Completely Iterative Monads in Semantics of Coinductive Programs

Maciej Piróg
Kellogg College
University of Oxford

A dissertation submitted for the degree of
Doctor of Philosophy
Michaelmas 2014

Completely Iterative Monads in Semantics of Coinductive Programs

Maciej Piróg
Kellogg College, Oxford

D.Phil. in Computer Science
Michaelmas 2014

Some programs are not merely sets of batch instructions performed in isolation. They interact, either directly with the user, or with other threads and resources. This dissertation tackles the problem of mathematical description (denotational semantics) of the observable behaviour of such programs.

In the tradition of denotational semantics and functional programming, one can distinguish between pure computations, which are regarded as mathematical functions, and effectful ones, like those generating behaviour. Both effects in general and behaviour of interactive systems have been thoroughly studied, and they both have elegant category-theoretic explanations: the frameworks of, respectively, monads and coalgebra. The point of this dissertation is to explore the area where the two meet.

The thesis of this dissertation is that the right kind of monads to describe the observable behaviour of programs are completely iterative monads (cims), introduced by Elgot and more recently studied by Adámek and others. They are monads equipped with a certain corecursion scheme that allows us to describe the computation in a coinductive, step-wise manner.

To support this, we introduce a formal coinductive semantics parametrised with a cim. We study its properties and show that it instantiates to a number of known approaches, based on metric spaces and final coalgebras.

Then, we focus on studying properties of cims, especially those important in semantics and programming, like composability. We show that a number of constructions used in denotational semantics to model different aspects of behaviour are instances of the constructions that we introduce. The most important structure that we study are coinductive resumptions, generalising previous results by Moggi or Hyland, Plotkin, and Power.

The language of our development is category-theoretic, and so are the properties that we investigate. We are interested in universal properties, distributive laws, algebras, and monadicity. Thus, the results apply not only to semantics and programming, but can be construed as a general investigation of algebraic structures with iteration.

There goes Bill
From the mill
Toting millet on his back.
But through a hole
The millet goes
Out from the sack.

'The less in bag,
The less to carry,'
Enjoys himself Billy,
But when back home
The grains—all gone,
Realises silly.

So back he goes
Picking those
Crumbs he left behind,
Until the grains
Are all regained
And in the bag confined.

There goes Bill...

— Julian Tuwim

(A Polish repetitive nursery rhyme, trans. MP)

Acknowledgements

I would like to thank my supervisor, Jeremy Gibbons, for letting me explore a myriad of different ideas and directions in my research, for being always appreciative and helpful.

I would like to express my gratitude to Ralf Hinze (who acted as my departmental advisor and internal examiner) and Ben Worrell, who both assessed my transfer and confirmation, as well as my external examiner Jiří Adámek for giving me tonnes of helpful references and remarks.

I thank my love, Klaudia, who must be the most patient and forgiving person in the whole creation; always encouraging, yet—I must admit it—sometimes a bit distracting.

And to my parents and my sister Anna, always curiously checking out on my progress.

I was fortunate to meet many magnificent academics and students in the Department, who greatly influenced my work, especially the members of the Algebra of Programming group and friends: Richard Bird, Geraint Jones, Josh Ko, Tom Harper, Dan James, Nick Wu, Kwok-Ho Cheung, José Pedro Magalhães, Faris Abou-Saleh, and Sam Staton.

Equally enjoyable was the opportunity to meet all the other computer scientists and practitioners from outside of Oxford: Tarmo Uustalu, Stefan Milius, Larry Moss, Wouter Swierstra, Francesco Cesarini, Danel Ahman, Laurence Day, Eugen Jiresch, Rob Steward, and Tomas Petricek.

The list wouldn't be complete without my friends from Kellogg: Cooper, Nikis Meletiadis, Alessandro R. G., François Laperche, Davide Puddu, and Luke Seamone.

Thanks go also to my fellow computer scientists whose paths crossed mine a while back in Wrocław, and with whom I still collaborate one way or another: Darek Biernacki, Filip Sieczkowski, Agata Murawska, Jurek Marcinkowski (who happened to be a visiting scholar in Oxford in the fall of 2010), Leszek Pacholski, Marek Materzok, Rafał Szałański.

Finally, I'd like to thank my personal Java expert, Piotrek Kafel, to whom I once showed a thick book on one aspect of software development and asked him what the book was about, as I had failed understand anything form the introductory chapter. I received a short and precise answer, explaining the *entire* book in one sentence. That moment, I realised that it is possible to write a fairly successful longish piece with little content. Hence this dissertation.

Contents

Chapter 1

Introduction

1.1 In one sentence

In this dissertation, we use completely iterative monads—introduced in the 1970s by Elgot and more recently studied by Adámek and others (see [84])—as a device for specifying formal semantics of effectful coinductive programs and as useful structures in programming with infinite data structures.

1.2 Motivation

Some programs cannot be treated merely as batch calculations that silently transform a predefined input into a single answer, since they interact with their environment, concurrent processes, the human user, or the network. In such cases, we are often less interested in obtaining any final results than we are interested in the runtime behaviour of the (possibly non-terminating) program itself—think a web server or a system with a graphical user interface. A similar pattern can be identified in the case of programming with infinite data structures common in non-strict languages, such as Haskell [99]. A program—or, more often, a function that is a part of a producer-consumer scheme (see Hughes [64])—incrementally unfolds the structure, for example, a stream containing the subsequent digits of the expansion of π [45]. Such programs are often called *corecursive* or *coinductive*, as in the mathematical principle of coinduction, which is used for defining and reasoning about structures in terms of 'decomposition' or 'observation' (see Sangiorgi [107]).

The goal of this dissertation is to explore denotational (category-theoretic in particular) semantics of coinductive programs. Our starting point are the two following general questions: 1) *How does one describe the behaviour of a program?* and 2) *How*

does one model impurity in programming languages (for example, effects that generate behaviour)? Such considerations are hardly new to computer science. Both of these questions have inspired a great amount of research, resulting in two successful category-theoretic frameworks based on, respectively, coalgebra and monads.

Moggi [89] used monads to provide a modular semantics for λ-calculus with computational effects. Various kinds of effects are formalised as monads (possibly with additional structure, like strength) on a category (also with additional structure, like cartesian closure to model higher-order functions). An impure function that takes a value of a type A to a result of a type B is not modelled simply as a mathematical function $A \to B$, but as a Kleisli arrow $A \to MB$, where M is a monad representing a particular kind of effects (like non-determinism, mutable store, or input/output). In this setting, our question (2) becomes: *What kind of monads can be used to model effects that generate behaviour?* An obvious instance of the problem is given for the monadic input/output system implemented in Haskell, where the IO monad is used to explicitly structure code that generates behaviour. In this case, we ask for mathematical models of the IO monad.

The problem of describing evolving systems is also not a new one. A plethora of independent solutions have been proposed to deal with the notion of the behaviour of a system, including automata theory initiated by McCulloch and Pitts [81], process calculi invented by Milner [87], or modal and temporal logics first used for this purpose by Pnueli [104]. More recently, a large class of these developments have found a common description: the framework of coalgebra, which is the category-theoretic implementation of the notion of coinduction (see, for example, Adámek [8] or Jacobs and Rutten [68]). In a nutshell, it is simply the dual of the more common initial algebra semantics [48]: we recognise the set of possible single observations about the behaviour of a system as an endofunctor F on a category, and we represent the system as a morphism $f : X \to FX$, where we call X the *carrier*.

Coalgebra provides both operational and denotational approaches to semantics. The former is obtained when we understand the coalgebra as a state-transition system and the carrier is the type of internal states. The latter is obtained with finality. The carrier of the final coalgebra for the functor, denoted νF, is identified with the type of trees of execution possible for the given kind of behaviour. The induced unique homomorphism $[\![f]\!] : X \to \nu F$ is the 'trace semantics' of the system, which abstracts away the notion of the internal state.

The question thus becomes how to combine these two frameworks. On one hand, we want the semantics to be structured in a compositional way using monads. On

the other hand, we want to employ a coalgebraic interpretation of programs.

An obvious example of a 'coalgebraic' monad is given by finite and infinite terms generated by a signature, or, more generally, an endofunctor on a category. Formally, for an endofunctor Σ, it is given via the carriers of final coalgebras (provided they exist) as

$$\Sigma^\infty A = \nu X.\Sigma X + A,$$

where $\nu X.\Sigma X + A$ is the carrier of the final coalgebra of the functor $\Sigma(\text{-}) + A$, where A acts as the type of variables. It is a monad, in which the multiplication is given by substitution, while the unit by embedding of variables.

The endofunctor Σ is the behaviour endofunctor, and a value of the monad Σ^∞ is a trace with variables, which allow sequential composition of programs. For example, to model programs with input/output, we set $\Sigma X = (O \times X) + X^I$, for a type of input I and output O.

