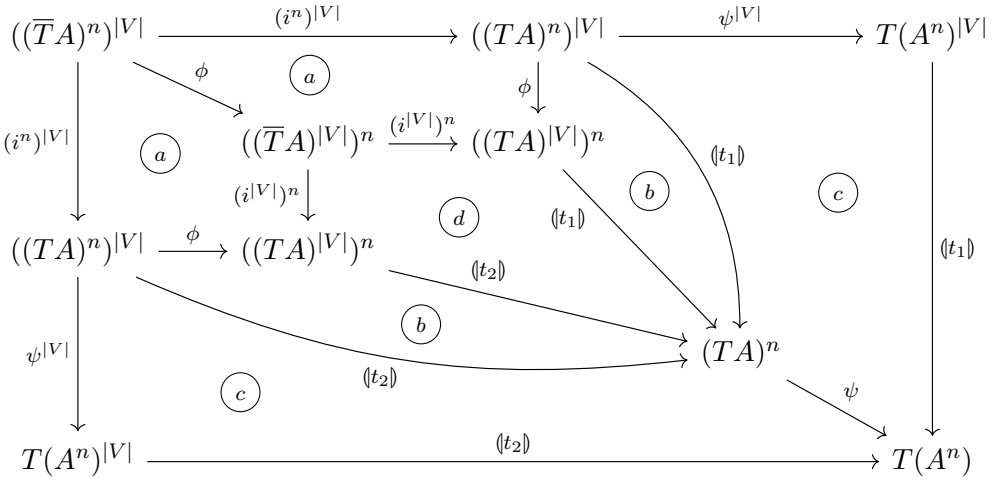
Proof. Let $\mathcal{A}$ and $\mathcal{B}$ be (Σ, E)-algebras with respective carriers A and B, and consider a Σ-algebra homomorphism $f\colon \mathcal{A} \to \mathcal{B}$ and an equation $t_1 = t_2$ in E. Commutativity of



shows that $Tf \circ i_{\mathcal{A}}$ satisfies $t_1 = t_2$, and therefore there is a unique morphism $\overline{T}f\colon \overline{T}A \to \overline{T}B$ making the diagram below commute.

$$\begin{array}{ccc} \overline{T}A & \xrightarrow{\;i\;} & TA \\ {\scriptstyle \overline{T}f}\,\vdots & & \downarrow{\scriptstyle Tf} \\ \overline{T}B & \xrightarrow{\;i\;} & TB \end{array} \tag{6.10}$$

Note that the third diagram in (6.7) says that ψ is a Σ-algebra homomorphism. Since for any (Σ, E)-algebra $\mathcal{A}$ with carrier A and all $n \in \mathbb{N}$ and equations $t_1 = t_2$ in E the diagram



commutes, we have a unique morphism $\overline{\psi}\colon (\overline{T}A)^n \to \overline{T}(A^n)$ completing the diagram

below.

$$\begin{array}{ccc} (\overline{T}A)^n & \xrightarrow{i^n} & (TA)^n \\ {\scriptstyle \overline{\psi}}\,\vdots & & \downarrow{\scriptstyle \psi} \\ \overline{T}(A^n) & \xrightarrow{i} & T(A^n) \end{array} \tag{6.11}$$

This allows us to define a Σ-algebra $\widehat{\overline{T}}\mathcal{A}$ with carrier $\overline{T}A$: for each $\sigma \in \Sigma$, define

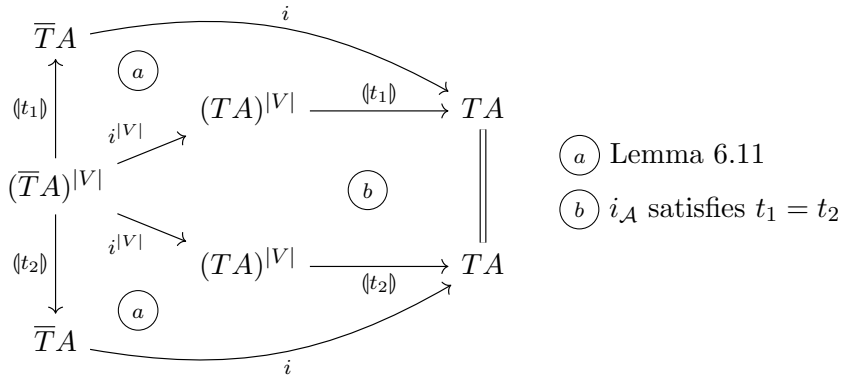
$$\sigma_{\widehat{\overline{T}}\mathcal{A}} = (\overline{T}A)^{|\sigma|} \xrightarrow{\overline{\psi}} \overline{T}(A^n) \xrightarrow{\overline{T}\sigma} \overline{T}A.$$

Composing (6.11) with (6.10) in

$$\begin{array}{ccccc} (\overline{T}A)^{|\sigma|} & \xrightarrow{\overline{\psi}} & \overline{T}(A^{|\sigma|}) & \xrightarrow{\overline{T}\sigma} & \overline{T}A \\ {\scriptstyle i^{|\sigma|}}\downarrow & & \downarrow{\scriptstyle i} & & \downarrow{\scriptstyle i} \\ (TA)^{|\sigma|} & \xrightarrow{\psi} & T(A^{|\sigma|}) & \xrightarrow{T\sigma} & TA \end{array}$$

we see that $i_{\mathcal{A}}$ is a Σ-algebra homomorphism $\widehat{\overline{T}}\mathcal{A} \to \widehat{T}\mathcal{A}$.

Consider any equation $t_1 = t_2$ in E. From commutativity of



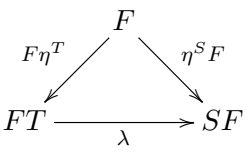
and Lemma 6.14 we find that $t_1 = t_2$ holds in $\widehat{\overline{T}}\mathcal{A}$. Therefore, $\widehat{\overline{T}}\mathcal{A}$ is a (Σ, E)-algebra.

From the uniqueness property it is standard to argue that the operation $\widehat{\overline{T}}$ on Σ-algebra homomorphisms between (Σ, E)-algebras preserves identities and composition: it follows from the fact that T preserves them. Naturality of i comes down precisely to (6.10). □

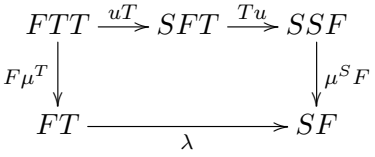
Our restriction therefore defines a functor on $\mathbf{Alg}(\Sigma, E)$. We now show how to provide it with a monadic structure. For this purpose and to later prove the uniqueness

of this monad, we make a brief use of the notion of *oplax morphism of monads* and refer the reader to [41] for more details on this construct. In a nutshell, it defines a natural transformation verifying similar properties to a $\mathcal{K}l$-law.

Definition 6.16. An oplax morphism of monads is a natural transformation $u\colon FT \to SF$, for F a functor and S, T two monads, such that the following diagrams commute.



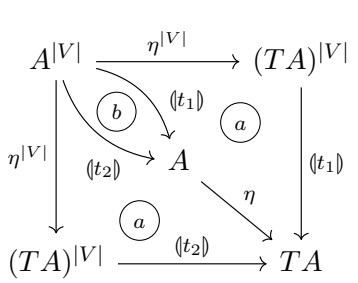
(6.12)



(6.13)

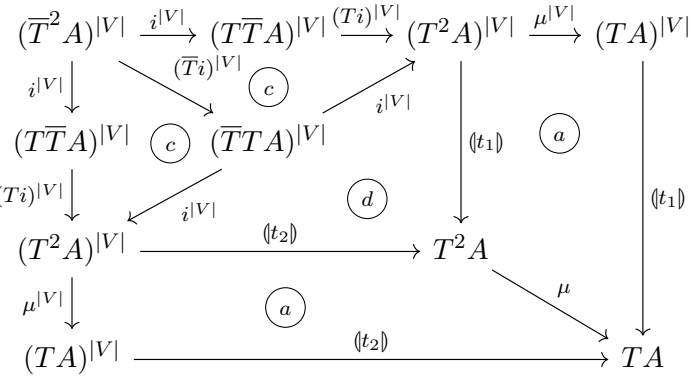
Theorem 6.17. *The functor $\widehat{\overline{T}}$ forms a monad on $\mathbf{Alg}(\Sigma, E)$, in such a way that the natural transformation $i\colon U\widehat{\overline{T}} \to TU$ is an oplax morphism of monads.*

Proof. Consider any (Σ, E)-algebra $\mathcal{A}$ with carrier A and an equation $t_1 = t_2$ in E. From commutativity of



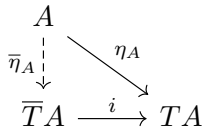
(a) Lemma 6.11

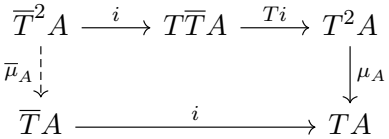
(b) $t_1 = t_2$ holds in $\mathcal{A}$



(c) naturality of i (d) i satisfies $t_1 = t_2$

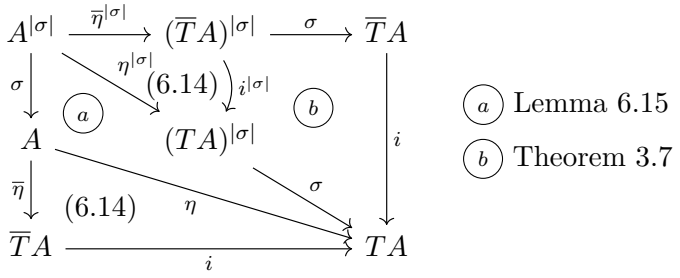
we find that $\eta_A\colon A \to TA$ and $\mu_A \circ Ti_{\mathcal{A}} \circ i_{\widehat{\overline{T}}\mathcal{A}}\colon \overline{T}^2A \to TA$ satisfy $t_1 = t_2$. Therefore, there are unique morphisms $\overline{\eta}_A\colon A \to \overline{T}A$ and $\overline{\mu}_A\colon \overline{T}^2A \to \overline{T}A$ making the diagrams below commute.



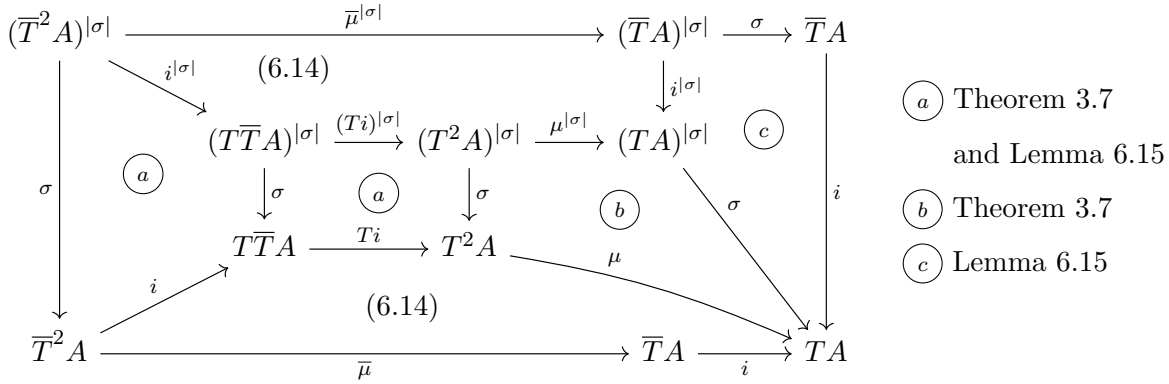


(6.14)

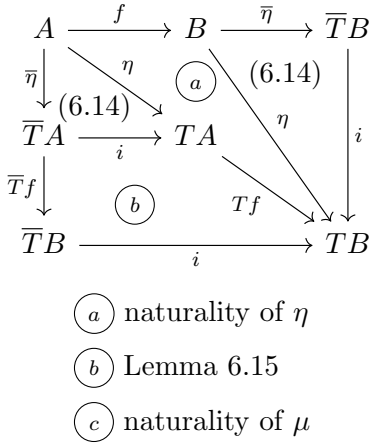
Recall from Lemma 6.14 that $i_{\mathcal{A}}$ is a mono. Therefore, commutativity of

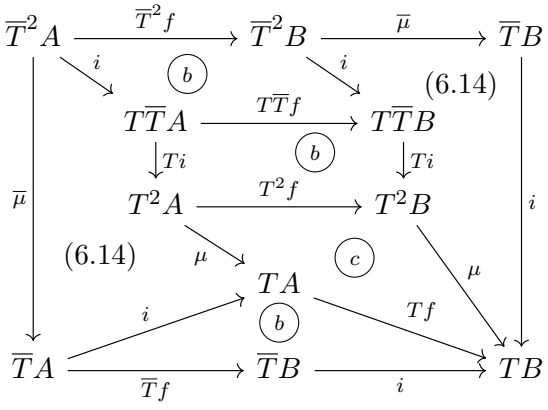


and



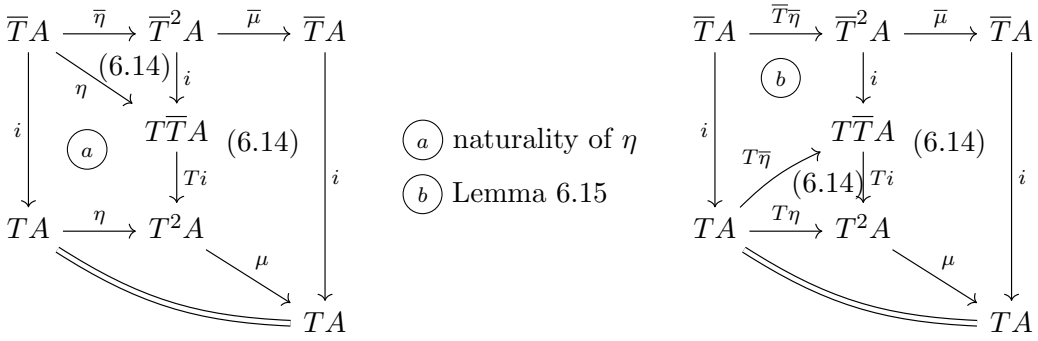
for every $\sigma \in \Sigma$ shows that $\overline{\eta}_{\mathcal{A}}$ and $\overline{\mu}_{\mathcal{A}}$ are Σ-algebra homomorphisms. Furthermore, commutativity of