In many cases, however, the construction given above is not enough, especially when subsequent steps of the computation are meant to interact. A typical example is given by mutable state. In the inductive case, it is often modelled by the state monad in a cartesian closed category, $\mathsf{St}\, A = (S \times A)^S$, where S is the type of state being mutated. A computation in this monad, given an initial state, returns a final state paired with a final value, forgetting all the states in between. In the coinductive case, however, we often want to witness the alteration of the state, for example, in form of a stream of the subsequent states. Such a stream can be defined by means of infinite terms as

$$\overrightarrow{S} A = (S \times (\text{-}))^\infty A.$$

A version of the state monad that coinductively unfolds such a stream of intermediate states and gives a final answer (only when the stream is finite!) can be given as (see Section 3.4)

$$\mathsf{Sts}\, A = (\overrightarrow{S} A)^S.$$

Another relevant structure is resumptions, introduced by Milner [86] in the 1970s to denote the external behaviour of a communicating agent in concurrency theory. As given by Abramsky [2] in category-theoretic terms, a resumption is an element of the carrier of the final coalgebra νR of the functor $RX = (X \times O)^I$ on $\textsc{Set}$, where I and O are the sets of possible inputs and outputs respectively. Informally, a resumption is a function that consumes some input and returns some output paired with another resumption, and finality implies that the process of consuming and producing can be iterated indefinitely. There are many possible generalisations, for example to

the final coalgebra of the functor $\mathcal{P}^{\text{fin}}((\text{-}) \times O)^I$ for agents with finitely-branching nondeterministic behaviour, or, depending on the computational effect realised by the agent, any monad in place of $\mathcal{P}^{\text{fin}}$, as studied by Hasuo and Jacobs [55].

Cenciarelli and Moggi [28] noticed that, similarly to the monad of terms, resumptions can be given a monadic structure, and defined the inductive resumption monad in a general fashion (in any category with coproducts and enough final coalgebras) as $TA = \mu X.M(\Sigma X + A)$ (the carrier of the initial algebra of a functor $M(\Sigma(\text{-}) + A)$), where M is a monad, Σ is an endofunctor, and A is an object of variables, which allows to sequentially compose resumptions. Their resumption monad is given by initial algebras, so it is not exactly a generalisation of the resumptions studied by Milner. Intuitively, this monad models resumptions that can be iterated only finitely many times. Thus, it is natural to ask about final coalgebras of functors of the shape $M(\Sigma(\text{-}) + A)$, that is, the *coinductive* resumption monad.

The inductive resumption monad can alternatively be given as the composition $M(\Sigma M)^*$, where $(\Sigma M)^*$ is the free monad generated by ΣM as an endofunctor. In this dissertation, we discuss the monad $M(\Sigma M)^\infty$. Also, we generalise the two constructions to, respectively, MS^* and MS^∞ for any right M-module S, see Chapter 4.

Constructions similar to the examples above appear in the literature in various forms and disguises, see Section 6.2 for references. In this dissertation, we look for their generalisations and unifying properties.

We turn our attention to the line of research initiated by Elgot *et al.* [36, 37], later continued by Aczel, Adámek, Milius, and Velebil [6, 12, 14, 15, 84], as well as Moss and Milius [85, 91]. Elgot's original idea was to use algebraic tools to describe computation in general, without recourse to domain theory. One of his results was the introduction of *completely iterative monads* (cims) [37], later generalised by Adámek *et al.* [6], which are monads equipped with a certain corecursion scheme. In short, a monad M is a cim if for all morphisms (called *equation morphisms*)

$$e : X \to M(X + A)$$

that satisfy a property called *guardedness*, there exists a unique 'solution' of e, denoted

$$e^\dagger : X \to MA,$$

that is coherent with the monadic structure of M (all the necessary definitions are given in Chapter 2). We think of X as the type of variables (or: seeds of the corecursion), while A is the type of final values (parameters). All three examples given

above (infinite terms, coinductive state, and resumptions) are examples of cims. The first example is a very important one: for an endofunctor Σ, the monad Σ^∞ is the free object in the category of cims generated by Σ.

An advantage of the axiomatic approach of cims is that we can study properties of coinductive programs independently of any (co)recursive structure of the underlying category. Usually, one starts with a domain, that is a category of (or enriched with) complete partial orders or complete metric spaces, and study fixed points à la Kleene or Banach. Here, we are interested in the properties that follow solely from the definition of monads with iteration. Obviously, concrete instances of such monads can come from the domain-theoretic constructions. For example, in Chapter 3, we show that the 'delay' monad known from the metric semantics is a cim, and the Banach unique fixed point theorem accounts for the existence and uniqueness of solutions. A similar approach was taken by Milius and Moss [85] in their category-theoretic description of solutions to recursive program schemes using Elgot algebras [85] (that is, Eilenberg–Moore algebras for free cims).

The category-theoretic approach has many advantages: generalisations to other categories than Set, more conscious proof techniques and identification of assumptions, duality. Moreover, category theory provides guidelines on what kind of properties and structures one should look for in order to collect a number of useful tools. For example, when introducing a new monad, one could try to find a 'nicer' or 'easier' characterisation of their Eilenberg–Moore algebras, which automatically gives one a number of benefits, like a description of distributive laws of that monad [25, Ch. 9], or its continuation-passing-style version via the codensity monad construction [59].

1.3 Goals and a by-product

There are two main goals of this dissertation:

– We show how one can use the solution property of cims (and other monads with iteration) to give coinductive semantics to a programming language with an iteration operator. The main gain is simplicity, since no reference to any additional structure of the underlying category, like order or metric enrichment, is required. We also manage to unify semantics based on different structures (such as metric, ultrametric, and final coalgebras), and find a high-level, algebraic description of phenomena like extensional collapse, which relates metric and cpo-based models.

– We study cims: their properties, examples, and general constructions. Our most important example is the *coinductive* resumption monad. Thanks to our categorical, high-level approach, we obtain a number of interesting properties of our constructions for free.

There are also some secondary goals, which are concerned with program construction and verification. They are not explored in detail in this dissertation, but the constructions and properties tackled are relevant to the following applications:

– Since most of the constructions presented in this dissertation are constructed by means of final coalgebras, they are expressible in any general-purpose programming language that supports coinductive types. Thus, we provide a handful of new data structures that can be used in program construction, together with a formal theory behind them. Structures similar to resumptions do appear in applications, for example in the context of so-called lazy input/output in Haskell [72].

– Another reason to study such data structures is down-to-earth reasoning and testing of interactive programs. The constructions that use final coalgebras to describe semantics are expressible in Haskell, leading to what is known as *functional specification* of effects. Such specifications were used by Swierstra [111] in his PhD dissertation (see also Swierstra and Altenkirch [112]) to model Haskell's IO monad (but only in the inductive fashion). The gain is that we can reason about effectful programs as if they were pure, and, more importantly, we can automatically test them with tools such as QuickCheck [30].

There is also a by-product that is worth mentioning: We point out the relevance of the notion of modules over monads in semantics and programming. For a monad M, a (right) M-module is an endofunctor S together with a natural transformation $SM \to S$ coherent with the monadic structure of M. Modules play two important roles in this dissertation: they were used by Adámek *et al.* to formalise the notion of guardedness (with some additional structure they can be seen as properties, see Section 2.3.2), and they are used as building blocks of our resumption monads. For example, given an endofunctor Σ, the composition ΣM can—quite trivially—be given a structure of an M-module. That is why our inductive resumption monad MS^* subsumes Cenciarelli and Moggi's [28] construction $M(\Sigma M)^*$.

1.4 Contributions and structure

This dissertation is based on three conference papers [100, 101, 102]. They are co-authored by Jeremy Gibbons, though all the technical results are mine. The material was extended and the presentation improved. The material given in Chapter 4 and Sections 4.2.2 and 5.3 has not been published before.

The following summarises the contributions presented in each chapter.

Chapter 2. This chapter introduces the preliminaries needed for the rest of the dissertation. The first part of the chapter introduces the notations and assumptions that are used throughout the dissertation. It surveys the definitions of monads, adjoint functors, distributive laws, and liftings, and lists their most important properties. They are basic fundamental results devised in the early days of category theory.

The second part of the chapter introduces completely iterative monads (cims), and—as technical devices—completely iterative algebras (cias) and Elgot algebras. All of this material can be found in Adámek *et al.*'s works [6, 12, 14, 15, 84].

Chapter 3. This chapter demonstrates how to use cims in semantics of programming languages. We first introduce the *composition theorem*, which states that a monad that arises as a composition of a cim with an adjunction (that is, UMF, where M is a cim and $F \dashv U$) inherits the complete iterativity from M.

Then, we introduce a generic coinductive semantics of a generalised While language. The semantics is parametrised with a cim that represents the underlying effect (not necessarily mutable state). One instance of our generic semantics is the metric approach in the style of America and Rutten [19] and Escardó [38]. We also show a trace semantics, which is based on a version of the state monad that accumulates the intermediate states.

Chapter 4. This chapter is an interlude concentrating on modules over monads. In the first part of the chapter, we develop a basic theory of modules, which extends some known results, for example by Dubuc [33]. In detail, we show a relationship between modules and adjunctions, and establish an isomorphism between appropriate distributive laws and liftings. These technical results are used in subsequent chapters to deal with the notion of guardedness.

Then, we introduce the inductive resumption monad MS^*, where M is a monad, S is an M-module, and S^* is the free monad generated by S as an endofunctor. We show that MS^* is canonical: it is the free monad generated by the module S.

More precisely, MS^* is the free object generated by S in the category of monads with respect to a 'diagonal' functor from the category of monads to the category of modules.

Finally, we introduce the notion of algebras for an M-module S. Such an algebra consists of an algebra for S as an endofunctor, an Eilenberg–Moore algebra for M, and a coherence condition. We show that algebras for an M-module S are precisely Eilenberg–Moore algebras for the monad MS^*.

Chapter 5. In this chapter, we introduce the coinductive resumption monad MS^∞. We show that it is a cim. If M is also a cim, we show that there exist solutions of equation morphism that unfold the structure of M and S^∞ simultaneously. This result requires a more complicated proof technique than the composition theorem.

We characterise the monad MS^∞ with a universal property. If we consider a slightly different category: the category of monads completely iterative with respect to two-sided modules, it follows that the monad $M(\Sigma M)^\infty$ is a coproduct od M and Σ^∞. This shift of categories is not too expensive: every cim has a free extension to the two-sided case, and this extension does not change the 'monad' part of the cim.

We also discuss Eilenberg–Moore algebras for the coinductive resumption monad. We characterise them as algebras similar to the algebras for modules: we additionally want the property that the S-algebra part is an Elgot algebra with some additional coherence.

Chapter 6. This section summarises the introduced constructions, puts them in the context of related work, and discussed a number of ideas that are relevant to the material presented in this dissertation, but have not yet been worked out in detail. They include similar constructions (for example, coproducts of ideal cims, or a finitary case), generalisations (like the 2-categorical approach), and applications (different kinds of semantics, dualisation).

Appendix A. Because of their length, some proofs in the main text are omitted or given only as sketches. In such cases, the reader will find the full proofs in this appendix. The proofs are mostly longish calculations—not always trivial, but not very enlightening either.

Appendix B. The notion of guardedness of equation morphisms is formalised in terms of ideal monads. They can be thought of as predicates on monads (intuitively, 'this computation performs a visible action'). The second appendix expands on the topic by proving that so-called strict ideals form a fibration (similar to the subobject fibration) over the category of monads. We also discuss predicates on composite monads that arise from distributive laws.

Chapter 2

Preliminaries: monads and cims

In this chapter, we summarise some category-theoretic preliminaries needed for the material presented in the subsequent chapters. In Section 2.1, we give the basic notations. Then, in Section 2.2, we discuss the rudiments of the theory monads and adjunctions. A reader familiar with the subject on the level of an introductory textbook (such as Barr and Wells's [25]) can in good conscience skip this part, since the presented results are mostly standard and a summary of notation is displayed at the very end of the dissertation in the 'Index and notation' section.

Finally, in Section 2.3, we discuss the essential definitions and results concerning monads and algebras with iteration. This line of research was initiated by Elgot *et al.* [36, 37] and continued by Aczel, Adámek, Milius, and Velebil [6, 12, 14], as well as Moss [91] and Ghani *et al.* [43, 42]. Again, a reader familiar with completely iterative algebras and monads [84] and Elgot algebras [15] will not acquire much new knowledge here. We aim for a compact presentation in reference style. For full proofs and more examples see the relevant papers by the aforementioned authors and the in-text references.

In detail, we discuss: monads and their relationship with adjunctions, free objects, and different kinds of distributive laws and their relationship with liftings to Eilenberg–Moore and Kleisli categories. This gives us not only a better conceptual understanding of the objects used in semantics and programming, but also a powerful abstract tool set to manipulate and reason about them. A good example is the construction proposed by Hyland, Plotkin, and Power [65] of the monadic structure of Cenciarelli and Moggi's [28] resumption monad, described in Section 2.2.8. A similar approach is taken throughout this dissertation. Then, we concentrate specifically on the theory of monads with iteration introduced by Adámek *et al.* [6]. We introduce modules over monads (which are structures composable with a monad on one side—left or right), for which we find new applications in Chapters 4 and 5. We present the

concept of idealised monads, which is used to model guardedness, and finally monads and algebras with iteration.

2.1 Notation

By $\mathbb{C}, \mathbb{D}, \ldots$ we denote categories. We reserve $\mathbb{C}$ for some base category with finite coproducts. Variables that represent objects in different categories are usually named A, B, C, X, Y, Z.

By letters from the middle of the alphabet ($F, G, H, J, \ldots$) we denote functors. The category with functors of the type $\mathbb{C} \to \mathbb{D}$ as objects and natural transformations as morphisms is denoted $\mathbb{D}^{\mathbb{C}}$. We denote the composition of functors by juxtaposition, so FG is a functor defined as $(FG)A = F(GA)$. The letters M, N, T, K usually denote endofunctors furnished with a monadic structure. We always use η and μ for the unit and the multiplication respectively. If there is more than one monad in context, we use superscripts, like η^M and μ^M, to assign the natural transformation to the appropriate monad. We abuse notation by identifying a monad with its underlying endofunctor, but no confusion should arise. We follow the same pattern for other kinds of endofunctors with additional structure, like modules over monads and idealised monads.

For two natural transformations $f : F \to G$ and $g : G \to H$, we denote their sequential composition in the same way as the composition of morphisms, that is $g \cdot f : F \to H$. For two natural transformations $h : F \to G$ and $k : H \to J$, we denote their parallel composition (aka the Godement product [47]) as $h * k : FH \to GJ$. The components of a natural transformation are accessed by subscripts, like $k_A : HA \to JA$, while the parallel composition of a natural transformation with a functor is denoted $kF = k * \mathsf{id}_F : HF \to GF$. For the sake of readability, we sometimes omit the subscripts, especially in the cases when they can be easily reconstructed from the explicitly given types of the domain and codomain.

Coproducts are denoted $A + B$, the coproduct injections are called inl and inr, and the coproduct mediator of two morphisms is displayed as $[f, g]$. Products are denoted $A \times B$, the projections are called outl and outr, and the mediator is denoted $\langle f, g \rangle$.

2.2 Monads and adjunctions

2.2.1 Algebras and coalgebras

Definition 2.1. For an endofunctor F and an object (called a *carrier*) A, we call a pair $\langle A, a : FA \to A\rangle$ an *F-algebra* (or: an algebra for F). A morphism between two algebras $\langle A, a : FA \to A\rangle$ and $\langle B, b : FB \to B\rangle$ is given by a morphism $f : A \to B$ such that $f \cdot a = b \cdot Ff$. We call the category of F-algebras and their morphisms $\textsc{Alg}(F)$.

Notation 2.2. Since the morphisms of the categories $\textsc{Alg}(F)$ and $\textsc{Coalg}(F)$ (defined below) are also morphisms in the base category, to avoid confusion, we call them *homomorphisms*.

An important role is played by the initial algebra (if it exists). Intuitively, it can be seen as the least fixed point of a functor.

Notation 2.3. If it exists, the initial algebra for a functor F (that is, the initial object in $\textsc{Alg}(F)$) is denoted $\langle \mu F, \chi^F : F\mu F \to \mu F\rangle$. We omit the superscript in χ^F when F is obvious from the context. The unique homomorphism from the initial algebra to an algebra $\langle A, a : FA \to A\rangle$ (called a *fold*) is denoted $(\!|a|\!) : \mu F \to A$.

Note that we use μ both for the multiplication of a monad and the carrier of an initial algebra for a functor. It is always clear from the context which one is meant. The fact that μF is a fixed point was first noticed by Lambek [75]:

Lemma 2.4. *If the initial F-algebra exists, the morphism χ^F is an isomorphism.*

Dual to algebras are coalgebras:

Definition 2.5. For F and A as above, we define an F-coalgebra as a pair $\langle A, c : A \to FA\rangle$. A morphism between two coalgebras $c : A \to FA$ and $d : B \to FB$ is given as a morphism $f : A \to B$ such that $d \cdot f = Ff \cdot c$. We call the category of F-coalgebras as objects and appropriate morphisms $\textsc{Coalg}(F)$.

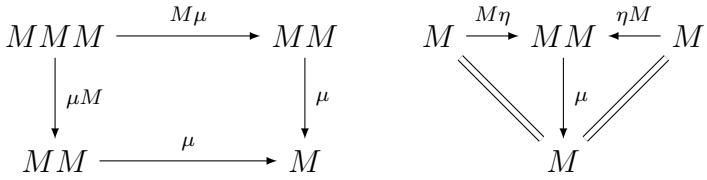
Notation 2.6. If the final F-coalgebra (that is, the final object in $\textsc{Coalg}(F)$) exists, it is denoted $\langle \nu F, \psi^F : \nu F \to F\nu F\rangle$. We omit the superscript in ψ^F when F is obvious from the context. The unique homomorphism from a coalgebra $\langle A, c : A \to FA\rangle$ to the final F-coalgebra (called an *unfold*) is denoted $[\![c]\!] : A \to \nu F$.

By duality, Lambek's lemma holds for final coalgebras as well:

Theorem 2.7 (Lambek's dual lemma)**.** *If the final F-coalgebra exists, the morphism ψ^F is an isomorphism.*

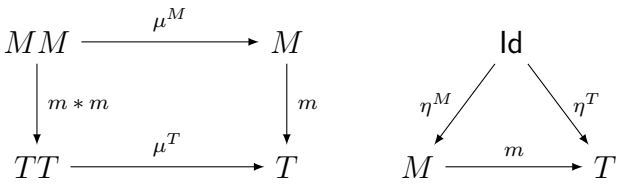
2.2.2 Monads

Definition 2.8. A *monad* on a category $\mathbb{C}$ is a triple $\langle M, \eta, \mu \rangle$, where $M : \mathbb{C} \to \mathbb{C}$ is an endofunctor, while $\eta : \mathsf{Id} \to M$ (called the *unit* of the monad) and $\mu : MM \to M$ (the *multiplication*) are natural transformations such that the following diagrams commute:



We call η and μ the *monadic structure* of M. Throughout this dissertation the natural transformations that form the monadic structure of a monad are always called η and μ, sometimes with superscripts to associate the natural transformations to the appropriate monad. With this convention, we usually refer to the monad simply by its endofunctor, as in the following definition:

Definition 2.9. A *morphism* between monads M and T is a natural transformation $m : M \to T$ such that the following diagrams commute:



By Mnd($\mathbb{C}$) we denote the category with monads on $\mathbb{C}$ as objects and monad morphisms as arrows. When the base category is obvious from the context, we write simply Mnd.