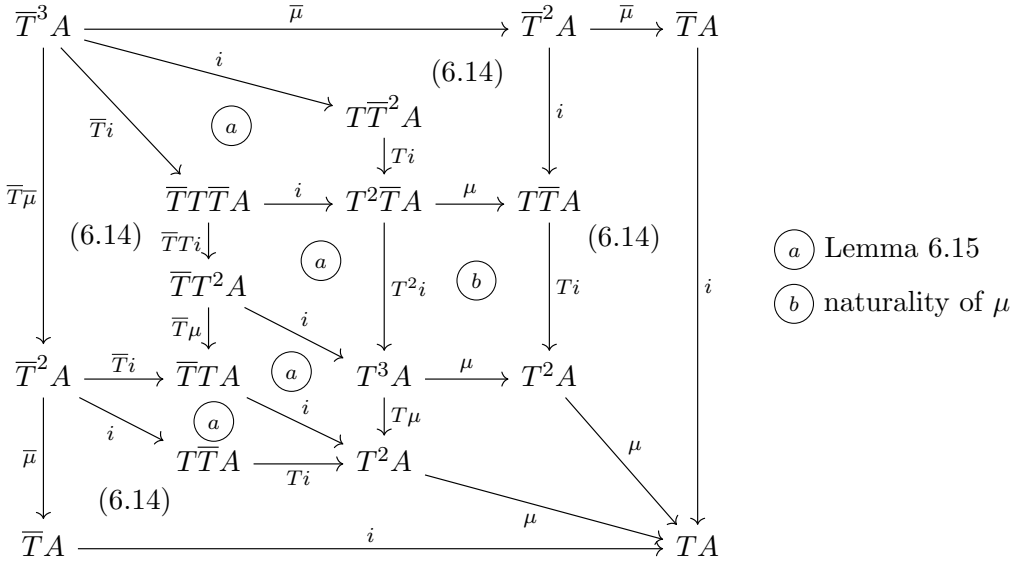


for all (Σ, E)-algebras $\mathcal{A}$ and $\mathcal{B}$ with respective carriers A and B and every Σ-algebra homomorphism $f\colon \mathcal{A} \to \mathcal{B}$ implies naturality of $\overline{\eta}$ and $\overline{\mu}$. As for the monad laws, observe

that the diagrams



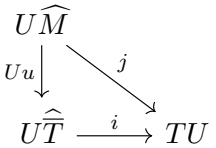
and



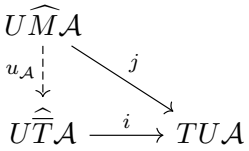
reduce them to the monad laws that hold for T. The diagrams in (6.14) state precisely that i is an oplax morphism of monads. □

Finally, we show the uniqueness of our construction with regard to the original monad T.

Theorem 6.18. *If $(\widehat{M}, \tilde{\eta}, \tilde{\mu})$ is any monad on $\mathbf{Alg}(\Sigma)$ with an oplax morphism of monads $j\colon U\widehat{M} \to TU$, then there is a unique monad morphism $u\colon \widehat{M} \to \widehat{\overline{T}}$ making the diagram below commute.*



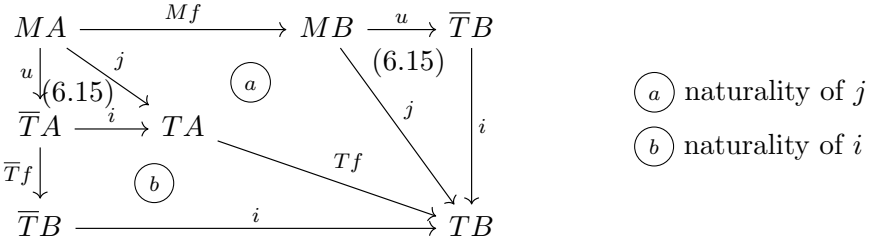
Proof. For any (Σ, E)-algebra $\mathcal{A}$, $\widehat{M}\mathcal{A}$ is a (Σ, E)-algebra and therefore satisfies all equations in E. From Lemma 6.11 it follows that $j_{\mathcal{A}}$ satisfies all equations that hold in $\widehat{M}\mathcal{A}$, which include those in E. Therefore, there is a unique morphism $u_{\mathcal{A}} \colon U\widehat{M}\mathcal{A} \to U\widehat{\overline{T}}\mathcal{A}$ making the diagram below commute.



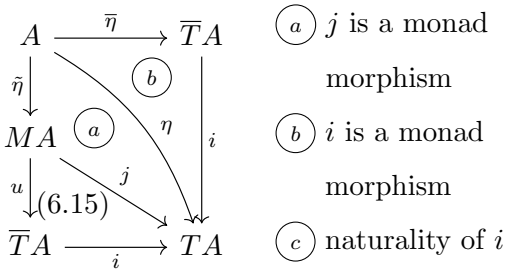
(6.15)

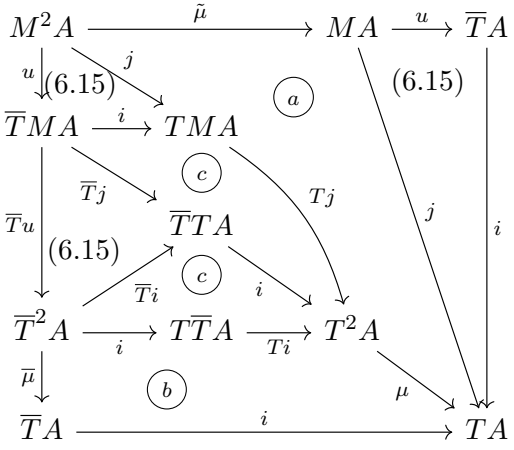
Since a morphism between (Σ, E)-algebras is just a Σ-algebra homomorphism, we can say that $u_{\mathcal{A}}$ is a morphism $\widehat{M}\mathcal{A} \to \widehat{\overline{T}}\mathcal{A}$ and accordingly could have written $Uu_{\mathcal{A}}$ in the above diagram.

Let A be the carrier of $\mathcal{A}$. We write MA for the carrier of $\widehat{M}\mathcal{A}$. To see that u is natural, consider a (Σ, E)-algebra $\mathcal{B}$ with carrier B and a Σ-algebra homomorphism $f \colon \mathcal{A} \to \mathcal{B}$. The commutative diagram



is a naturality square for u composed with the mono (Lemma 6.14) i. The commutative diagrams below further use the fact that i is monic to show that u is a monad morphism.





Uniqueness is pointwise and immediate. □

Let us briefly summarise the contributions of this section: first we define, for an algebra $\mathcal{A}$ and an equation $t_1 = t_2$, an object $\overline{T}A$ that is said to satisfy the equation. $\overline{T}A$ admits an injection into TA and a universal property. We then proceed to show that finding such an object for any $\mathcal{A}$ actually amounts to defining a functor $\widehat{\overline{T}}$ on $\mathbf{Alg}(\Sigma, E)$. Finally, we prove that $\widehat{\overline{T}}$ is actually a monad on $\mathbf{Alg}(\Sigma, E)$. In terms of monads S and T, our construction ca be described in the following way: if T does not preserve all the equations of E, in other words admits no lifting to $\mathbf{Alg}(\Sigma, E)$, we can now define $\widehat{\overline{T}}$ as the closest candidate to such a lifting.

6.3.3 Examples

We now present a few examples of this construction, applying our method to well-known cases of 'incompatible' monads.

Composing powerset and distribution $\mathcal{PD}$

We broadly described the example of $\mathcal{P}$ and $\mathcal{D}$ at the beginning of this section, let us now give a more formal explanation. The obstacle to monad composition was the family of equations expressing *idempotency* $p \oplus_\lambda p = p$, which is not preserved by $\mathcal{P}$. To apply the current method, we need now to find an object $\overline{\mathcal{P}}A$ and a morphism $i \colon \overline{\mathcal{P}}A \to \mathcal{P}A$ preserving all equations $p = p \oplus_\lambda p$ for $\lambda \in [0, 1]$.

Lemma 6.19. *Let $i \colon J \to \mathcal{P}A$. i satisfies $p = p \oplus_\lambda p$ for all $\lambda \in [0, 1]$ if and only if $\forall x \in J$, $i(x)$ is a* convex set.

Proof. Let $\lambda \in [0, 1]$, $x \in J, U = i(x)$.

$$\begin{aligned}
(\!(p)\!)_{\mathcal{PA}} \circ i)(x) &= U \\
(\!(p \oplus_\lambda p)\!)_{\mathcal{PA}} \circ i)(x) &= \mathcal{P}(\oplus_\lambda) \circ \psi \circ (\langle \mathrm{id}, \mathrm{id} \rangle)(U) \\
&= \mathcal{P}(\oplus_\lambda)(U \times U) \\
&= \{x \oplus_\lambda y \mid x, y \in U\}
\end{aligned}$$

Therefore i preserves the desired terms if and only if $\forall x \in J$, $\forall \lambda \in [0, 1]$, $U = \{x \oplus_\lambda y \mid x, y \in U\}$, which is equivalent to U being a convex set. □

In other words, i satisfies the desired preservation conditions if and only if it maps J to the *convex sets* of $\mathcal{P}(A)$. We define $\overline{\mathcal{P}}$ as the *convex powerset*: $\overline{\mathcal{P}}(A) = \mathcal{P}_C(A) = \{U \subseteq A \mid \forall x, y \in U, \lambda \in [0,1], x \oplus_\lambda y \in U\}$, and i as the inclusion in $\mathcal{P}$.

Lemma 6.20. *$i \colon \overline{\mathcal{P}} \to \mathcal{P}$ satisfies the commutativity of diagram* (6.9)*.*

Proof. Let J be an object such that $j \colon J \to \mathcal{P}(A)$ preserves the desired equations. Then by Lemma 6.19, for all $y \in J$, $j(y)$ is a convex set, in other words belongs to $\overline{\mathcal{P}}(A)$. Then $u \colon U \to \overline{\mathcal{P}}(A)$ defined by $u(y) = j(y)$ is the only morphism making the diagram (6.9) commute. □

Therefore by theorem 6.17, $\widehat{\overline{\mathcal{P}}}$ forms a monad on $\mathbf{Alg}(\Sigma, E)$: the convex powerset appears as a natural choice when trying to find a replacement for $\mathcal{P}$ on the category of convex algebras. Note that this is precisely the choice made by Silva et al. in [3]. They aim to define a determinisation procedure for probabilistic automata, but the lack of a distributive law $\mathcal{DP} \to \mathcal{PD}$ complicates the matter. They finally tweak their construction to make use of the convex powerset instead, more compatible with $\mathcal{D}$. Our method confirms that their choice is meaningful on a categorical level.

Iterated Finite Powerset $\mathcal{P}_f\mathcal{P}_f$

Composing the powerset monad with itself does not yield a monad, as shown by Klin and Salamanca in [21]. Similarly with the case of $\mathcal{PD}$, an obstacle lies in the non-preservation of idempotence by $\mathcal{P}_f$. We can now apply our method to 'fix' this composition: let $\mathcal{A}$ be a bounded join-semilattice, in other words an algebra for $\mathcal{P}_f$, and A be its carrier set. Let us now look for an object $\overline{\mathcal{P}_f}A$ and a morphism $i \colon \overline{\mathcal{P}_f}A \to \mathcal{P}_f A$ preserving the equation $p = p + p$.

Lemma 6.21. *Let $i \colon J \to \mathcal{P}_f A$. i satisfies $p = p + p$ if and only if $\forall x \in J$, $i(x)$ is a subset of A closed under $+$.*

Proof. Let $x \in J, U = i(x)$.

$$
\begin{aligned}
(\!|p|\!)_{\mathcal{P}_f\mathcal{A}} \circ i)(x) &= U \\
(\!|p + p|\!)_{\mathcal{P}_f\mathcal{A}} \circ i)(x) &= \mathcal{P}_f(+) \circ \psi \circ (\langle \mathrm{id}, \mathrm{id} \rangle)(U) \\
&= \mathcal{P}_f(+)(U \times U) \\
&= \{x + y, (x, y) \in U\}
\end{aligned}
$$

Hence i preserves $p = p + p$ if and only if $\forall x, i(x) = \{x + y, (x, y) \in i(x)\}$.