Example 2.10 (Monads as algebraic theories)**.** A great number of examples of monads come from the field of universal algebra. The simplest ones are the free algebras in equational varieties, like free monoids, groups, rings, and such. For example, the free monoid generated by a set X can be seen as a set of finite strings of elements from X, usually denoted X^* (the (-)* notation is also used in a more general construction of free monads in Section 2.2.7). The functor part of the free monoid monad on Set takes the set of generators to the set of strings (that is, $X \mapsto X^*$) with the obvious action on morphisms. The monadic structure is given by $\eta_X(x) = x$ (where the right-hand side denotes a singleton string) and $\mu_X((a_1 \dots a_n) \dots (z_1 \dots z_m)) = a_1 \dots a_n \dots z_1 \dots z_m$, that is, the concatenation of the string of strings.

In the general case, for a variety given by a signature Σ and a set of equations E, its free monad is given by the functor $X \mapsto \mathcal{T}_\Sigma(X)/{\approx_E}$ (that is, the quotient of the set of Σ-terms with variables from X by the equivalence induced by E) with the obvious action on morphisms. The unit is given by $\eta_X(x) = x$ (the embedding of variables), while the multiplication is defined on representatives of the equivalence classes as $\mu_X([t]_{\approx_E}) = [t\,\mathsf{id}]_{\approx_E}$, where the identity $\mathsf{id} : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(X)$ is understood as a substitution for $\mathcal{T}_\Sigma(X)$ treated as variables.

Example 2.11 (Monads in programming)**.** In programming language semantics, as proposed by Moggi [89], monads denote types of effectful computations. The multiplication amounts to sequential composition, while different kinds of monad represent different computational effects, like the writer monad ($X \mapsto O \times X$) for output, the exception monad ($X \mapsto X + E$, for an object of exceptions E), the state monad on cartesian closed categories ($X \mapsto (S \times X)^S$), or the list monad (aka the free monoid monad) to represent finitary nondeterminism. Later, these ideas were adopted by Wadler [119, 120] to structure functional programs. In this setting, monads become first-class citizens of the language: the underlying endofunctor is a data type, while the monadic structure amounts to two polymorphic functions.

2.2.3 Adjoint functors

Definition 2.12 (Kan [70])**.** Given two functors $F : \mathbb{C} \to \mathbb{D}$ and $U : \mathbb{D} \to \mathbb{C}$, we say that F is a *left adjoint* to U (or, equivalently, U is a *right adjoint* to F) if there exists a isomorphism $\varphi(\text{-})_{X,A} : \mathbb{D}[FX, A] \to \mathbb{C}[X, UA]$ natural in X and A. We denote the inverse of $\varphi(\text{-})$ as $\varphi^{-1}(\text{-})$.

By an innocuous abuse of notation, by $F \dashv G$ we mean both a pair of adjoint functors (that is, an *adjunction*, written also as $F \dashv G : \mathbb{C} \rightharpoonup \mathbb{D}$ to indicate the involved categories) and the fact that the two functors are mutually adjoint. The morphisms $\varphi(\text{-})$ and $\varphi^{-1}(\text{-})$ are called, respectively, left and right *adjuncts*. The morphisms of $\varphi(f)$ and $\varphi^{-1}(g)$ are called transpositions of f and g respectively.

Lemma 2.13. *A left adjoint to a functor is uniquely determined up to a natural isomorphism, that is, if $F \dashv U$ and $F' \dashv U$ then $F \cong F'$. The same holds for right adjoints.*

Definition 2.14. For an adjunction $F \dashv U : \mathbb{C} \rightharpoonup \mathbb{D}$, we define two natural transformations η (the *unit* of the adjunction) and ε (the *counit*) as transpositions of

identities:

$$\eta = \varphi(\mathsf{id}_F) : \mathsf{Id}_{\mathbb{C}} \to UF \qquad \varepsilon = \varphi^{-1}(\mathsf{id}_U) : FU \to \mathsf{Id}_{\mathbb{D}}$$

Lemma 2.15 (Huber [63])**.** *An adjunction $F \dashv U : \mathbb{C} \rightharpoonup \mathbb{D}$ induces a monad on $\mathbb{C}$ given as $\langle UF, \eta : \mathsf{Id} \to UF, U\varepsilon F : UFUF \to UF \rangle$.*

Example 2.16 (Monads as algebraic theories, continued)**.** Many examples of adjunctions $F \dashv U : \mathbb{C} \rightharpoonup \mathbb{D}$ arise when $\mathbb{D}$ is a category of some kind of algebras, while $\mathbb{C}$ is the category of their carriers or substructures (as in abelian group is a substructure of a ring). The functor U ('underlying') forgets about the algebraic structure and maps to the carrier, while F maps a set of generators to its free algebra. For a more concrete example, we can treat a variety of algebras $\mathbb{V}$ as a category with algebras as objects and appropriate homomorphisms as arrows. For instance, there is a category of monoids (Mon), a category of groups (Grp), and so on. We can define the forgetful functor $U : \mathbb{V} \to \textsc{Set}$ that maps an algebra to its carrier in the category of sets. In such a setting, U has a left adjoint and the composition of the two functors induces the free monad of the variety, already described in Example 2.10.

Adjunctions enjoy a great number of useful properties. The ones that are used in this dissertation include the following:

Lemma 2.17. *Let $F \dashv U$ be a pair of adjoint functors. Then:*

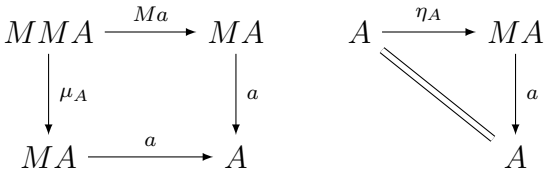
- *U is continuous (preserves limits) and F is cocontinuous (preserves colimits),*
- *$\varepsilon F \cdot F\eta = \mathsf{id}_F$ and $U\varepsilon \cdot \eta U = \mathsf{id}_U$ ('zig-zag' equalities),*
- *for $f : FX \to A$, it is the case that $\varphi(f)_{X,A} = Uf \cdot \eta_X$,*
- *for $g : X \to UA$, it is the case that $\varphi^{-1}(g)_{X,A} = \varepsilon_A \cdot Fg$.*

As stated in the next lemma, adjunctions compose. This turns out to be our ultimate tool to compose monads. It is rather 'low-level', so later on we present a couple of more abstract constructions: distributive laws and liftings.

Lemma 2.18. *For two adjunctions $F_1 \dashv U_1 : \mathbb{C} \rightharpoonup \mathbb{D}$ and $F_2 \dashv U_2 : \mathbb{D} \rightharpoonup \mathbb{E}$, it is the case that $F_2F_1 \dashv U_1U_2 : \mathbb{C} \rightharpoonup \mathbb{E}$.*

2.2.4 Eilenberg–Moore algebras and monadicity

Definition 2.19 ([35])**.** For a monad M, an *Eilenberg–Moore* (or: *monadic*) M-algebra is a pair $\langle A, a : MA \to A\rangle$ such that the following diagrams commute:



A homomorphism between two Eilenberg–Moore M-algebras is an ordinary homomorphism between M-algebras. By $\textsc{MAlg}(M)$ we denote the category of Eilenberg–Moore M-algebras as objects and homomorphisms as arrows. This category is called the *Eilenberg–Moore category* of M.

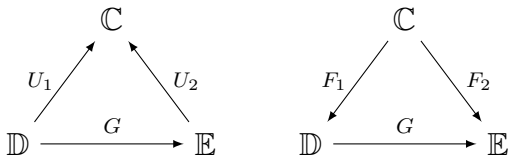
Lemma 2.20. *For a monad M on $\mathbb{C}$, we define a forgetful functor $U^{\mathrm{EM}} : \textsc{MAlg}(M) \to \mathbb{C}$ as $U^{\mathrm{EM}}\langle A, a\rangle = A$ on objects and $U^{\mathrm{EM}} f = f$ on morphisms. It has a left adjoint $F^{\mathrm{EM}} : \mathbb{C} \to \textsc{MAlg}(M)$ given as $F^{\mathrm{EM}} A = \langle MA, \mu_A : MMA \to MA\rangle$ on objects and $F^{\mathrm{EM}} f = Mf$ on morphisms.*

Definition 2.21. For an object A, we call the algebra $F^{\mathrm{EM}} A$ the *free* Eilenberg–Moore algebra generated by A.

Corollary 2.22. *One can verify that the monad induced by the adjunction $F^{\mathrm{EM}} \dashv U^{\mathrm{EM}} : \mathbb{C} \rightharpoonup \textsc{MAlg}(M)$ is equal to M. Hence, given an adjunction $F \dashv U : \mathbb{C} \rightharpoonup \mathbb{D}$ and a monad M on $\mathbb{D}$, the composition $UMF = UU^{\mathrm{EM}} F^{\mathrm{EM}} F$ is, by Lemmata 2.18 and 2.15, a monad.*

The corollary above states that every monad is induced by an adjunction. Moreover, a monad can be induced by more than one adjunction (another one is, for example, the Kleisli adjunction, see Section 2.2.6).

Definition 2.23. Adjunctions that give rise to a monad M on $\mathbb{C}$ form a category, which we call $\textsc{Adj}(M)$. A morphism between $F_1 \dashv U_1 : \mathbb{C} \rightharpoonup \mathbb{D}$ and $F_2 \dashv U_2 : \mathbb{C} \rightharpoonup \mathbb{E}$ is given by a functor $G : \mathbb{D} \to \mathbb{E}$ such that the following diagrams commute:



Lemma 2.24. *The Eilenberg–Moore adjunction* $F^{\mathrm{EM}} \dashv U^{\mathrm{EM}} : \mathbb{C} \rightharpoonup \textsc{MAlg}(M)$ *is the final object of* $\textsc{Adj}(M)$*. Given another adjunction* $F \dashv U : \mathbb{C} \rightharpoonup \mathbb{D}$*, we call the unique morphism* $K : \mathbb{D} \to \textsc{MAlg}(M)$ *in* $\textsc{Adj}(M)$ *the comparison functor.*

Let $\varepsilon : FU \to \mathsf{Id}_{\mathbb{D}}$ be the unit of $F \dashv U$. The comparison functor can be given as $KA = \langle UA,\, UFUA \xrightarrow{U\varepsilon} UA \rangle$ on objects and $Kf = Uf$ on morphisms.

The category of Eilenberg–Moore algebras fully describes the monad. This can be formalised as follows:

Definition 2.25. By EM we denote the category with objects givn by Eilenberg–Moore categories of monads on $\mathbb{C}$. Morphisms between $\textsc{MAlg}(M)$ and $\textsc{MAlg}(T)$ are given by functors $G : \textsc{MAlg}(M) \to \textsc{MAlg}(T)$ that preserve carriers, that is, ones for which the following diagram commutes:

$$\begin{array}{ccc} \textsc{MAlg}(M) & \xrightarrow{\;G\;} & \textsc{MAlg}(T) \\ & {\scriptstyle U^{\mathrm{EM}}} \searrow \quad \swarrow {\scriptstyle U^{\mathrm{EM}}} & \\ & \mathbb{C} & \end{array}$$

Lemma 2.26 ([25], Ch. 3, Theorem 6.3)**.** *Consider the functor* $\Delta : \textsc{Mnd} \to \mathrm{EM}^{\mathsf{op}}$ *defined as follows:*

$$\begin{aligned} &\Delta M = \textsc{MAlg}(M) \\ &(\Delta(k : T \to K))\langle A,\, a : KA \to A \rangle = \langle A,\, TA \xrightarrow{k} KA \xrightarrow{a} A \rangle \\ &(\Delta(k : T \to K))f = f \end{aligned}$$

It is an isomorphism with the inverse defined as:

$$\begin{aligned} &\Delta^{-1}(\textsc{MAlg}(M)) = M \\ &\Delta^{-1}(G : \textsc{MAlg}(T) \to \textsc{MAlg}(M)) = \left(M \xrightarrow{M\eta^T} MT \xrightarrow{p} T \right) \end{aligned}$$

where $G\langle TA,\ \mu^T_A : TTA \to TA \rangle = \langle TA,\ p_A : MTA \to TA \rangle$ *and* p_A *form a natural family of morphisms.*

Eilenberg–Moore algebras for a monad M can often be alternatively described by a different kind of objects, usually some kind of algebras whose additional structure or constraints account for the monadic structure of M. Such a situation is referred to as *monadicity*:

Definition 2.27. Given an adjunction $F \dashv U : \mathbb{C} \rightharpoonup \mathbb{D}$ that induces a monad M, we call it *monadic* if the comparison functor is an isomorphism.

Note that some authors call monadic functors 'strictly monadic', leaving the term 'monadic' to the broader case when the comparison functor K is an equivalence of categories.

Example 2.28 (Monads as algebraic theories, continued)**.** Let $\mathbb{V}$ be a variety of algebras understood as a category, as in Example 2.16. The free–underlying adjunction is monadic. In other words, $\mathbb{V} \cong \textsc{Malg}(M)$, where M denotes the free $\mathbb{V}$-algebra monad. For instance, the category of groups is isomorphic to the category of the Eilenberg–Moore algebras for the free group monad.

A convenient set of necessary and sufficient conditions for an adjunction to be monadic is given by (the strict version of) Beck's monadicity theorem [26] (see also Mac Lane [78, Ch. VI.7]). First, we need the following definitions:

Definition 2.29. A morphism $c : B \to C$ is a *split coequaliser* of two morphisms $h_0, h_1 : A \to B$ if there exist morphisms s and t such that the following diagram commutes and in which the two horizontal compositions are the identities:

$$\begin{array}{ccccc} B & \xrightarrow{t} & A & \xrightarrow{h_0} & B \\ \downarrow{\scriptstyle c} & & \downarrow{\scriptstyle h_1} & & \downarrow{\scriptstyle c} \\ C & \xrightarrow{s} & B & \xrightarrow{c} & C \end{array}$$

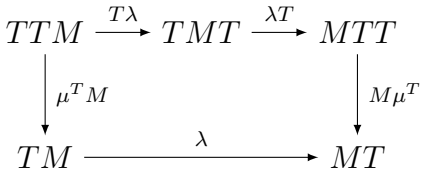
Definition 2.30. A functor $U : \mathbb{D} \to \mathbb{C}$ *creates coequalisers* for a pair of morphisms $h_0, h_1 : A \to B$ in $\mathbb{D}$ when for each coequaliser $u : UB \to X$ of Uh_0 and Uh_1 in $\mathbb{C}$, there exists a unique object C and a unique morphism $e : B \to C$ such that $UC = X$ and $Ue = u$, and e is the coequaliser of h_0 and h_1.

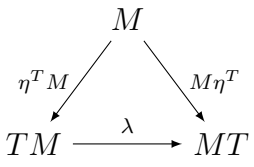
Theorem 2.31 (Beck's theorem)**.** *An adjunction $F \dashv U : \mathbb{C} \rightharpoonup \mathbb{D}$ is monadic if and only if U creates coequalisers for those pairs of morphisms $h_0, h_1 : A \to B$ in $\mathbb{D}$ for which Uf and Ug have a split coequaliser in $\mathbb{C}$.*

2.2.5 Distributive laws and liftings

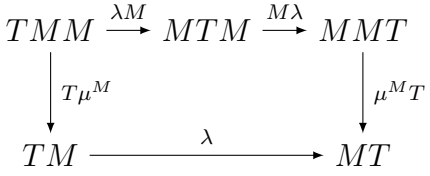
Definition 2.32 (Beck [27])**.** For two monads M and T on a category $\mathbb{C}$, a *distributive law* between M and T is a natural transformation $\lambda : TM \to MT$ such that the

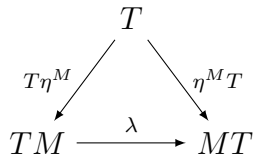
following diagrams commute:





(2.1)





(2.2)

If M is an endofunctor (not necessarily a monad) and the diagrams (2.1) commute, we call λ a *distributive law of a monad over an endofunctor*. Symmetrically, if T is an endofunctor (not necessarily a monad) and the diagrams (2.2) commute, we call λ a *distributive law of an endofunctor over a monad*.

A distributive law can be used to compose two monads:

Lemma 2.33. *A distributive law between monads $\lambda : TM \to MT$ induces a monad structure on the composition MT given by:*

$$\eta^{MT} = \left(\mathsf{Id} \xrightarrow{\eta^T} T \xrightarrow{\eta^M T} MT\right)$$
$$\mu^{MT} = \left(MTMT \xrightarrow{M\lambda T} MMTT \xrightarrow{\mu^M * \mu^T} MT\right)$$

Moreover, the natural transformations $\eta^M T : T \to MT$ and $M\eta^T : M \to MT$ are monad morphisms.

Example 2.34 (Monads as algebraic theories, continued)**.** The most obvious example of a distributive law in the field of algebra are rings. A ring can be seen as a monoid ('multiplication') and an abelian group ('addition') together with a distributivity of the former over the latter. Indeed, the free ring monad arises as a composition of the two simpler free monads via the obvious distributivity law.

Example 2.35 (Monads in programming, continued)**.** Composition of monads is useful when one wants to program with two different computational effects at once, for example, both mutable state and exceptions. Some monads come equipped with a distributive law over any other monad, for example, the exception monad $X \mapsto X + E$ distributes over any other monad via $[M\mathsf{inl}, \eta \cdot \mathsf{inr}] : M + E \to M(\mathsf{Id} + E)$. Such monads give rise to pointed endofunctors F in Mnd, that is, endofunctors equipped with a natural transformation $\mathsf{Id} \to F$ in Mnd. Such endofunctors model so-called *monad transformers* used in programming.

A (not so) different way of specifying interaction between two monads is given in terms of *liftings*. Liftings come in two kinds: Eilenberg–Moore and Kleisli. We first discuss the former:

Definition 2.36. Let M and T be monads on $\mathbb{C}$. We call a monad $\lceil M \rceil$ on $\textsc{Malg}(T)$ an *Eilenberg–Moore lifting* of the monad M if the following conditions are met:

$$\begin{array}{ccc} \textsc{Malg}(T) & \xrightarrow{\lceil M \rceil} & \textsc{Malg}(T) \\ \downarrow{\scriptstyle U^{\mathrm{EM}}} & & \downarrow{\scriptstyle U^{\mathrm{EM}}} \\ \mathbb{C} & \xrightarrow{M} & \mathbb{C} \end{array} \qquad \begin{aligned} U^{\mathrm{EM}}\eta^{\lceil M \rceil} &= \eta^M U^{\mathrm{EM}} \\ U^{\mathrm{EM}}\mu^{\lceil M \rceil} &= \mu^M U^{\mathrm{EM}} \end{aligned}$$

If M is an endofunctor (not necessarily a monad) and the diagram above commutes, we call $\lceil M \rceil$ an *Eilenberg–Moore lifting of an endofunctor*.