For all $x \in i(x), x + x = x$, hence it is always true that $i(x) \subseteq \{x + y, (x, y) \in i(x)\}$. The converse inclusion is true if and only if $i(x)$ is closed under $+$. □

In the same spirit as the above examples, we define $\widehat{\overline{\mathcal{P}_f}}$ as the set of join-closed subsets of A. What we obtain here is the *finite join-closed powerset monad* $\mathcal{P}_J$: **JSL** $\to$ **JSL** defined on **JSL** by $\mathcal{P}_J(X) = \{U \mid U \subseteq X, U \text{ finite}, U \text{ closed under } \cup_X\}$ where $\cup_X$ is the binary join operation of X, $\mathcal{P}_J f \colon \mathcal{P}_J X \to \mathcal{P}_J Y, U \mapsto f[U]$, $\eta_X(x) = \{x\}$ and $\mu_X(S)$ defined by our method. We define i as the inclusion of $\mathcal{P}_J$ in $\mathcal{P}_f$.

Lemma 6.22. *$i \colon \overline{\mathcal{P}_f} \to \mathcal{P}_f$ satisfies the commutativity of diagram*(6.9).

Proof. Let J be an object such that $j \colon J \to \mathcal{P}_f(A)$ preserves the desired terms, then by Lemma 6.21, for all $y \in J$, $j(y)$ is an ideal of A. Then $u \colon U \to \overline{\mathcal{P}_f}(A)$ defined by $u(y) = j(y)$ is the only morphism making the diagram (6.9) commute. □

Therefore by theorem 6.17, $\widehat{\overline{\mathcal{P}_f}}$ forms a monad on $\mathbf{Alg}(\Sigma, E) = \mathbf{JSL}$. This time it is the join-closed powerset monad, which we will denote by $\mathcal{P}_J$, that replaces $\mathcal{P}_f$ on the category of bounded join-semilattices.

6.3.4 Application: Generalised Determinisation of Alternating Automata

Our method highlights the fact that, though $\mathcal{P}_f$ admits no lifting to **JSL**, $\mathcal{P}_J$ is a the closest candidate to such a lifting. The same consideration applies to $\mathcal{P}_C$ and $\mathcal{D}$. In [3], the authors apply this remark to the case of non-deterministic probabilistic automata. Recall that these transition systems associate $\mathcal{P}_f$ and $\mathcal{D}$, and the classical procedure to determinise them fails because $\mathcal{P}_f$ cannot be lifted to $\mathcal{EM}(\mathcal{D})$. However, using $\mathcal{P}_C$ gives an alternate method to solve this problem. We show in this section how the same property of $\mathcal{P}_J$ and $\mathcal{P}_f$ can be applied to the determinisation of another type of automata.

Definitions

Alternating automata are detailed in [7] and [39]. We give here a categorical version of the definition. Recall that **JSL** $= \mathcal{EM}(\mathcal{P}_f)$ is the category of bounded join-semilattices.

1. An *Alternating Automaton* with respect to the alphabet A is a coalgebra for the functor $2 \times (\mathcal{P}_f \mathcal{P}_f(-))^A$, that is an object S together with a morphism $S \to 2 \times (\mathcal{P}_f \mathcal{P}_f S)^A$.

2. Let $\mathcal{U}, \mathcal{F}$ be respectively the *forgetul functor* **JSL** $\to$ **Set** and the functor building the *free join-semilattice* $U \mapsto (\mathcal{P}_f(U), \mu_U)$. $\mathcal{F}$ is left adjoint to $\mathcal{U}$ and this adjunction yields $\mathcal{P}_f = \mathcal{U} \circ \mathcal{F}$.

Generalised Determinisation of Alternating Automata

In this section we show how to operate a form of determinisation on alternating automata by using our previous results. First, let us prove a few categorical properties. Throughout this section, A is a non-empty set.

Lemma 6.23. *For an alphabet A, $2 \times (-)^A$ is a functor on* **JSL**.

Proof. For this proof, we can reuse the results from Chapter 5 to show that $2 \times (-)^A$ preserves all equations of **JSL**. First, recall that for all set A, the reader monad $\mathcal{R}_A$ is Cartesian, thus preserves all equations (see Example 4.37). In particular, its functorial part $(-)^A$ preserves all equations from **JSL**.

Similarly, $B \times (-)$ corresponds to the functorial part of the writer monad $\mathcal{W}_B$. We have shown that if B has a structure of bounded join-semilattice, then $\mathcal{W}_B$ is a relevant monad (see Example 4.32). The set 2 provided with the usual binary operation (for instance constituting $\mathbb{Z}_2$) is a bounded join-semilattice, therefore $\mathcal{W}_B$ is relevant and preserves all equations of **JSL**. Hence $B \times (-)$ is a functor of **JSL**.

We can immediately conclude that the composition of functors $2 \times (\mathcal{P}_J(-))^A$ is a functor on **JSL**. □

We note F the composition of functors $2 \times (\mathcal{P}_J(-))^A$ on **JSL**. We now show a series of transformations, by analogy with [3], that gradually turn an alternating automaton into its deterministic version.

Lemma 6.24. *There exists an injective natural transformation $i \colon \mathcal{U} \circ \mathcal{P}_J \to \mathcal{P}_f \circ \mathcal{U}$.*

Proof. It suffices to take the injection $i(U) = U$ mapping each semilattice to its set. □

We define C as $\mathcal{U} \circ \mathcal{P}_J \circ \mathcal{F}$. This functor maps a set X to the set of $\cup_X$-closed subsets of subsets of X. We can now define a notion of *quasi-lax lifting*:

Definition 6.25 (Quasi-Lax Lifting)**.** In [3], the authors define a functor $H \colon \mathcal{EM}(M) \to \mathcal{EM}(M)$ as a (L_1, L_2) *quasi-lax lifting* if both:

- There exists a natural transformation $\mathcal{U} \circ H \to L_2 \circ \mathcal{U}$
- The following diagram commutes:

$$\begin{array}{ccc} \mathcal{EM}(M) & \xrightarrow{H} & \mathcal{EM}(M) \\ {\scriptstyle\mathcal{F}}\uparrow & & \downarrow{\scriptstyle\mathcal{U}} \\ \mathbf{Set} & \xrightarrow{L_1} & \mathbf{Set} \end{array} \tag{6.16}$$

This concept is used as a replacement for the usual notion of lifting when the latter cannot be found. For instance in our situation, there exists no lifting of $\mathcal{P}_f$ to **JSL**. But it follows from the previous lemmas and from the definition of C that $\mathcal{P}_J$ is a $(C, \mathcal{P}_f)$ quasi-lax lifting, and $2 \times (\mathcal{P}_J)^A$ is a $(2 \times (C)^A, 2 \times (\mathcal{P}_f)^A)$ quasi-lax lifting.

$2 \times \mathcal{P}_f{\mathcal{P}_f}^A$ coalgebras to $2 \times C^A$ coalgebras

Let $u \colon \mathcal{P}_f\mathcal{P}_f \to C$ be the natural transformation realising the closure under $\cup$.

$$\begin{aligned} u_X \colon \mathcal{P}_f\mathcal{P}_f X &\to \mathcal{U}\mathcal{P}_J\mathcal{F}X \\ U &\mapsto \{A_1 \cup A_2 \cdots \cup A_n \mid n \geq 1, A_i \in U\} \end{aligned}$$

$2 \times u^A \colon 2 \times (\mathcal{P}_f\mathcal{P}_f)^A \to 2 \times (C)^A$, defined pointwise, is also a natural transformation. It is the first step of our conversion, as it maps alternating automata to $2 \times C^A$ coalgebras. The next lemma show how transitions are preserved by closure under $\cup$.

Lemma 6.26. *Let $(\mathcal{S}, c)$ be a F coalgebra represented as a transition system, S the carrier of the semilattice $\mathcal{S}$, $x_1, x_2, y_1, y_2, z \in S$, $a \in A$.*

1. $x_1 \vee x_2 \overset{a}{\mapsto} z$ *iff* $z = y_1 \vee y_2$ *and* $x_1 \overset{a}{\mapsto} y_1$, $x_2 \overset{a}{\mapsto} y_2$
2. $x_1 \vee x_2 \overset{a}{\not\mapsto} y_1 \vee y_2$ *iff* $x_1 \overset{a}{\not\mapsto} y_1$ *or* $x_2 \overset{a}{\not\mapsto} y_2$

Proof. Since F is a functor on **JSL**, c is a join-semilattice homomorphism. Hence For $a \in A$, $c(x_1 \vee x_2)(a) = (c(x_1) \vee c(x_2))(a)$. By definition of $(-)^A$, this is equivalent to $c(x_1)(a) \vee c(x_2)(a)$. □

$2 \times C^A$ coalgebras to $2 \times {\mathcal{P}_J}^A$ coalgebras

Finally, recall that C and $\mathcal{P}_J$ are related by properties of quasi-lax lifting. We use this fact to complete the conversion as summarised by the following lemma:

Lemma 6.27. *There is a one-to-one correspondence between* $2 \times C^A$ *coalgebras on* **Set** *and* $2 \times \mathcal{P}_J^A$ *coalgebras on* **JSL** *with carriers free algebras:*

$$\frac{c\colon S \to 2 \times (CS)^A}{c^\#\colon \mathcal{F}S \to 2 \times (\mathcal{P}_J \mathcal{F}S)^A}$$

Proof. This result is a consequence the quasi-lax lifting properties, shown as Lemma 25 in [3]. □

Let us give more insight on the steps of our transformation. We progressively turn a $2 \times (\mathcal{P}_f \mathcal{P}_f)^A$ automaton into a $2 \times (\mathcal{P}_J)^A$ automaton on free algebras:

1. Start with $(S \xrightarrow{c_1} 2 \times (\mathcal{P}_f \mathcal{P}_f S)^A)$
2. Close under $\cup$ and obtain $(S \xrightarrow{c_2} 2 \times (CS)^A)$, defined as $(S \xrightarrow{c_1} 2 \times \mathcal{P}_f \mathcal{P}_f S^A \xrightarrow{2 \times u_S{}^A} 2 \times (CS)^A)$

   The next steps explain the correspondence described in lemma 6.27.
3. Apply $\mathcal{P}_f$ and obtain $(\mathcal{P}_f S \xrightarrow{\mathcal{P}_f c_2} \mathcal{P}_f(2 \times (CS)^A))$
4. $\mathcal{F}S = (\mu\colon \mathcal{P}_f \mathcal{P}_f S \to S)$ is an object of **JSL**. Let us apply $2 \times (\mathcal{P}_J)^A$ to it and obtain an algebra $(\alpha\colon \mathcal{P}_f(2 \times (CS)^A) \to 2 \times (CS)^A)$ (whose type is determined by being a $\mathcal{P}_f$-algebra, with carrier $2 \times (CS)^A$ by properties of the quasi-lax lifting)
5. Finally, compose α with $\mathcal{P}_f c_2$ to obtain $\mathcal{P}_f S \xrightarrow{\mathcal{P}_f c_2} \mathcal{P}_f(2 \times (CS)^A) \xrightarrow{\alpha} 2 \times (CS)^A$, which is an arrow $\mathcal{U}c^\#\colon \mathcal{U}\mathcal{F}S \to \mathcal{U}(2 \times (\mathcal{P}_J \mathcal{F}S)^A)$

We refer to [3] for more details on how this conversion can be used in a context of bisimulations.

6.4 Non-Monoidal Liftings

So far we have focused our attention on lifting the algebraic features of a monad S via the monoidal structure of another monad T. When the lifted version of the operations do not verify the equations presenting S, it seems to indicate that T and S do not compose. However, it is possible that another lifting allows all laws to hold. In this chapter we present cases where less conventional liftings succeed to build distributive laws when the monoidal option fails.

6.4.1 Lifting Associative and Commutative Operations

Let us first consider a monad S presented by a signature $\Sigma = \{*\}$ and the following equations:

$$x * (y * z) = (x * y) * z \tag{6.17}$$
$$x * y = y * z \tag{6.18}$$
$$x * (y * (y * z)) = x * (y * z) \tag{6.19}$$

We focus our attention on combining S with the powerset monad. In this section, we focus on the powerset monad $\mathcal{P}$ rather that its finite version $\mathcal{P}_f$. In our monoidal point of view, S does not distribute over $\mathcal{P}$ since the law 6.19 features a variable duplication (Theorem 3.1). Let us however present an alternative lifting of $*$ that preserves this equation. Note that we consider S as the monad generating free Σ-terms under equivalences defined by the previous equations. Because of 6.17 we will omit bracketing in those terms.

Theorem 6.28. *There exists a distributive law $S\mathcal{P} \to \mathcal{P}S$.*

Proof. Let us define the following lifting for $*$:

$$\hat{*}\colon \mathcal{P}SX \times \mathcal{P}SX \to \mathcal{P}SX$$
$$U, V \mapsto \{u_1 * \cdots * u_k * v_1 * \cdots * v_n \mid u_i \in U, v_i \in V, n, k \geq 1\}$$

Since we no longer rely on monoidality, we cannot assume that linear equations are automatically preserved. It is however straightforward to see that the commutativity of $*$ transfers to $\hat{*}$.

$$\begin{aligned} U\hat{*}V &= \{u_1 * \cdots * u_k * v_1 * \cdots * v_n \mid u_i \in U, v_i \in V, n, k \geq 1\} \\ &= \{v_1 * \cdots * v_n * u_1 * \cdots * u_k \mid u_i \in U, v_i \in V, n, k \geq 1\} \qquad \text{by commutativity} \\ &= V\hat{*}U \end{aligned}$$

For the case of associativity, note first that each term t of $U\hat{*}(V\hat{*}W)$ is formed of a subterm $u_1 * \cdots * u_n$ followed by a series of subterms of the shape $v_1 * \cdots * v_k * w_1 * \ldots w_n$ (for $u_i \in U, v_i \in V, w_i \in W$). By commutativity, these elements can be rearranged to obtain an term equivalent to t, this time of the shape $u_1 * \cdots * u_k * v_1 * \cdots * v_n * w_1 * \cdots * w_m$.