Note that $\lceil\text{-}\rceil$ is not a function from monads (or endofunctors) to liftings. A monad can have zero or more than one liftings.

Lemma 2.37. *Distributive laws of a monad T over a monad M are in 1-1 correspondence with liftings of M to* $\textsc{Malg}(T)$.

Given a distributive law $\lambda : TM \to MT$, the corresponding lifting is given as follows:

$$\begin{aligned} &\lceil M \rceil \langle A,\ a : TA \to A \rangle = \langle MA,\ TMA \xrightarrow{\lambda_A} MTA \xrightarrow{Ma} MA \rangle \\ &\lceil M \rceil k = Mk \\ &\eta^{\lceil M \rceil} = \eta^M \\ &\mu^{\lceil M \rceil} = \mu^M \end{aligned}$$

Remark 2.38. Given a distributive law $\lambda : TM \to MT$, the induced monad MT can be described as $MT = MU^{\mathrm{EM}}F^{\mathrm{EM}} = U^{\mathrm{EM}}(\Gamma\lambda)F^{\mathrm{EM}}$, which makes its monadic structure an instance of Corollary 2.22.

2.2.6 Kleisli categories

Definition 2.39 ([73])**.** For a monad M on a category $\mathbb{C}$, we define its *Kleisli category*, denoted $\textsc{Kleisli}(M)$, or simply $\textsc{Kleisli}$ when M is obvious from the context. It has the same objects as $\mathbb{C}$. For two objects A and B, an arrow $A \twoheadrightarrow B$ in $\textsc{Kleisli}$ is an arrow $f : A \to MB$ in $\mathbb{C}$. The identity is given by the $\mathbb{C}$-morphism $\eta^M_A : A \to MA$.

The composition of Kleisli-morphisms $f : A \nrightarrow B$ and $g : B \nrightarrow C$ is given as the following composition in $\mathbb{C}$:

$$A \xrightarrow{f} MB \xrightarrow{Mg} M^2B \xrightarrow{\mu^M_B} MB$$

Notation 2.40. Since $\mathbb{C}$ and Kleisli share the collection of objects, to avoid confusion, we use a different notation for morphisms in Kleisli, namely $\nrightarrow$.

Lemma 2.41 ([73])**.** *Define the forgetful functor* $U^{\mathrm{Kl}} : \textsc{Kleisli} \to \mathbb{C}$ *as:*

$$U^{\mathrm{Kl}}A = MA$$
$$U^{\mathrm{Kl}}(f : A \nrightarrow B) = \left(MA \xrightarrow{Mf} M^2B \xrightarrow{\mu^M_B} MB\right)$$

It has a left adjoint $F^{\mathrm{Kl}} : \mathbb{C} \to \textsc{Kleisli}$ *defined as:*

$$F^{\mathrm{Kl}}A = A$$
$$F^{\mathrm{Kl}}(f : A \to B) = \left(A \xrightarrow{f} B \xrightarrow{\eta^M_B} MB\right)$$

The adjunction $F^{\mathrm{Kl}} \dashv U^{\mathrm{Kl}}$ *induces* M*. In fact, it is the initial object in* $\textsc{Adj}(M)$*.*

Remark 2.42. If $\mathbb{C}$ has coproducts, the fact that $\textsc{Kleisli}(\mathbb{C})$ and $\mathbb{C}$ share the collection of objects enables us to define coproducts in Kleisli. Left adjoints are cocontinuous (Lemma 2.17), so the F^{Kl}-image of the coproduct diagram of $A + B$ is the coproduct diagram of $F^{\mathrm{Kl}}A = A$ and $F^{\mathrm{Kl}}B = B$.

Similarly to the liftings in Eilenberg–Moore categories, we can define *Kleisli liftings* (sometimes called *extensions*, see Manes and Mulry [79] for discussion):

Definition 2.43. Let M and T be monads on $\mathbb{C}$. We call a monad $\lfloor T \rfloor$ on $\textsc{Kleisli}(M)$ a *Kleisli lifting* of M if the following conditions are met:

$$\begin{array}{ccc} \textsc{Kleisli}(M) & \xrightarrow{\lfloor T \rfloor} & \textsc{Kleisli}(M) \\ {\scriptstyle F^{\mathrm{Kl}}}\uparrow & & \uparrow{\scriptstyle F^{\mathrm{Kl}}} \\ \mathbb{C} & \xrightarrow{\quad T \quad} & \mathbb{C} \end{array} \qquad \begin{aligned} \eta^{\lfloor T \rfloor} F^{\mathrm{Kl}} &= F^{\mathrm{Kl}} \eta^T \\ \mu^{\lfloor T \rfloor} F^{\mathrm{Kl}} &= F^{\mathrm{Kl}} \mu^T \end{aligned}$$

There exists an equivalent of the isomorphism Γ, which leads to the following corollary:

Lemma 2.44. *Distributive laws of a monad* T *over a monad* M *are in 1-1 correspondence with Kleisli liftings of* T *to* $\textsc{Kleisli}(M)$*.*

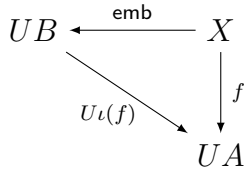
In detail, given a distributive law $\lambda : TM \to MT$, the Kleisli lifting $\lfloor T \rfloor$ is given as follows:

$$\begin{aligned}
&\lfloor T \rfloor A = TA \\
&\lfloor T \rfloor (f : A \to MB) = \left(TA \xrightarrow{Tf} TMB \xrightarrow{\lambda_B} MTB \right) \\
&\eta^{\lfloor T \rfloor} = \left(\mathsf{Id} \xrightarrow{\eta^T} T \xrightarrow{\eta^M} MT \right) \\
&\mu^{\lfloor T \rfloor} = \left(TT \xrightarrow{\mu^T} T \xrightarrow{\eta^M} MT \right)
\end{aligned}$$

In the other direction, concider the unit ε of the Kleisli adjunction $F^{\mathrm{Kl}} \dashv U^{\mathrm{Kl}}$. For an object A in $\textsc{Kleisli}(T)$, the morphism $\varepsilon_A : TA \to A$ in $\textsc{Kleisli}(T)$ is given by the morphism $\mathsf{id}_{TA} : TA \to TA$ in $\mathbb{C}$. Let $\lfloor S \rfloor : \textsc{Kleisli}(T) \to \textsc{Kleisli}(T)$ be a Kleisli lifting. The distributive law induced by S is given by the family of morphisms $\lambda_A = \lfloor S \rfloor \varepsilon_A : STA \to TSA$.

2.2.7 Free objects and free monads

Definition 2.45. Given a functor $U : \mathbb{D} \to \mathbb{C}$ and an object X of $\mathbb{C}$, a *free object* with respect to U generated by X is an object B of $\mathbb{D}$ together with a morphism $\mathsf{emb} : X \to UB$ in $\mathbb{C}$ such that for every object A in $\mathbb{D}$ and morphism $f : X \to UA$, there exists a unique morphism $\iota(f) : B \to A$ in $\mathbb{D}$ such that the following diagram commutes:



(2.3)

Lemma 2.46. *Let B together with* emb *form a free object as in the definition above. Let $f : X \to UA$ be a morphism. Then, the following equalities hold:*

- *cancellation:* $\iota(f) \cdot \mathsf{emb} = f$
- *reflection:* $\iota(\mathsf{emb}) = \mathsf{id}_{UB}$
- *fusion:* $Ug \cdot U\iota(f) = U\iota(Ug \cdot f)$ *for any morphism* $g : A \to C$.

Proof. Cancellation is the diagram (2.3). Reflection:

$$\mathsf{id} \cdot \mathsf{emb} = \mathsf{emb} = \iota(\mathsf{emb}) \cdot \mathsf{emb} \qquad \text{(identity, diagr. (2.3))}$$

And so reflection follows from the uniqueness of $\iota(\text{-})$ in the diagram (2.3). Fusion:

$$Ug \cdot U\iota(f) \cdot \mathsf{emb} = Ug \cdot f = U\iota(Ug \cdot f) \cdot \mathsf{emb} \qquad \text{(diagr. (2.3) } \times 2\text{)}$$

So fusion also follows from the uniqueness of $\iota(\text{-})$. □

An important example of a free object is the *algebraically free monad* generated by an endofunctor. It is a generalisation of the monad of terms for a signature Σ with the embedding of variables and the substitution as its monadic structure.

Definition 2.47. Let $\Sigma : \mathbb{C} \to \mathbb{C}$ be an endofunctor. If the initial $(\Sigma(\text{-}) + A)$-algebra exists for all A, the assignment $\Sigma^* A = \mu X.\Sigma X + A$ together with the obvious action on morphisms form an endofunctor. It can be given a monadic structure as follows:

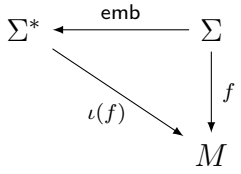
$$\eta^*_A = \left(A \xrightarrow{\mathsf{inr}} \Sigma\Sigma^* A + A \xrightarrow{\chi^{\Sigma(\text{-})+A}} \Sigma^* A\right)$$
$$\mu^*_A = (\!|f_A|\!), \text{ where}$$
$$f_A = \left(\Sigma\Sigma^* A + \Sigma^* A \xrightarrow{\mathsf{inl}+\mathsf{id}} \Sigma\Sigma^* A + A + \Sigma^* A \xrightarrow{[\chi^{\Sigma(\text{-})+A}, \mathsf{id}]} \Sigma^* A\right)$$

Theorem 2.48 (Barr [24])**.** *The monad Σ^* together with the natural transformation*

$$\mathsf{emb}_A = \left(\Sigma A \xrightarrow{\Sigma\eta^*_A} \Sigma\Sigma^* A \xrightarrow{\mathsf{inl}} \Sigma\Sigma^* A + A \xrightarrow{\chi^{\Sigma(\text{-})+A}} \Sigma^* A\right)$$

form the free object in Mnd *generated by* Σ*.*

To instantiate the definition of a free object, the above means that a natural transformation $f : \Sigma \to M$ uniquely extends to a coherent monad morphism $\iota(f) : \Sigma^* \to M$. In other words, $\iota(f)$ is a unique monad morphism that makes the following diagram commute:



(2.4)

We call Σ^* algebraically free, because it arises from an adjunction between the base category and the category of Σ-algebras:

Theorem 2.49. *If $\mu X.\Sigma X + A$ exists for all A, then the forgetful functor $U^{\mathrm{Alg}} : \mathrm{Alg}(\Sigma) \to \mathbb{C}$ has a left adjoint F^{Alg}, and Σ^* is the monad induced by $F^{\mathrm{Alg}} \dashv U^{\mathrm{Alg}}$. Moreover, this adjunction is monadic, which entails that $\mathrm{MAlg}(\Sigma^*) \cong \mathrm{Alg}(\Sigma)$.*

2.2.8 Cenciarelli and Moggi's resumption monad

We illustrate the results described in the previous sections with the example of Hyland, Plotkin, and Power's [65] construction of the monadic structure for Cenciarelli and Moggi's generalised resumption monad [28]:

Theorem 2.50. *Let M be a monad and Σ be an endofunctor such that $(\Sigma M)^*$ exists. The composition $M(\Sigma M)^*$ is a monad.*

Proof. We define a functor $\lceil M \rceil$ on $\textsc{Alg}(\Sigma M)$ as follows:

$$\lceil M \rceil \langle A, a : \Sigma MA \to A \rangle = \langle MA,\ \Sigma MMA \xrightarrow{\Sigma \mu^M_A} \Sigma MA \xrightarrow{a} A \xrightarrow{\eta^M_A} MA \rangle$$
$$\lceil M \rceil f = f$$

We furnish $\lceil M \rceil$ with a monadic structure induced by M, that is $\eta_{\langle A,a \rangle} = \eta^M_A$ and $\mu^M_{\langle A,a \rangle} = \mu^M_A$. Simple diagram chasing reveals that these are algebra homomorphisms, so $\lceil M \rceil$ is indeed a monad.

By Theorem 2.49, it is the case that $\textsc{Alg}(\Sigma M) \cong \textsc{MAlg}((\Sigma M)^*)$, so $\lceil M \rceil$ can be seen as a monad in $\textsc{MAlg}((\Sigma M)^*)$ and an Eilenberg–Moore lifting of M. Thus, Theorem 2.37 gives us a distributive law $\lambda : (\Sigma M)^* M \to M(\Sigma M)^*$, and so, via Theorem 2.33, a monadic structure on $M(\Sigma M)^*$. □

The direct definition of λ can be given by a fold $\lambda_A = (\!|f_A|\!)$, that is the unique homomorphism from the initial algebra, of the natural transformation f defined as:

$$f = \left(\Sigma MM(\Sigma M)^* + M \xrightarrow{\eta^M \Sigma \mu^M (\Sigma M)^* + \mathsf{id}} M\Sigma M(\Sigma M)^* + M \xrightarrow{[M\mathsf{emb}, M\eta^*]} M(\Sigma M)^* \right)$$

In the light of the rolling lemma (given below), we can see that Cenciarelli and Moggi's resumption monad can be given by carriers of initial algebras: $M(\Sigma M)^* A \cong \mu X.M(\Sigma X + A)$.

Lemma 2.51 (rolling lemma, see [23])**.** *For two endofunctors F and G, if μFG and μGF exist, then $\mu FG \cong F \mu GF$.*

Hyland, Plotkin, and Power [65] proved that Cenciarelli and Moggi's resumption monad is a coproduct:

Theorem 2.52. *The monad $M(\Sigma M)^*$ is a coproduct of M and Σ^* in* $\textsc{Mnd}$.

2.3 Monads and algebras with iteration

This section describes *completely iterative monads* (cims) introduced by Elgot [37] and recently studied by Adámek *et al.* [6]. Cims are monads furnished with a certain corecursion scheme. An example is the free cim generated by an endofunctor, which can be seen as a generalisation of the monad of infinite terms. It plays an important role in most of the constructions that we subsequently introduce in this dissertation. As a technical device, we also describe algebras with iteration: completely iterative algebras (cias) and complete Elgot algebras (Eilenberg–Moore algebras of the free cim).

Recall that we work in a background category $\mathbb{C}$ with finite coproducts. Whenever we use μF and νF as carriers of (co)initial (co)algebras, we implicitly assume they exist, but in general we do not assume that every endofunctor on $\mathbb{C}$ has a (co)initial (co)algebra. Some examples require additional structure on $\mathbb{C}$, like products or exponentials; in such cases we are explicit about it.

2.3.1 A coalgebraic approach to effectful programs

Coalgebra is one of the basic tools to handle corecursive programs, both in terms of presenting their semantics and in terms of structuring programs. In semantics, a G-coalgebra $\langle X,\ s : X \to GX\rangle$ in Set can be understood as a state transition system: X is the set of possible states, G is a functor that describes behaviours possible in one step of the computation, and s is the transition function. For example, G could represent a request–response behaviour, that is, $GA = (A \times O)^I$ for a set of possible requests (inputs) I and a set of possible responses (outputs) O. In such a case, the type of s becomes $X \to (X \times O)^I$, which is isomorphic to $X \times I \to X \times O$ (given a state of the system and a request, it produces a response paired with a new state of the system).

If G has a final coalgebra $\langle \nu G,\ \xi : \nu G \to G\nu G\rangle$, the carrier νG is the set of all possible traces of execution of the system. In the case of $GA = (A \times O)^I$, the set νG is a set of all possible infinite request–response interactions. The unique coalgebra homomorphism $[\![s]\!] : X \to \nu G$ is a function that executes the computation: given an initial state, it returns a trace of execution.

For a concrete example, we model a counter that awaits commands: inc to increment the counter and dec to decrement the counter. As a response, it gives the state of the counter after the operation. Let $\mathbb{Z}$ be the set of integers, $GA = (A \times \mathbb{Z})^{\{\mathsf{inc},\,\mathsf{dec}\}}$,

and $s : \mathbb{Z} \to G\mathbb{Z}$ be defined as follows:

$$s(n)(\mathsf{inc}) = \langle n+1,\, n+1 \rangle$$
$$s(n)(\mathsf{dec}) = \langle n-1,\, n-1 \rangle$$

In functional programming, similar constructions are used to handle infinite data (codata). An endofunctor G is a *shape functor* and the carrier of the final coalgebra $\langle \nu G,\ \xi : \nu G \to G\nu G \rangle$ is the data type obtained as the greatest fixed point of G. The action of an algebra $\langle X,\ s : X \to GX \rangle$ encodes one step of the unfolding of a value, and the unique morphism to the final coalgebra $[\![s]\!] : X \to \nu G$ is the unfolding.

For example, consider the following pseudo-Haskell program that produces a stream of Fibonacci numbers:

```
data Stream = Cons Int Stream

fibAux :: (Int, Int) -> Stream
fibAux (a, b) = Cons a fibAux (b, a+b)

fib :: Stream
fib = fibAux (1, 1)
```

The program above can easily be described in the coalgebraic setting: the `Stream` data type can be interpreted as the carrier of the final coalgebra of the shape functor `data StreamS x = ConsS Int x`, while `fibAux` is the unique homomorphism from the coalgebra, whose action is given as:

```
s :: (Int, Int) -> StreamS (Int, Int)
s (a, b) = ConsS a (b, a+b)
```

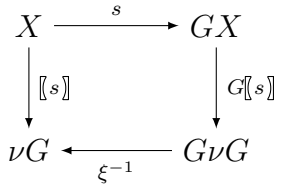
How about effectful programs? Consider the following implementation of the counter:

```
counter :: Int -> IO ()
counter n = do x <- readLn
               let y = case x of
                         "Inc" -> n + 1
                         "Dec" -> n - 1
               print y
               counter y
```

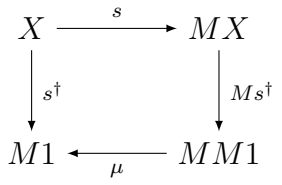
Clearly, this program is of a 'coalgebraic' shape. The type of the action of the coalgebra is `Int -> IO Int`.

Although the program above is given by a coalgebra of the shape $X \to MX$, we are not interested in the final coalgebra of M as a functor as our domain of interpretation, because it does not take into account the monadic structure of M.