Therefore we have:

$$U\hat{*}(V\hat{*}W) = \{u_1 * \cdots * u_k * v_1 * \cdots * v_n * w_1 * \cdots * w_m \mid u_i \in U, v_i \in V, w_i \in W, n, k, m \geq 1\}$$
$$= (U\hat{*}V)\hat{*}W$$

Let us now prove that the equation 6.19 holds for subsets of SX, keeping in mind that $u_i \in U, v_i, v'_i \in V, w_i \in W, k, l, m, n \geq 1$.

$$U\hat{*}(V\hat{*}(V\hat{*}W)) = \{u_1 * \cdots * u_k * v_1 * \cdots * v_l * v'_1 * \cdots * v'_m * w_1 * \cdots * w_n\}$$
$$\subseteq U\hat{*}(V\hat{*}W)$$

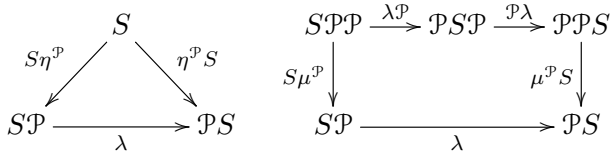
The converse inclusion relies on the following facts: first, any element of $U\hat{*}(V\hat{*}W)$ of the form $u_1 * \cdots * u_k * v_1 * \cdots * v_l * w_1 * \cdots * w_n$ with $l > 1$ is clearly in $U\hat{*}(V\hat{*}(V\hat{*}W))$ as well. Moreover, any other element x is of the form $u_1 * \cdots * u_k * v * w_1 * \cdots * w_n$, and by law 6.19 we have $x = u_1 * \cdots * u_k * v * v * w_1 * \cdots * w_n$ therefore $x \in U\hat{*}(V\hat{*}(V\hat{*}W))$, and the lifted version of the law holds.

We have shown that the algebraic structure of S is preserved by $\mathcal{P}$. It is not sufficient to prove that we have built a distributive law: we can lift $\mathcal{P}$ to $\mathcal{EM}(S)$, which amounts to building an $\mathcal{EM}$-law, but it remains to show that it is a Kl-law too (see section 2.3.3). Note that in the previous chapters we obtained this result automatically when relying on the monoidal structure.

Let us first give a more explicit definition of the distributive law associated with our lifting. We consider SX as the set of free terms generated by the signature and equations of S. It is easy to see that because of associativity of $*$, any $t \in SX$ can be written as $x_1 * x_2 * \cdots * x_n$ for $n \geq 1, x_i \in X$.

$$S\mathcal{P}X \to \mathcal{P}SX$$
$$U \in \mathcal{P}X \mapsto U$$
$$U_1 * \cdots * U_n \mapsto \{u_1^1 * \cdots * u_{k_1}^1 * u_1^2 * \cdots * u_{k_1}^2 * \cdots * u_1^n * \cdots * u_{k_1}^n \mid u_j^i \in U_i, n, k_i \geq 1\}$$

We now have to show that diagrams DL. 1 and DL. 3 commute:



(DL. 1) (DL. 3)

Let us treat the diagram on the left: let X be a set. Any $x \in X$ also belongs to SX, and for those elements the diagram trivially commutes. Any other S-term on X can be written $x_1 * \cdots * x_n$. For the sake of clarity, let us treat $a * b$, with $a, b \in X$. We have $\eta^{\mathcal{P}}_{SX}(a * b) = \{a * b\}$, and $\lambda_X \circ S\eta^{\mathcal{P}}_X(a * b) = \{a\}\hat{*}\{b\} = \{a * \cdots * a * b * \cdots * b\}$. By repeated applications of law 6.19, we have $a * \cdots * a * b * \cdots * b = a * b$, therefore both branches of the diagram agree and build the set $\{a * b\}$. The case of $x_1 * \cdots * x_n$ follows naturally.

Similarly, we show that DL. 3 commutes. Let X be a set, commutation is again immediate for a S-term U such that $U \in S\mathcal{P}\mathcal{P}X$. Let us now treat $U * V$, for $U = \{U_1, \ldots, U_n\}, V = \{V_1, \ldots, V_m\} \in \mathcal{P}\mathcal{P}X$. The left branch merges all subsets of U and of V then distributes, building $\{u_1 * \cdots * u_n * v_1 * \cdots * v_k\}$, where $u_i \in U_1 \cup \cdots \cup U_n, v_i \in V_1 \cup \cdots \cup V_m$. The right branch builds the following set:

$$\{u^1_1 * \cdots * u^1_{k_1} * \cdots * u^p_1 * \cdots * u^p_{k_p} * v^1_1 * \cdots * v^1_{l_1} * \cdots * v^q_1 * \cdots * v^q_{l_q}\}$$

Where $p, q, l_i, k_i \geq 1$, and where all elements $u^i_{\ldots}$ belong to some $U'_i \in U$ (similarly, all $v^i_{\ldots}$ belong to one of the subsets of V). It is easy to see that it corresponds to the set built by the previous branch. Therefore the diagram commutes, proving that we have built a distributive law. □

We have succeeded to distribute the monad S over $\mathcal{P}$, by making use of the infinite subsets contained in the full powerset. Note that our distributive law can be used for other monads: the way we preserve equation 6.19 extends to a whole class of equations.

Theorem 6.29. *Let S be a monad presented by a signature $\Sigma = \{*\}$ and the following equations:*

- *Associativity:* $x * (y * z) = (x * y) * z$

- *Commutativity:* $x * y = y * x$
- *Any laws* $u = v$ *where* u *and* v *both contain* $*$ *and only differ by duplications of variables*

Then S *distributes over* $\mathcal{P}$*.*

The condition on variable duplication has to be understood as: there exists a linear term t, there exist substitutions of the type $x \mapsto x * x$, such that u and v are both obtained from t after a series of such substitutions. This idea is reminiscent of our concept of $t[x]$-equations introduced in Chapter 5, but this time allowing several duplications.

Proof. We use the same distributive law as in Theorem 6.28 and show that $u = v$ holds for elements of $\mathcal{P}SX$, in other words that, for $V = Var(u) = Var(v)$, $[\![u]\!]^V_{PSX} = [\![v]\!]^V_{PSX}$, naturally interpreting terms according to our lifting of the signature. Assume that $|V| = n$, let $U_1, \ldots, U_n \in \mathcal{P}SX$. Then by definition of our lifting, $[\![t]\!]^V_{PSX}(U_1, \ldots, U_n) = [\![u]\!]^V_{PSX}(U_1, \ldots, U_n) = [\![v]\!]^V_{PSX}(U_1, \ldots, U_n)$, as any $U \hat{*} V$ already contains terms with all possible variable duplications. Therefore any equation of the desired type is preserved by our lifting. Showing that this defines a valid distributive law is done in the same manner as in the proof of Theorem 6.28. □

The previous constructions seem to necessitate infinite subsets and therefore the use of the full powerset. However, for the same type of lifting, the corresponding subsets may sometimes fit in the finite powerset. Let us first consider a monad S presented by a signature $\Sigma = \{*\}$ and the following set E of equations:

$$x * (y * z) = (x * y) * z \tag{6.20}$$
$$x * y = y * x \tag{6.21}$$
$$x * (x * y) = x * y \tag{6.22}$$

Consider the Finite Powerset monad $\mathcal{P}_f$. We show the existence of a distributive law $S\mathcal{P}_f \to \mathcal{P}_f S$ by showing that $\mathcal{P}_f$ is a monad on $\mathbf{Alg}(\Sigma, E)$. Let us define the following lifting for $*$:

$$\begin{aligned} \hat{*}\colon \mathcal{P}_f SX \times \mathcal{P}_f SX &\to \mathcal{P}_f SX \\ U, V &\mapsto \{u_1 * \cdots * u_k * v_1 * \cdots * v_n \mid u_i \in U, v_i \in V, n, k \geq 1\} \end{aligned}$$

This lifting is well-defined as, for U, V finite, $\{u_1 * \cdots * u_k * v_1 * \cdots * v_n \mid u_i \in U, v_i \in V, n, k \geq 1\}$ is finite: terms $u_1 * \cdots * u_k * v_1 * \cdots * v_n$ can be seen as words whose letters are

distinct unless they only have two letters, and whose order does not matter. Therefore $\mathcal{P}_f$ is a functor on Σ-algebras. It is easy to see that this lifting preserves equations in E, hence $\mathcal{P}_f$ is a functor on (Σ, E)-algebras. We show now that $\mathcal{P}_f$ is a monad on (Σ, E)-algebras:

- η is a (Σ, E)-algebra morphism as $\{a\}\hat{*}\{b\} = \{a * b\}$.
- μ is a (Σ, E)-algebra morphism. Let $(S, T) = (\{S_1, \dots S_n\}, \{T_1, \dots T_m\}) \in \mathcal{P}_f\mathcal{P}_f A \times \mathcal{P}_f\mathcal{P}_f A$. $\hat{*} \circ (\mu \times \mu)(S, T) = \{a_1 * \cdots * a_k * b_1 * \cdots * b_l \mid a_i \in \text{ some } S_j, b_i \in \text{ some } T_j\} = \mu \circ \hat{*}(S, T)$

6.4.2 Lifting Non-associative Operations

So far our work has been greatly facilitated by the associativity of the considered law. Without this law, similar liftings are still possible (we could for instance gather all possible ways to place brackets and replicate the previous approach) but their general form may not be easily readable. When presented with this situation, it may be more sensible to define an ad-hoc lifting depending on the equations we want to preserve. We give here an example of such a case, due to Bartek Klin and Julian Salamanca.

Let S be the monad presented by the signature $\Sigma = \{*\}$ and the following equation:

$$x * (x * y) = x * y$$

Theorem 6.30. *There exists a distributive law of S over $\mathcal{P}$*

Proof. In a similar manner as the previous section, we define a special lifting of $*$:

$$\hat{*} \colon \mathcal{P}SX \times \mathcal{P}SX \to \mathcal{P}SX$$
$$U, V \mapsto \{u_1 * (u_2 * (\cdots * (u_k * v)) \dots) \mid u_i \in U, v \in V, k \geq 1\}$$

It becomes clear that once again this lifting preserves the desired equation. We have $U\hat{*}(U\hat{*}V) = \{u_1 * (u_2 * (\cdots * (u_k * v)) \dots) \mid u_i \in U, v \in V, k \geq 2\}$, and moreover $u * (u * v) = u * v$ hence $\{u_1 * (\cdots * (u_k * v) \dots) \mid k \geq 2\} = \{u_1 * (\cdots * (u_k * v) \dots) \mid k \geq 1\} = U\hat{*}V$. Showing that this defines a valid distributive law is achieved the same way as previously. □

Our goal in this section was to show that some dup equations, although seeming incompatible with the powerset monad, can be preserved by a clever lifting if we allow infinite subsets. Note that our results do not cover laws of idempotency ($x * x = x$), as one side of the equation does not contain the operation $*$. Such a law is still very likely to prevent composition with $\mathcal{P}$.

6.5 Discussion and Related Work

The previous chapters explored a series of conditions that allow for monad composition via preservation of algebraic features. In the current chapter, we have explored situations where our conditions are not verified, meaning that the method of Chapter 3 cannot be applied. What are our options when trying to compose two monads T and S that present incompatible features? Our contribution in this chapter is a series of strategies to circumvent such obstacles by operating tailor-made modifications on our monads.

The first method amounts to removing from S the problematic feature: an equation that T cannot preserve. This yields a modified monad S' with which T can be composed, but the cost is losing a potentially important feature of S. For a strategy that does not affect S, recall from Chapter 5 that the categorical properties of affineness and relevance are sufficient to preserve diverse classes of equations. When combined in the case of Cartesian monads, all equations are guaranteed to be preserved. If T fails to preserve part of the presentation of S, we can restrict it to its affine, relevant or Cartesian part. This construction results in a new monad T' which automatically composes with S. Again, the drawback of this method is that it results in potentially extreme modifications of the original monad: for instance, enforcing relevance of the powerset monad $\mathcal{P}$ requires to restrict it to the maybe monad $\bot$, a much less expressive construct. The idea of this strategy can be refined to obtain our third method.

When trying to compose T and S, we consider the free algebra SX and recall that there exists a lifting $\widehat{T}$ of T to the category of S-algebra (Theorem 3.7). If some equations presenting it cannot be preserved by T, we restrict $\widehat{T}$ to a functor on S-algebras which, in the manner of an equaliser, preserves the desired equations. The idea behind this strategy is based on the work of Bonchi, Silva and Sokolova in [3]. The authors are faced with the task of composing the monads $\mathcal{P}$ and $\mathcal{D}$, but it is well-known since the works of Varacca in [37] and Varacca and Winskel in [38] that $\mathcal{P}$ does not preserve the structure of a convex algebra $\mathcal{A}$. To overcome this incompatibility, the authors of [3] tweak $\mathcal{P}$ to obtain the *convex powerset*. This monad is defined on the category of convex algebras and preserves the equations of idempotence incompatible with $\mathcal{P}$. Bonchi, Sokolova and Vignudelli bring this reasoning further with the study undertook in [4] and show that this construction actually results in the monad $\mathcal{C}$ of *convex subsets of distributions*. This monad, previously studied by Jacobs in [16], is shwon in [4] to be presented by the theory of *convex semilattices*.

Our method takes inspiration from [3] and generalises the construction undertaken in that paper to any monoidal monad and any non-preserved equation. A future direction for our work could be to generalise the results of [4] together with our findings from Section 6.3. In some cases where distributive laws cannot be constructed, one could show that building a composite monad can still be achieved not only in the case of $\mathcal{P}$, but for any monoidal monad.

Finally, we presented a series of cases where a distributive law can be found between a monad presented by a non-linear theory and the powerset monad. These ad-hoc

constructions are mostly inspired by an idea of Bartek Klin and taken a step further in this chapter. These ideas are however not yet generalised: our reasoning relies heavily on defining a convenient lifting of only one algebraic operation, and does not extend yet to a more complex signature.

Chapter 7

An Algebraic Perspective

The previous chapters have explored distributive laws in a purely categorical framework: in Chapter 3, we define a method which allows to compose monads T and S, then discuss conditions for its success in Chapter 5, expressed as properties of T and of a theory $\mathbb{S}$ presenting S. However, one aspect that we have not studied yet is the algebraic presentation $\mathbb{T}$ of T. Let X be a set. As we did with the monad S, we can consider the set TX of free $\mathbb{T}$-terms generated by X (the carrier of the free T-algebra). We can then examine how categorical properties translate on algebraic terms, and study the effect of our distributive laws $ST \to TS$ on objects of STX considered as algebraic terms. Throughout this chapter, in order to focus on the algebraic presentation of T we assume that it is a monoidal finitary monad (presented by a theory $\mathbb{T}$ of finite arity). For instance, we do not consider the powerset monad $\mathcal{P}$, because its presentation includes an infinitary union operation; instead we will consider its finite counterpart $\mathcal{P}_f$.

First, we study the action of the monoidal map ψ on algebraic terms. Then we will prove that the monadic properties of commutativity, affineness and relevance have algebraic formulations as well as intuitive consequences. We will examine the shape of distributive laws resulting from our method with an algebraic approach, and finally we apply those techniques to a new subclass of equations to show that their preservation implies relevance.

7.1 Monoidal Maps for Terms

First we focus on the central feature of monoidal monads: the map $\psi_{X,Y} \colon TX \times TY \to T(X \times Y)$. Before studying it under a new light, let us first recall a few notions of algebra and define some notation. We recall the definition of algebraic terms seen in Chapter 2.

Definition 7.1. For X a set, the set of $\Sigma-$*terms* over X (or 'generated by X') is defined

as follows:

- For $x \in X$, x is a term which we denote by (x) for more clarity.
- If $t_1, \dots t_n$ are terms and σ is a symbol of Σ with arity n, then $\sigma(t_1, \dots t_n)$ is a term.

For $\mathbb{T} = (\Sigma, E)$ a theory, the set of $\mathbb{T}$-terms over X is the set of equivalence classes of Σ-terms (that we will denote by a representative) under the equations in E.

For a monad T presented by the theory $\mathbb{T} = (\Sigma, E)$ and a set X, TX can be seen as the set of $\mathbb{T}$-terms generated by X. For a morphism for $f\colon X \to Y$, Tf maps a term $M \in TX$ to $M[f(x)/x]$; η_X maps x to the term (x) and μ_X maps a term over terms to the collapsed term. In this section, we consider elements of $TX \times TY$ as pairs of terms and we study the effect of $\psi_{X,Y}$ on them by using properties of monoidal monads. First, we focus on the simplest terms: the generators $(x), (y)$.

Lemma 7.2. *for $x, y \in X$, $\psi((x), (y)) = (x, y)$.*

Proof. This follows from the axiom MM.1. as we have $\psi((x), (y)) = \psi(\eta_X(x), \eta_X(y)) = \eta_{X\times X}(x, y) = (x, y)$. □

We can now move on to a slightly more difficult case with a term of the form $\sigma(t_1, \dots t_n)$.

Lemma 7.3. *for $a \in Y$, $\sigma \in \Sigma$, $x_1, \dots, x_n \in X$ we have*

$$\psi_{X,Y}(\sigma(x_1, \dots, x_n), (a)) = \sigma((x_1, a), \dots, (x_n, a)) \tag{7.1}$$

Proof. First, consider $\psi_{X,1}\colon TX \times T1 \to T(X \times 1)$. We denote by $*$ the unique element of the set 1. by axiom MF.2 we have the following:

$$\begin{aligned}\psi_{X,1} \circ (\mathrm{id} \times \eta_1)(\sigma(x_1, \dots, x_n), *) &= (T\rho^{-1} \circ \rho)(\sigma(x_1, \dots, x_n), *) \\ &= T\rho^{-1}(\sigma(x_1, \dots, x_n)) \\ &= \sigma((x_1, *), \dots, (x_n, *))\end{aligned}$$

For $a \in X$, we also denote by a the morphism $1 \to X$ selecting the element a. Note that for every $t \in TX$ we have $\psi_{X,Y}(t,(a)) = \psi_{X,Y} \circ (\mathrm{id} \times \eta_1) \circ (\mathrm{id} \times a)(t,*)$.

$$\begin{aligned}\psi_{X,Y} \circ (\mathrm{id} \times \eta_1) \circ (\mathrm{id} \times a) &= \psi_{X,Y} \circ (\mathrm{id} \times Ta) \circ (\mathrm{id} \times \eta_1) && \text{naturality}\\ &= T(\mathrm{id} \times a) \circ \psi_{X,1} \circ (\mathrm{id} \times \eta_1) && \text{naturality}\end{aligned}$$

Therefore we have:

$$\begin{aligned}\psi_{X,Y}(\sigma(x_1,\ldots,x_n),(a)) &= \psi_{X,Y} \circ (\mathrm{id} \times \eta_1) \circ (\mathrm{id} \times a)(\sigma(x_1,\ldots,x_n),*)\\ &= T(\mathrm{id} \times a) \circ \psi_{X,1} \circ (\mathrm{id} \times \eta_1)(\sigma(x_1,\ldots,x_n),*)\\ &= T(\mathrm{id} \times a) \circ \psi_{X,1}(\sigma(x_1,\ldots,x_n),(*))\\ &= T(\mathrm{id} \times a)\sigma((x_1,*),\ldots,(x_n,*))\\ &= \sigma((x_1,a),\ldots,(x_n,a))\end{aligned}$$

□

We can now establish a more general inductive formulation of the action of $\psi_{X,Y}$ on algebraic terms.

Lemma 7.4. *for $v \in TY$, $f \in \Sigma$, $t_1,\ldots,t_n \in TX$ we have*

$$\psi_{X,Y}(\sigma(t_1,\ldots,t_n),v) = \sigma(\psi_{X,Y}(t_1,v),\ldots,\psi_{X,Y}(t_n,v)) \tag{7.2}$$

This result relies on the axiom MM.2. First, note that $\sigma((t_1),\ldots,(t_n))$ and (v) are respectively terms of TTX and TTY. Then we have

$$\begin{aligned}\psi_{X,Y}(\sigma(t_1,\ldots,t_n),v) &= \psi_{X,Y} \circ (\mu_X \times \mu_Y)(\sigma((t_1),\ldots,(t_n)),(v)) && \text{by } MM.2\\ &= \mu_{X\times Y} \circ T\psi_{X,Y} \circ \psi_{TX,TY}(\sigma((t_1),\ldots,(t_n)),(v))\\ &= \mu_{X\times Y} \circ T\psi_{X,Y}(\sigma((t_1,v),\ldots,(t_n,v))) && \text{by Lemma 7.3}\\ &= \mu_{X\times Y}(\sigma((\psi_{X,Y}(t_1,v)),\ldots,(\psi_{X,Y}(t_n,v))))\\ &= \sigma(\psi_{X,Y}(t_1,v),\ldots,\psi_{X,Y}(t_n,v))\end{aligned}$$

So far we have only examined the binary monoidal map ψ, but this work is immediately extended to the m-ary case. Let $X_1,\ldots,X_m$ be sets and let $v_2 \in TX_2,\ldots,v_m \in TX_m$ By definition of $\psi^{(m)}$ for $m > 2$, we immediately obtain

$$\psi^{(m)}_{X_1,\ldots,X_m}(\sigma(t_1,\ldots,t_n),v_2,\ldots,v_m) = \sigma(\psi^{(m)}_{X_1,\ldots,X_m}(t_1,v_2,\ldots v_m),\ldots,\psi^{(m)}_{X_1,\ldots,X_m}(t_n,v_2,\ldots,v_m)) \tag{7.3}$$

Our formulation for the case of $\psi^{(m)}$ is already very general, but only operates inductively on the first term in the tuple. Luckily, we have shown in Chapter 5 that monoidal maps behave well when their inputs are permuted. This allows to establish an inductive formulation for the general case of $\psi^{(m)}$. Note that the cases of $m = 0, 1$ are not treated because of their triviality: $\psi^{(0)} = \eta_1$ which maps $* \in 1$ to $(*) \in T1$, and $\psi^{(1)} = \mathrm{id}$.

Lemma 7.5. *Let $m > 2$, $i \leq m$.*

$$\begin{aligned}&\psi^{(m)}_{X_1,\dots,X_m}(v_1,\dots,v_{i-1},\sigma(t_1,\dots,t_n),v_{i+1},\dots,v_m)\\ &\quad= \sigma(\psi^{(m)}_{X_1,\dots,X_m}(v_1,\dots,v_{i-1},t_1,v_{i+1},\dots,v_m),\dots,\psi^{(m)}_{X_1,\dots,X_m}(v_1,\dots,v_{i-1},t_n,v_{i+1},\dots,v_m))\end{aligned}$$

Proof. Let α be the permutation swapping the first and the i-th element of a m-tuple. We have

$$\begin{aligned}&\psi^{(m)}_{X_1,\dots,X_m}(v_1,\dots,v_{i-1},\sigma(t_1,\dots,t_n),v_{i+1},\dots,v_m)\\ &\quad= \psi^{(m)}_{X_1,\dots,X_m}\circ\alpha(\sigma(t_1,\dots,t_n),\dots,v_{i-1},v_1,v_{i+1},\dots,v_m)\\ &\quad= T\alpha\circ\psi^{(m)}_{X_1,\dots,X_m}(\sigma(t_1,\dots,t_n),\dots,v_{i-1},v_1,v_{i+1},\dots,v_m) \qquad \text{by Lemma 4.23}\\ &\quad= T\alpha(\sigma(\psi^{(m)}_{X_1,\dots,X_m}(t_1,v_2,\dots,v_{i-1},v_1,v_{i+1},\dots v_m),\dots,\\ &\qquad\qquad \psi^{(m)}_{X_1,\dots,X_m}(t_n,v_2,\dots,v_{i-1},v_1,v_{i+1},\dots,v_m))) \qquad \text{by (7.3)}\\ &\quad= \sigma(\psi^{(m)}_{X_1,\dots,X_m}(v_1,v_2,\dots,v_{i-1},t_1,v_{i+1},\dots v_m),\dots,\\ &\qquad\qquad \psi^{(m)}_{X_1,\dots,X_m}(v_1,v_2,\dots,v_{i-1},t_n,v_{i+1},\dots,v_m))\end{aligned}$$

□

The previous formula is exhaustive, but not very economical to write; we now present a more convenient notation. For t a term of TX, we use denote by $t[f(x)/x]$ the term which we obtain when substituting each variable (represented as x) in t with $f(x)$. For instance for $t = y + (z * y)$, we have $t[(x, a)/x] = (y, a) + ((z, a) * (y, a))$.

Theorem 7.6. *for $u \in TX, v \in TY$ we have*

$$\psi_{X,Y}(u,v) = u[v[(x,y)/y]/x] \tag{7.4}$$

Proof. We proceed by induction on u.

- If $u = (a)$ for $a \in X$, we reason by induction on v. First, if $v = (b)$ for $b \in Y$ we have

$$\begin{aligned} \psi_{X,Y}(u,v) = \psi_{X,Y}((a),(b)) &= (a,b) && \text{by (7.1)} \\ &= u[v[(a,b)/y]/x] \end{aligned}$$

  If $v = \sigma(t_1, \ldots, t_n)$ then we have

$$\begin{aligned} \psi_{X,Y}(u,v) &= \psi_{X,Y}((a), \sigma(t_1, \ldots, t_n)) \\ &= \sigma(\psi(a,t_1), \ldots, \psi(a,t_n)) && \text{by (7.2) and Lemma 7.5} \\ &= \sigma(a[[t_1/y]/x], \ldots, a[[t_1/y]/x]) && \text{by induction hypothesis} \\ &= a[[\sigma(t_1, \ldots, t_n)/y]/x] \end{aligned}$$

- Therefore our hypothesis is verified in the case where $u = (a)$. let us now show the case $u = \sigma(t_1, \ldots, t_n)$.

$$\begin{aligned} \psi_{X,Y}(u,v) &= \psi_{X,Y}(\sigma(t_1, \ldots, t_n), v) \\ &= \sigma(\psi(t_1,v), \ldots, \psi(t_n,v)) && \text{by (7.2)} \\ &= \sigma(t_1[v[(x,y)/y]/x], \ldots, t_n[v[(x,y)/y]/x]) && \text{by induction hypothesis} \\ &= u[v[(x,y)/y]/x] \end{aligned}$$

□

We can now make use of this formulation to study properties of monoidal monads in terms of their algebraic presentation.

7.2 Algebraic Characterisation of Classes of Monads

In this section, we study the algebraic consequences of the categorical properties of diverse monadic classes. For this purpose, we consider again for a monad T and a set X the set TX of Σ-terms up to equivalence under the equations in E. The core idea of our reasoning is the following: by examining the free algebra TX generated by X, we can deduce equalities which hold in every T-algebra. Once again, we consider a monoidal monad T presented by the theory (Σ, E).

Theorem 7.7. *Let t_1, t_2 be two Σ-terms generated by X. If t_1 and t_2 are equal in TX, then for all T-algebra $\mathcal{A}$ the equality $t_1 = t_2$ holds (where this time, the elements of X are seen as variables in t_1, t_2).*

Proof. We recall the monad $(\mathsf{F}_\Sigma, \eta^{\mathsf{F}_\Sigma}, \mu^{\mathsf{F}_\Sigma})$ of free Σ-terms. Let $\mathcal{A}$ be a T-algebra with carrier A (hence also a F_Σ-algebra), and let t_1, t_2 be terms of TX which we assume are equal in the theory (Σ, E). Finally, we denote by q be the quotient map $\mathsf{F}_\Sigma X \to TX$. TX is an algebra for F_Σ, therefore by the universal property of Lemma 2.61, q is the unique map such that $\eta^T_X = q \circ \eta^{\mathsf{F}_\Sigma}_X$.

We will now show that under our assumptions, $t_1 = t_2$ holds on $\mathcal{A}$. Let f be any map $X \to A$, and let $f^\#\colon \mathsf{F}_\Sigma X \to A$ be the map $\mathsf{F}_\Sigma X \to A$ obtained from it via the universal property of F_Σ (Lemma 2.61). Let us now use the universal property of T: by Theorem 2.60, there exists a unique map $f'\colon TX \to A$ such that $f' \circ \eta^T_X = f$. Now, $f' \circ q$ and $f^\#$ are two maps $\mathsf{F}_\Sigma X \to A$, moreover we have both $f^\# \circ \eta^{\mathsf{F}_\Sigma}_X = f$ and $f' \circ q \circ \eta^{\mathsf{F}_\Sigma}_X = f$. By unicity, we must have $f' \circ q = f^\#$. Finally, note that if t_1 and t_2 are equated in the theory (Σ, E), then immediately we have $q(t_1) = q(t_2)$. Hence

$$\begin{aligned} f^\#(t_1) &= f' \circ q\ (t_1) \\ &= f' \circ q\ (t_2) \\ &= f^\#(t_2) \end{aligned}$$

We conclude that $t_1 = t_2$ holds in the algebra $\mathcal{A}$, which proves our theorem. □

Note that this theorem requires to reinterpret elements of X as variables to express the final equation in any algebra of the monad. For this purpose and for more clarity, it will sometimes be convenient to rename them. We can now make use of this strategy to derive algebraic properties of monads of each class, starting with the general setting of monoidal monads.

7.2.1 Monoidal Monads

We first consider the case of monoidal monads in general and establish algebraic properties of their presentation. Their main feature is the map ψ which we have now fully studied on an algebraic level. First, recall that in our framework a monoidal monad T is

always symmetric (or 'commutative', see Theorem 2.30). In other words, the equality $T\mathsf{swap} \circ \psi = \psi \circ \mathsf{swap}$ (SYM) is verified. By making use of (7.4), we translate this equality in algebraic terms and immediately obtain the following characterisation.

Theorem 7.8. *For a set X, a monoidal monad T and for every pair of terms $s, t \in TX$, we have an equality between the following terms of $T(X \times X)$.*

$$s[t[(x,y)/y]/x] = t[s[(x,y)/x]/y] \tag{7.5}$$

Proof. Follows immediately from (7.4) and $T\mathsf{swap} \circ \psi = \psi \circ \mathsf{swap}$ applied to the pair of terms (s,t). □

By Theorem 7.7, this equation holds on any T-algebra for every monoidal monad T. However, it does not constitute a very intuitive description of the theory presenting T. In the rest of this section, we study a series of consequences of (7.5) to better understand the presentation of monoidal monads. This subject often misunderstood; for instance one common misconception is that for a symmetric monoidal ('commutative') monad, operations of the signature have to be commutative. Is is not the case, as shows for instance the distribution monad, presented by mostly non-commutative operations. In what follows, we study precisely the question and examine several cases of presentations of monoidal monads. Let us now make use of this result to derive a first equation in the case where T is presented with binary operations.

Theorem 7.9. *Let T be a commutative monad presented with Σ, E. If f, g are binary symbols of Σ, then $f(g(x,y), g(z,t)) = g(f(x,z), f(y,t))$ holds.*

Proof. By choosing $s = f(a,b)$, $t = g(c,d)$ we obtain the following equation on terms of $T(X \times X)$:

$$\begin{aligned} T\mathsf{swap} \circ \psi(s,t) &= \psi \circ \mathsf{swap}(s,t) \\ T\mathsf{swap} \circ \psi(f(a,b), g(c,d)) &= \psi(g(c,d), f(a,b)) \\ f(g((a,c),(a,d)), g((b,c),(b,d))) &= g(f((a,c),(b,c)), f((a,d)),(b,d))) \qquad \text{by (7.4)} \end{aligned}$$

By Theorem 7.7, we interpret elements of $(X \times X)$ as variables and conveniently rename them to obtain the equation $f(g(x,y), g(z,t)) = g(f(x,z), f(y,t))$, which holds in any theory presenting a commutative monad. □

Intuitively, one could say that the different binary operations of the signature must 'commute' with each other.

Corollary 7.10. *Let T be a commutative monad presented with Σ, E. If $\bullet$ is a binary symbol of Σ, then:*

1. *$(x \bullet y) \bullet (z \bullet t) = (x \bullet z) \bullet (y \bullet t)$ holds.*
2. *Moreover, if $\bullet$ has a unit e, then $x \bullet y = y \bullet x$ holds.*

Proof. 1. We apply Proposition 7.9 with $f = g = \bullet$.

2. From the previous equation we derive: $(e \bullet b) \bullet (c \bullet e) = (e \bullet c) \bullet (b \bullet e)$, then apply unit laws.

□

Therefore for any monoidal monad, if a binary symbol of the signature admits a unit, it must also be commutative. This covers for instance the cases of the finite powerset and multiset monads.

7.2.2 Affine Monads

Let us now focus on a first subclass of monoidal monads; we recall that T is affine if $T1 = 1$. By reasoning on TX as containing algebraic terms again, we can derive the following property:

Theorem 7.11. *T is affine if and only if all T-terms generated by one variable x are equal to x.*

Proof. Let t_1, t_2 be terms of $T1$, in other words terms formed on a single generator. Because $T1 = 1$ they must be equal, therefore by Theorem 7.7 the equation $t_1 = t_2$ (where all instances of $* \in 1$ are replaced with a single variable x) holds on any T-algebra. □

Again, let us give more insight on this result by studying some consequences.

Corollary 7.12. *If T is affine and (Σ, E) is a presentation of T-algebras, then:*

- Σ *contains no constant*
- *for all* f n*-ary symbol in* Σ*,* f *is idempotent:* $f(x, x, \dots x) = x$*.*

We can provide a precise and economical characterisation in the case of a presentation including only one operation symbol.

Theorem 7.13. *Let* (Σ, E) *be a presentation of* T*-algebras made up of only one symbol* f*. Then* T *is affine if and only if* f *is idempotent.*

Proof. Follows immediately from Theorem 7.11 for this case of presentation. □

Example 7.14. The *non-empty Powerset Monad* is presented with an idempotent operation $+$ and no constant, it is affine. △

7.2.3 Relevant Monads

We now move on to the class of relevant monads and recall their definition.

Definition 7.15. T is relevant if the following diagram commutes:

$$\begin{array}{ccc} TX & \xrightarrow{T\Delta} & T(X \times X) \\ {\scriptstyle\Delta}\downarrow & \nearrow_{\psi} & \\ TX \times TX & & \end{array} \tag{7.6}$$

Again, our aim is to provide algebraic properties of such monads. We make use of (7.2) to obtain the following equality between terms of $T(X \times X)$.

Theorem 7.16. *If* T *is relevant, then for all term* t *of* TX *we have*

$$t[(x, x)/x] = t[t[(x, y)/x]/y] \tag{7.7}$$

This equation does not immediately give explicit consequences, to understand it better we study a few particular cases.

Corollary 7.17. *Let* T *be a relevant monad presented with* (Σ, E)*. If* f *is a unary symbol of* Σ*, then* $f(f(x)) = f(x)$ *holds in any* T*-algebra.*

Proof. By taking $t = f(a)$ we obtain from (7.7) the equality $f((a, a)) = f(f(a, a))$, then by Theorem 7.7 we obtain the desired equation on any algebra. □

As for monoidal monads, the case of a unital binary operation yields interesting results.

Corollary 7.18. *Let T be a relevant monad presented with Σ, E. If f is a binary symbol of Σ and has a right unit $1 \in \Sigma$, then f is the left projection.*

Proof. By taking $t = f(a, b)$ we obtain the equality on $T(X \times X)$:

$$f((a, a), (b, b)) = f(f((a, a), (a, b)), f((b, a), (b, b)))$$

Let $x, y \in X$. Again make use of Theorem 7.7 and rename our variables as:

$$(a, a) \mapsto x$$

$$(a, b) \mapsto 1$$

$$(b, a) \mapsto y$$

$$(b, b) \mapsto 1$$

We obtain $f(x, 1) = f(f(x, 1), f(y, 1))$, hence $x = f(x, y)$ since 1 is the right unit. □

Similarly, a binary operation with a left unit has to be the right projection. The next theorems follows immediately and shed some light on possible presentations for relevant monads.

Corollary 7.19. *If T is a relevant monad, a presentation of T cannot have a binary operation with a left unit and a right unit.*

We call *trivial* a binary operation that is equal to a projection, as in fact one of its argument is systematically ignored.

Corollary 7.20. *If T has a presentation featuring a nontrivial binary operation with a left or right unit, then T is not relevant.*

We can clearly see with this last result that many relevant monads will not have a very expressive presentation, because of the strong requirement of (7.7). We can bring the idea of Theorem 7.16 even further to obtain a precise characterisation of relevant monads in algebraic terms.

Theorem 7.21. *Suppose T is a monoidal monad presented by $\mathbb{T}$. Then T is relevant iff for every n-ary operator f of $\mathbb{T}$ we have*

$$\overleftrightarrow{f}((x_{ij})_{ij}) \;=\; \vec{f}((x_{ii})_i)), \tag{7.8}$$

where $(x_{ij})_{ij}$ is a $n \times n$ matrix over any set X and

$$\begin{aligned} \vec{f}((y_i)_i) &\equiv f(y_1, \ldots, y_n) \\ \overleftrightarrow{f}((y_{ij})_{ij}) &\equiv f(f(y_{11}, \ldots, y_{1n}), \ldots, f(y_{n1}, \ldots, y_{nn})). \end{aligned}$$

Proof. Assume (7.8) holds. Let X be given. Any element of $T(X \times X)$ is of the form $M[\vec{x} \otimes \vec{y}]$, where M is a $\mathbb{T}$-term and $\vec{x} \otimes \vec{y} \equiv ((x_1, y_1), \ldots, (x_n, y_n))$ for $x_i, y_i \in X$. Note that $\chi(M[\vec{x} \otimes \vec{y}]) = (M[\vec{x}], M[\vec{y}])$. We will prove by induction over M that

$$\psi(M[\vec{x}], M[\vec{y}]) \;=\; M[\vec{x} \otimes \vec{y}], \tag{7.9}$$

which shows that T is relevant. So assume f is any n-ary operation of $\mathbb{T}$ and M_i are $\mathbb{T}$-terms for which (7.9) holds. We compute

$$\begin{aligned} &\psi\big(\, \vec{f}((M_i[\vec{x}])_i)\,,\; \vec{f}((M_i[\vec{y}])_i)\,\big) \\ &\qquad = \;\overleftrightarrow{f}\big(\,(\psi(M_i[\vec{x}], M_j[\vec{y}]))_{ij}\,\big) \\ &\qquad \overset{(7.8)}{=} \vec{f}\big(\,(\psi(M_i[\vec{x}], M_i[\vec{y}]))_i\,\big) \\ &\qquad \overset{(7.9)}{=} \vec{f}\big(\,(\psi(M_i[\vec{x} \otimes \vec{y}]))_i\,\big) \end{aligned}$$

and so indeed (7.9) holds for all M by induction and so T is relevant. The proof of the converse is straightforward. □

7.3 The 'Times Over plus' Law

The backbone of our work in Chapter 3 is the distributive law $ST \to TS$ obtained from a monoidal structure. In this section we examine this object on an algebraic level by studying elements of STX as S-terms over T-terms (denoted by $s[t_x/x]$) generated by the set X. We show that our method generates a very common distributive law, sometimes called 'times over plus' (see [42]).

Let S be a monad, T a monoidal monad such that a distributive law can be constructed by the method of Chapter 3. Assume that S is presented by a theory $\mathbb{S} = (\Sigma, E)$. For instance, recall that a distributive law can be constructed if $\mathbb{S}$ is linear (Theorem 3.13), or if E only contains linear or strict-drop equations and T is affine (Theorem 4.28), or finally if S is presented by linear and strict-dup equations and T is relevant (Theorem 4.34). We call λ the corresponding distributive law $ST \to TS$. First, let us study the effect of λ on simple terms of TSX by using some basic distributive law properties.

Theorem 7.22. *Let t be a term of TX, s a term of SX. We have:*

1. $\lambda((t)) = t[(x)/x]$
2. $\lambda(s[(x)/x]) = (s)$

Proof. The equality 1 follows from Diagram DL. 2, as $\lambda((t)) = \lambda \circ \eta^S_{TX}(t) = T\eta^S_X = t[(x)/x]$. The equality 2 follows similarly from Diagram DL. 1. □

In this result, the explicit notation for variables such as (x) is necessary to clearly show the type of our terms, for instance $t[(x)/x] \in TSX$. But notation aside, the intuition is that our distributive law acts almost trivially on terms which originate solely from SX or TX. A clear consequence of 1 is the action of λ on variables, namely on the term $((x)) \in STX$ for $X \in X$.

$$\lambda(((x))) = (((x))) \tag{7.10}$$

We will now study the particular case of monads whose presentation includes binary operations, as it corresponds to most known examples. Similar results can be obtained for different arities by the same method. We show that with binary symbols, our distributive law features a well-known pattern.

Theorem 7.23. *Let $\bullet, +$ be binary symbols from the theories respectively presenting S and T. Let $a, b, c \in X$. for the term $t = a \bullet (b + c) \in STX$ we have $\lambda(t) = (a \bullet b) + (a \bullet c)$.*

Proof. We recall the steps of the construction of λ, this time with an algebraic point of view.

- Recall that we represent Σ with a polynomial functor H_Σ, and that we construct from the monoidal structure of T a distributive law $\lambda_\Sigma \colon T\mathsf{H}_\Sigma \to \mathsf{H}_\Sigma T$.
- Recall the lifting $\widehat{T}$ of T defined in 3.7. $\widehat{T}$ maps a S-algebra (A, α) to the algebra $(TA, T\alpha \circ \lambda_\Sigma)$. We have shown that this lifting is monadic, which is equivalent to

the existence of a distributive law. To construct it explicitly, we follow the next two steps.

- Consider the free S algebra (SX, μ^S). $\widehat{T}$ maps it to an algebra that we denote with $(TSX, \widehat{\mu^S})$.
- Finally, we construct λ as $\widehat{\mu^S} \circ ST\eta^S$ (see Theorem 2.66).

We now apply this process to the term $t = a \bullet (b + c) \in STX$.

- First, we calculate $ST\eta^S(a \bullet (b + c)) = (a) \bullet ((b) + (c))$.
- We consider the free S-algebra (SX, μ^S). It is isomorphic via a functor $J \colon \mathcal{EM}(S) \to \mathbf{Alg}(\Sigma, E)$ to the H_Σ-algebra (SX, α). Intuitively, α maps each tuple of $\mathsf{H}_\Sigma SX$ to an S-term computed by referring to the corresponding algebraic operation. For instance, for (a, b) a pair in the summand of $\mathsf{H}_\Sigma SX$ that corresponds to $\bullet \in \Sigma$, we have $\alpha(a, b) = a \bullet b$.
- To calculate the operation of $\widehat{\mu^S}$ on $(a) \bullet ((b) + (c)) \in STSX$, we first map via J it to the corresponding element of $\mathsf{H}_\Sigma TSX$, namely the pair $((a), ((b) + (c)))$. We then apply $T\alpha \circ \lambda_\Sigma$. Recall that λ_Σ is a cotupling of monoidal maps. The pair $((a), ((b) + (c)))$ is therefore mapped to $\psi((a), ((b) + (c))) = ((a), (b)) + ((a), (c))$ (see (7.2)). Finally, $T\alpha((a), (b)) + ((a), (c)) = (a \bullet b) + (a \bullet c)$.

□

We now address the more general case of a term of STX constructed with the binary operation $\bullet$, and give an inductive formulation of the action of λ on this term.

Theorem 7.24. *Let $s[t_x/x]$ and $s'[t'_x/x]$ be two terms of STX. Then we have:*

$$\lambda(s[t_x/x] \bullet s'[t'_x/x]) = T \bullet \circ \psi(\lambda(s[t_x/x]), \lambda(s'[t'_x/x])) \tag{7.11}$$

Proof. By the correspondence in Theorem 2.66, our distributive law λ is the unique H_Σ-algebra homomorphism mapping (STX, α_{TX}) to $\widehat{T}(SX, \alpha_X)$. We consider again the

free H_Σ-algebra constructed on STX and we denote it by (STX, α). Then we have $\lambda_X \circ \alpha = \widehat{T}(STA, \alpha) \circ \mathsf{H}_\Sigma(\lambda)$.

The term $(s[t_x/x] \bullet s'[t'_x/x])$ belongs to STX and corresponds to $\alpha(s[t_x/x], s'[t'_x/x])$, where once again the pair $(s[t_x/x], s'[t'_x/x])$ belongs to the summand associated with $\bullet$ inside the coproduct $\mathsf{H}_\Sigma STX$. Hence we have:

$$\begin{aligned}
\lambda(s[t_x/x] \bullet s'[t'_x/x]) &= \lambda(\alpha(s[t_x/x], s'[t'_x/x])) \\
&= \widehat{T}(\alpha) \circ \ \mathsf{H}_\Sigma(\lambda)(s[t_x/x], s'[t'_x/x]) \\
&= \widehat{T}(\alpha)(\lambda(s[t_x/x]), \lambda(s'[t'_x/x])) \\
&= \widehat{T}(\alpha)(\lambda(s[t_x/x]), \lambda(s'[t'_x/x])) \\
&= T\bullet \circ \psi(\lambda(s[t_x/x]), \lambda(s'[t'_x/x]))
\end{aligned}$$

□

Let us now study the effect of our distributive law on a few algebraic terms.

Example 7.25. For $a, b, c, d \in X$, consider $(a + b) \bullet (c + d) \in STX$. By Theorem 7.24 we can easily compute the effect of our distributive law on this term:

$$\begin{aligned}
\lambda((a + b) \bullet (c + d)) &= T\bullet \circ \psi(\lambda(a + b), \lambda(c + d)) \\
&= T\bullet \circ \psi(a + b, c + d) && \text{by Theorem 7.22} \\
&= T\bullet (((a, c) + (b, c)) + ((a, d) + (b, d))) && \text{by Theorem 7.6} \\
&= ((a \bullet c) + (b \bullet c)) + ((a \bullet d) + (b \bullet d))
\end{aligned}$$

△

Example 7.26. For $a, b, c, d, e \in X$, consider $(a \bullet (b + c)) \bullet (d \bullet e) \in STX$. We first compute $\lambda(a \bullet (b + c))$, which amounts to $(a \bullet b) + (a \bullet c)$ by Theorem 7.23. Then we have $\lambda(d \bullet e) = (d \bullet f)$ by Theorem 7.22 (this time we omit the variable notation for clarity).

We then calculate the action of the monoidal map and make use of Theorem 7.6.

$$\begin{aligned}\lambda((a \bullet (b+c)) \bullet (d \bullet e)) &= T \bullet \circ \psi(\lambda(a \bullet (b+c)), \lambda(d \bullet e))) \\ &= T \bullet \circ \psi((a \bullet b) + (a \bullet c), d \bullet e) \\ &= T \bullet ((a \bullet b, d \bullet e) + (a \bullet c, d \bullet e)) \\ &= ((a \bullet b) \bullet (d \bullet e)) + ((a \bullet c) \bullet (d \bullet e))\end{aligned}$$

△

This 'times over plus' distributive law is a very intuitive transformation: it simply describes the fact that operations of the monad S 'distribute' over operations of T in a natural way, for instance in the way multiplication distributes over addition for natural numbers. We will see in Section 7.5 that it is not the case for all distributive laws.

Recall that a distributive law $ST \to TS$ allows to construct the composite monad TS. The question of the algebraic presentation of TS is rather tricky and we only briefly focus on it in this thesis. For more details, we refer to the concept of composite theory in [34] and [42] as well as Fabio Zanasi's thesis [40]. In a nutshell, evaluating λ on terms allows us to study the algebraic presentation of the composite monad TS. To understand precisely this presentation, we borrow this result from the works of Maaike Zwart and Fabio Zanasi [40]:

Theorem 7.27. *Let S, T be two monads respectively presented with (Σ_S, E_S) and (Σ_T, E_T) and let λ be a distributive law $ST \to TS$. First, we define a set of equations E_λ: let $s[t_x/x]$ and $t[s_y/y]$ be representatives of respectively an element in STX and an element in TSX. Then $s[t_x/x] = t[s_y/y] \in E_\lambda$ if and only if λ maps the equivalence class of $s[t_x/x]$ to the equivalence class of $t[s_y/y]$.*

TS is a monad presented by theory (Σ_{TS}, E_{TS}), where:

- $\Sigma_{TS} = \Sigma_T \cup \Sigma_S$
- $E_{TS} = E_T \cup E_S \cup E_\lambda$

Therefore, one can obtain the full presentation of the composite monad TS by combining the presentations for S and T as well the collection of all equations $\lambda(t) = t$ for t a term of STX, in a similar manner as the ones obtained in Examples 7.25 and 7.26.

7.4 Preservation of Discerning Equations

We now present a slightly different result, related to the findings of Chapter 5 but this time obtained via algebraic techniques. In this section, we characterise a new subclass of equations for which relevance is a necessary property for preservation. As we did in Chapter 5, we will consider a binary operation $\bullet$ of the signature and sometimes omit it for more clarity.

Definition 7.28. A 2-discerning equation $t_1 = t_2$ is a dup, non-drop equation, where only one variable, say x_1 out of $x_1, \ldots, x_n$ is duplicated and only one side (say t_2), for which we can distinguish the places where x_1 is duplicated in the following sense: the linear equation $s_2 = s_2'$ in $x_1, x_1', x_2, \ldots, x_n$ fixed by with $t_2 = s_2[x_1/x_1']$ and $s_2' = s_2[x_1/x_1', x_1'/x_1]$ is not derivable from $t_1 = t_2$.

Example 7.29. The equation $x \bullet (y \bullet y) = y \bullet x$ (that we will denote by $x(yy) = yx$) is *not* 2-discerning as it implies $xy = yx$ and thus in particular $x(yy') = x(y'y)$. On the other hand $y(xy) = yx$ is 2-discerning. This requires one to show that $y(xy') = y'(xy)$ isn't derivable from $y(xy) = yx$, which is easily seen by noting that all terms equal to $y'(xy)$ must start with y' as well. In fact, all of the following equations are 2-discerning, which are essentially all the different candidates on two variables besides $x(yy) = yx$.

$$(yy)x = yx \qquad (yx)y = yx \qquad (xy)y = yx$$
$$y(yx) = yx \qquad y(xy) = yx$$

△

The following theorem is the main technical result of this section, providing a new and precise characterisation of relevance:

Theorem 7.30. *Suppose $t_1 = t_2$ is a 2-discerning equation and T is a monoidal monad on* **Set**. *Then T is relevant if and only if T preserves $t_1 = t_2$.*

Proof. Assume T is a finitary monad presented by an algebraic theory $\mathbb{T}$, such that T preserves a 2-discerning equation $t_1 = t_2$. Suppose f is any n-ary operation of $\mathbb{T}$. Write m for the number of variables in t_1 aside from the duplicated one. Let X be any set with $n \times n$

matrix $(x_{ij})_{ij}$ over it. We have to show that (7.8) holds. We will take a slight detour: let Y denote the free model of $t_1 = t_2$ over $\{y_1, \ldots, y_n, r_1, \ldots, r_m\}$. Using preservation of $t_1 = t_2$, we see that

$$\begin{aligned}
&\overleftrightarrow{f}\,(\,(s_2[y_i, y_j, \vec{r}])_{ij}\,) \\
&\qquad = \overline{s_2}[\vec{f}(\vec{y}), \vec{f}(\vec{y}), \vec{r}] \\
&\qquad = \overline{t_2}[\vec{f}(\vec{y}), \vec{r}] \\
&\qquad = \overline{t_1}[\vec{f}(\vec{y}), \vec{r}] \\
&\qquad = \vec{f}(\,(t_1[y_i, \vec{r}])_i\,) \\
&\qquad = \vec{f}(\,(t_2[y_i, \vec{r}])_i\,) \\
&\qquad = \vec{f}(\,(s_2[y_i, y_i, \vec{r}])_i\,).
\end{aligned}$$

As $t_1 = t_2$ is 2-discerning the terms $s_2[y_i, y_j, \vec{r}]$ are distinct and so there exists a map $h\colon Y \to X$ such that $h(s_2[y_i, y_j, \vec{r}]) = x_{ij}$. Hence

$$\begin{aligned}
&\overleftrightarrow{f}((x_{ij})_{ij}) \\
&= \overleftrightarrow{f}\big(\,(h(s_2[y_i, y_j, \vec{r}]))_{ij}\,\big) \\
&= (Th)\big(\overleftrightarrow{f}\,(\,(s_2[y_i, y_j, \vec{r}])_{ij}\,)\big) \\
&= (Th)\big(\vec{f}(\,(s_2[y_i, y_i, \vec{r}])_i\,)\big) \\
&= \vec{f}(\,(h(s_2[y_i, y_i, \vec{r}]))_i\,) \\
&= \vec{f}(\,(x_{ii})_i)\,),
\end{aligned}$$

and so T is relevant by Proposition 7.21. □

We have explored the notions of monoidal, affine and relevant monads as well as distributive laws with an algebraic perspective. This sheds a new light on the findings of Chapters 3 to 5 and yields a series of original contributions: algebraic properties of our monadic classes of interest, description of our distributive law in terms of presentations of the monads, and finally a new characterisation of relevant monads obtained via algebraic techniques.

7.5 Algebraic View on Non-monoidal Liftings

In this section, we focus on some distributive laws which are not obtained from our monoidal method, this time with an algebraic point of view. Consider first the case of distributing the non-empty List monad $\mathcal{L}^+$ with itself. One distributive law $\lambda^{\mathcal{L}^+\mathcal{L}^+}$ is described by Manes and Mulry in [26] the following way: note that a non-empty list of non-empty lists can be written using brackets '[', ']', and commas ','. The distributive law operates by substituting every comma with '],[' and conversely, thus:

$$\lambda^{\mathcal{L}^+\mathcal{L}^+}([[a,b],[c],[d]]) = [[a],[b,c,d]] \tag{7.12}$$

The monad $\mathcal{L}^+$ is presented with an associative binary operation (which we respectively denote by $\bullet$ and $+$ for each copy of the monad). We calculate the action of $\lambda^{\mathcal{L}^+\mathcal{L}^+}$ on $a \bullet (b + c)$ and $(a + b) \bullet c$ to obtain the following equations:

$$x \bullet (y + z) = (x \bullet y) + z$$
$$(x + y) \bullet z = x + (y \bullet z)$$

This time, our equalities have little sense if we interpret $+$ and $\bullet$ as operations on real numbers. They nonetheless represent a valid distributive law, that we will call *rebracketing*. Can this distributive law be applied to other monads? We show here a series of necessary condition on such monads for this law to be usable. Let S, T be two monads respectively presented with the binary operations $\bullet, +$.

Theorem 7.31. *If $+$ is commutative and has a unit 0, and if rebracketing is a valid distributive law of S over T, then $\bullet$ is commutative.*

Proof. Consider the term $t = 0 \bullet (y + z)$. We have at the same time:

$$\begin{aligned} 0 \bullet (y + z) &= (0 \bullet y) + z && \text{(rebracketing)} \\ &= z + (0 \bullet y) && \text{(commutativity of } +) \\ &= (z + 0) \bullet y && \text{(rebracketing)} \\ &= z \bullet y && \text{(unit)} \end{aligned}$$

and:

$$\begin{aligned} 0 \bullet (y + z) &= 0 \bullet (z + y) && \text{(commutativity of } +) \\ &= (0 \bullet z) + y && \text{(rebracketing)} \\ &= y + (0 \bullet z) && \text{(commutativity of } +) \\ &= (y + 0) \bullet z && \text{(rebracketing)} \\ &= y \bullet z && \text{(unit)} \end{aligned}$$

Thus the terms $y \bullet z$ and $z \bullet y$ are equal on SX, thus by Theorem 7.7 $\bullet$ must be commutative on any S-algebra. □

In particular, rebracketing is not a distributive law of $\mathcal{L}$ over $\mathcal{P}$. Similarly, we see that this law entails invalid equations when monad T is not associative.

Theorem 7.32. *If rebracketing is a valid distributive law of S over T, then $+$ is associative.*

Proof. Consider the term $t = (a + b) \bullet (c + d)$. We have at the same time:

$$\begin{aligned} (a + b) \bullet (c + d) &= a + (b \bullet (c + d)) && \text{(rebracketing)} \\ &= a + ((b \bullet c) + d) && \text{(rebracketing)} \end{aligned}$$

and:

$$\begin{aligned} (a + b) \bullet (c + d) &= ((a + b \bullet c) + d) && \text{(rebracketing)} \\ &= (a + (b \bullet c)) + d && \text{(rebracketing)} \end{aligned}$$

We obtain the equality $(a+(b\bullet c))+d = a+((b\bullet c)+d)$ on TSX. By Theorem 7.7 and by the renaming $a \mapsto x, (b\bullet c) \mapsto y, d \mapsto z$, we obtain that the equation $x+(y+z) = (x+y)+z$ holds on any T-algebra. □

Therefore rebracketing cannot be used to distribute a monad over the free binary tree monad, as the latter is not associative.

7.6 Discussion and Related Work

Our work on algebraic techniques applied to distributive laws relates closely to Zwart and Marsden's paper [42]. In that study, the authors prove a series of 'no-go theorems', in other words results of incompatibility between monads. Their reasoning is based on translating categorical objects into algebraic language: distributive laws give rise to composite theories, and natural transformations allow to operate substitutions in terms of an algebraic theory. This allows them to establish general algebraic conditions under which no distributive law can be found between two monads. For instance, they settle many questions about the finite powerset and distribution monads: $\mathcal{P}^+$ does not distribute over $\mathcal{D}$, and $\mathcal{D}$ does not distribute over itself.

The first notable difference between [42] and our work is our focus on monoidal monads. Our results mostly involve the monoidal structure of a monad T to compose it with another monad S, and in the case where the conditions of Chapter 5 are not verified, we cannot say that no distributive law can be found: only that the monoidal one fails. The study undertook by Zwart and Marsden is more general, as it encompasses every possible distributive law. However, our work also differs by providing cases where a distributive law can effectively be built, thus completing the study of [42] with positive results.

Section 7.3 features a very brief connection with the study of composite theories. This concept, which described how two algebraic theories can be consistently combined, is the subject of precise studies in [34] and [42]. Composite theories can be seen as an algebraic counterpart to distributive laws, but are not the focus of this thesis. Fabio Zanasi's thesis [40] was also a source of inspiration as he precisely establishes the presentation of a composite monad, which corroborates some of Maaike Zwart's recent results. Theorem 7.27 also suggests an interesting direction for future works: the reduction of this composite theory to a simpler one, possibly replacing E_λ with a finite set generating the whole theory instead of considering all equations $\lambda(t) = t'$.

Bibliography

[1] Steve Awodey. *Category theory*. Oxford University Press, 2010.

[2] Jon Beck. Distributive laws. In *Seminar on triples and categorical homology theory*, pages 119–140. Springer, 1969.

[3] Filippo Bonchi, Alexandra Silva, and Ana Sokolova. The power of convex algebras. In *28th International Conference on Concurrency Theory (CONCUR 2017)*, Leibniz International Proceedings in Informatics (LIPIcs), 2017.

[4] Filippo Bonchi, Ana Sokolova, and Valeria Vignudelli. The theory of traces for systems with nondeterminism and probability. In *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–14. IEEE, 2019.

[5] Marcello M Bonsangue, Helle Hvid Hansen, Alexander Kurz, and Jurriaan Rot. Presenting distributive laws. In *International Conference on Algebra and Coalgebra in Computer Science*, pages 95–109. Springer, 2013.

[6] Ronald V Book and Friedrich Otto. String-rewriting systems. In *String-Rewriting Systems*, pages 35–64. Springer, 1993.

[7] Ashok K Chandra and Larry J Stockmeyer. Alternation. In *17th Annual Symposium on Foundations of Computer Science (sfcs 1976)*, pages 98–108. IEEE, 1976.

[8] Eugenia Cheng. Iterated distributive laws. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 150, pages 459–487. Cambridge University Press, 2011.

[9] JRB Cockett and RAG Seely. Linearly distributive functors. *Journal of Pure and Applied Algebra*, 143(1-3):155–203, 1999.

[10] Fredrik Dahlqvist, Alexandra Silva, and Louis Parlant. Layer by layer: composing monads. In *ICTAC*, 2018.

[11] ND Gautam. The validity of equations of complex algebras. *Archiv für Mathematische Logik und Grundlagenforschung*, 3(3-4):117–124, 1957.

[12] G Grätzer and H Lakser. Identities for globals (complex algebras) of algebras. In *Colloquium Mathematicum*, volume 56, pages 19–29. Institute of Mathematics Polish Academy of Sciences, 1988.

[13] Martin Hyland, Paul Blain Levy, Gordon Plotkin, and John Power. Combining continuations with other effects. In *Proc. Continuations Workshop*, 2004.

[14] Martin Hyland, Gordon Plotkin, and John Power. Combining effects: Sum and tensor. *Theoretical Computer Science*, 357(1):70–99, 2006.

[15] Bart Jacobs. Semantics of weakening and contraction. *Annals of pure and applied logic*, 69(1):73–106, 1994.

[16] Bart Jacobs. Coalgebraic trace semantics for combined possibilitistic and probabilistic systems. *Electronic Notes in Theoretical Computer Science*, 203(5):131–152, 2008.

[17] Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. In *CMCS*, volume 12, pages 109–129. Springer, 2012.

[18] Peter T Johnstone. Adjoint lifting theorems for categories of algebras. *Bulletin of the London Mathematical Society*, 7(3):294–297, 1975.

[19] David J King and Philip Wadler. Combining monads. In *Functional Programming, Glasgow 1992*, pages 134–143. Springer, 1993.

[20] Bartek Klin and Jurriaan Rot. Coalgebraic trace semantics via forgetful logics. In *International Conference on Foundations of Software Science and Computation Structures*, pages 151–166. Springer, 2015.

[21] Bartek Klin and Julian Salamanca. Iterated covariant powerset is not a monad. *MFPS XXXIV*, 2018.

[22] Anders Kock. Bilinearity and cartesian closed monads. *Mathematica Scandinavica*, 29(2):161–174, 1972.

[23] Anders Kock. Strong functors and monoidal monads. *Archiv der Mathematik*, 23(1):113–120, 1972.

[24] Harald Lindner. Affine parts of monads. 33:437–443, 1979.

[25] Saunders Mac Lane. *Categories for the working mathematician*, volume 5. Springer, 2013.

[26] Ernie Manes and Philip Mulry. Monad compositions i: general constructions and recursive distributive laws. *Theory and Applications of Categories*, 18(7):172–208, 2007.

[27] Ernie Manes and Philip Mulry. Monad compositions ii: Kleisli strength. *Mathematical Structures in Computer Science*, 18(3):613–643, 2008.

[28] Micah Blake McCurdy. Graphical methods for tannaka duality of weak bialgebras and weak hopf algebras in arbitrary braided monoidal categories. *arXiv preprint arXiv:1110.5542*, 2011.

[29] Paul-André Melliès. Functorial boxes in string diagrams. In *International Workshop on Computer Science Logic*, pages 1–30. Springer, 2006.

[30] Eugenio Moggi. *An abstract view of programming languages.* 1989.

[31] Eugenio Moggi. Notions of computation and monads. *Information and computation*, 93(1):55–92, 1991.

[32] Philip Mulry. Notions of monad strength. *Electronic Proceedings in Theoretical Computer Science*, 129, 09 2013. doi:10.4204/EPTCS.129.6.

[33] Louis Parlant, Jurriaan Rot, Alexandra Silva, and Bas Westerbaan. Preservation of Equations by Monoidal Monads. In Javier Esparza and Daniel Krá, editors, *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 77:1–77:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2020/12746, doi:10.4230/LIPIcs.MFCS.2020.77.

[34] Maciej Pirog and Sam Staton. Backtracking with cut via a distributive law and left-zero monoids. *Journal of Functional Programming*, 27, 2017.

[35] Tetsuya Sato. The giry monad is not strong for the canonical symmetric monoidal closed structure on meas. *Journal of Pure and Applied Algebra*, 222(10):2888–2896, 2018.

[36] Ana Sokolova, Bart Jacobs, and Ichiro Hasuo. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3, 2007.

[37] Daniele Varacca. *Probability, nondeterminism and concurrency: two denotational models for probabilistic computation.* PhD thesis, BRICS, 2003.

[38] Daniele Varacca and Glynn Winskel. Distributing probability over non-determinism. *Mathematical Structures in Computer Science*, 16(1):87–113, 2006.

[39] Moshe Y Vardi. Alternating automata: Unifying truth and validity checking for temporal logics. In *International Conference on Automated Deduction*, pages 191–206. Springer, 1997.

[40] Fabio Zanasi. *Interacting Hopf Algebras: the theory of linear systems.* PhD thesis, 2018.

[41] Marek Zawadowski. The formal theory of monoidal monads. *Journal of Pure and Applied Algebra*, 216(8-9):1932–1942, 2012.

[42] Maaike Zwart and Dan Marsden. No-go theorems for distributive laws. In *LICS*, pages 1–13. IEEE, 2019.

Acknowledgements

While working on the contents of this thesis, I had the pleasure of collaborating with many inspiring researchers, and highly benefited from the help of numerous people. I couldn't thank enough Alexandra Silva for the constant support, availability and kindness she showed throughout the years as my main supervisor ; her everlasting patience and generosity has been greatly appreciated. I am very grateful for the many exciting research sessions with Fredrik Dahlqvist and for his support as a secondary supervisor. I would like to offer my special thanks to Jurriaan Rot for his ever encouraging and pleasant assistance, and for inviting me to Nijmegen for a very inspiring research week. I would like to express my gratitude to Maaike Zwart and Dan Marsden for the productive discussions and research sessions we conducted in Oxford and London. I wish to thank Bas Westerbaan, Robin Piedeleu and Gerco Van Heerdt for all the stimulating conversations and published work on which we collaborated, as well as Bartek Klin and Julian Salamanca for their valuable and constructive suggestions. I am also very grateful for the the wholesome environment provided by the SilvaLab team and their guests : Tobias Kappé, Fabio Zanasi, Matteo Sammartino, Jana Wagemaker, Benjamin Kaminski, Joshua Moerman, Paul Brunet, Tao Gu and Simon Docherty. In addition, I would like to thank Laura-Louise Fairley for her encouragements and her help with proofreading, and my parents for their endless assistance. Finally, I could not have completed this thesis without the support of my friends Antoine Pierson, Hugo Prager, Pierre Ohlmann and Oscar Gayraud who provided stimulating discussions throughout the years as well as happy distractions to rest my mind outside of my research.