In other words, in the non-effectful case, we are looking for a morphism $[\![s]\!]$ that solves the following diagram:



In the effectful case, we are interested in solutions $s^\dagger$ that solve the following diagram, where 1 stands for the terminal object (unit type):

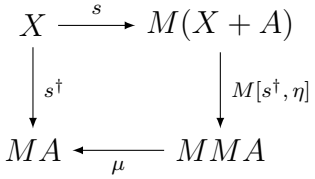


We need to slightly complicate the diagram above, because we need to take into account the fact that the monadic computation can return a value. For example, consider the following echo program, which, upon termination, return a report:

```
echo :: Int -> IO String
echo n = do x <- readLn
            case x of
              "stop" -> return ("Echoed " ++ show n ++ " times.")
              cs     -> putStrLn x >> echo (n+1)
```

Each step of this program either performs the next step by calling `echo (n+1)`, or returns a value `"Echoed..."`. Thus, its coalgebraic description is rather of the form $s : X \to M(X + A)$. Informally, if s returns a value in the left component (X), it means that the computation should continue, while returning the value in the right component (A) means that we have reached a final value. (Notice, though, that this intuition is too simplistic if M can return more than one value, as in the list monad.)

In the case of the program above, X is `Int`, and A is `String`. Thus, we are looking for solutions $s^\dagger$ to the following diagram:



Obviously, solutions $s^\dagger$ might not exist. For example, take M to be the state monad on Set for a set of states given by integers:

$$MA = (A \times \mathbb{Z})^{\mathbb{Z}}$$

Consider a function $s(x)(n) = \langle \mathsf{inl}\, x,\, n+1 \rangle$. Informally, s describes a computation that runs forever incrementing the state at each step. Clearly, it has no solutions.

We call a monad M *completely iterative* [6] if it has the property that there exist *unique* solutions to a certain morphisms $X \to M(X + A)$, called *guarded equation morphisms*, which we define in the next section.

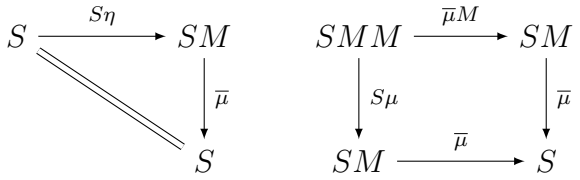
2.3.2 Guardedness, modules over monads, and idealised monads

While the notion of induction is based on well-foundedness, coinduction often needs a property called *guardedness*. Intuitively, it means that every step of a step-wise coinductive computation needs to contribute in some non-cancellative way to the result, so that every step brings some *progress* to the computation.

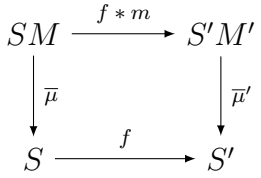
While it is fairly straightforward to capture the concept of guardedness or productiveness in the case of pure data structures (each step of the computation needs to create at least one constructor), it is far less obvious when dealing with monadic computations. A general solution was proposed Adámek *et al.* [12], who used ideals over monads to capture the subset of computations that allow iteration, for example, computations with observable behaviour. Note that modules over monads have more applications, which we discuss in Chapter 4.

Definition 2.53 ([33])**.** Let M be a monad. A pair $\langle S, \overline{\mu} \rangle$, where S is an endofunctor and $\overline{\mu} : SM \to S$ is a natural transformation (called an *action*), is an M*-module* (or:

a module over M) if the following diagrams commute:



We define a morphism between an M-module $\langle S, \overline{\mu}\rangle$ and an M'-module $\langle S', \overline{\mu}'\rangle$ as a pair of natural transformations $\langle m : M \to M',\ f : S \to S'\rangle$ such that m is a monad morphism and the following diagram commutes:



Abusing notation in the usual, we often denote a module $\langle S, \overline{\mu}\rangle$ simply as S.

Definition 2.54. We define the category MOD of modules. The objects are pairs $\langle M, S\rangle$, where M is a monad and S is an M-module. Morphisms are given by module morphisms, with the composition given component-wise.

Remark 2.55. Note that we can also define a *left module* of a monad M as an endofunctor L together with a natural transformation of the type $ML \to L$ such that the obvious analogues of the diagrams above commute (that is, a left module is an op-dual notion to a right module). The terms 'M-module' and 'module over monad' are used with a number of slightly different meanings in the literature. For example, some authors use them for Eilenberg–Moore algebras $a : MA \to A$. Indeed, Eilenberg–Moore algebras are exactly left modules for constant endofunctors. The definition that we address in this thesis is discussed, for exampe, by Mac Lane [78, Ch. VII.4] under the name actions (of monads on endofunctors). Modules can be further generalised, by not assuming that $\overline{M}$ or L are endo: they can be, respectively, functors to and from a category different than that of M. Such (left) modules were used by Street [110] to define so-called Eilenberg–Moore objects in 2-categories, and also by Hirschowitz and Maggesi [61, 62], and Ahrens [18] to model untyped λ-calculus and other reduction-based calculi.

Remark 2.56. A concise definition of a monad is that it is an internal monoid in the category of endofunctors and natural transformations, understood as a monoidal

category with the tensor given by the composition of endofunctors (see, for example, Mac Lane [78]). Thus, modules over monads are simply modules over internal monoids in the sense of monoidal categories.

Example 2.57. The following are examples of modules:

1. For a monad M, the pair $\langle M, \mu^M \rangle$ is an M-module.
2. Let $m : M \to T$ be a monad morphism. Then the pair $\langle T,\ \mu^T \cdot Tm \rangle$ is an M-module.
3. Let $\langle S, \overline{\mu}^M \rangle$ be an M-module and F be an endofunctor. Then, $\langle FS, F\overline{\mu}^M \rangle$ is an M-module.
4. With the definitions as above, let $\lambda : TM \to MT$ be a distributive law between monads. The pair $\langle ST,\ (\overline{\mu}^M * \mu^T) \cdot S\lambda T \rangle$ is a module of the induced monad MT.
5. If $\langle S, \overline{\mu} \rangle$ and $\langle S, \overline{\mu}' \rangle$ are two M-modules, the pair $\langle S + S',\ \overline{\mu} + \overline{\mu}' \rangle$ is also an M-module.
6. Let F be an endofunctor with a right adjoint U. Then, F is an UF-module with the action given by $\varepsilon F : FUF \to F$, where ε is the counit of the adjunction.

Modules often appear as subsets of computations with a certain closure property. For example, for a fixed integer $n \geq 1$, the functor of lists with at least n elements forms a module over the monad of non-empty lists. The closure property is the one defining modules: given a list with at least n elements, if we substitute a non-empty list for every element, we again get a list with at least n elements. Such a situation is an instance of an idealised monad:

Definition 2.58. An *idealised monad* is a triple $\langle M, \overline{M}, \sigma \rangle$ that consists of: a monad M, an M-module $\langle \overline{M}, \overline{\mu} \rangle$, and a natural transformation $\sigma : \overline{M} \to M$ such that $\langle \mathsf{id}, \sigma \rangle$ is a module morphism $\langle \overline{M}, \overline{\mu} \rangle \to \langle M, \mu \rangle$. We say that M is *idealised* with $\langle \overline{M}, \overline{\mu} \rangle$.

An *idealised monad morphism* between M and T, where T is idealised with $\langle \overline{T}, \overline{\mu}^T \rangle$, is a monad morphism $m : M \to T$ paired with a natural transformation $\overline{m} : \overline{M} \to \overline{T}$, such that the following diagrams commute:

$$\begin{array}{ccc} \overline{M}M & \xrightarrow{\overline{m} * m} & \overline{T}T \\ \downarrow{\scriptstyle \overline{\mu}^M} & & \downarrow{\scriptstyle \overline{\mu}^T} \\ \overline{M} & \xrightarrow{\overline{m}} & \overline{T} \end{array} \qquad \begin{array}{ccc} \overline{M} & \xrightarrow{\overline{m}} & \overline{T} \\ \downarrow{\scriptstyle \sigma^M} & & \downarrow{\scriptstyle \sigma^T} \\ M & \xrightarrow{m} & T \end{array}$$

If $M = \overline{M} + \mathsf{Id}$, we say that M is *ideal*. The category of idealised monads and idealised monad morphisms is called iMND.

Note that we use the notation with the bar over a variable that stands for a monad (as above, for example, $\overline{M}$) when we mean modules of idealised monads, as in Definition 2.58, while we use the variable S for modules that are not necessarily parts of the idealised monads (see Chapter 4). Usually, we refer to an idealised monad $\langle M, \overline{M}, \sigma \rangle$ simply as M.

Definition 2.59. For an endofunctor G and an idealised monad M, we say that a natural transformation $k : G \to M$ is *ideal* if it factors as $G \dashrightarrow \overline{M} \xrightarrow{\sigma^M} M$.

Example 2.60. Extending Example 2.57 (4) and (5), it is the case that:

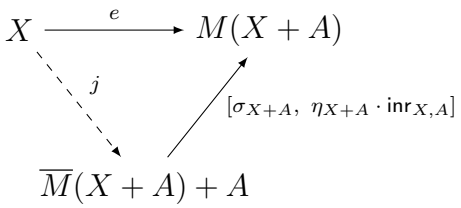
1. Let M be idealised with $\overline{M}$ and $\lambda : TM \to MT$ be a distributive law between monads. The induced monad MT is idealised with $\overline{M}T$. The associated module morphism is given by $\sigma^M T : \overline{M}T \to MT$.

2. Let M be idealised with $\overline{M}$ as well as with $\widetilde{M}$. Then, M is idealised with $\overline{M} + \widetilde{M}$. The associated module morphism is given by $[\sigma, \sigma'] : \overline{M} + \widetilde{M} \to M$, where σ and σ' are the respective associated morphisms of the two modules.

2.3.3 Completely iterative monads

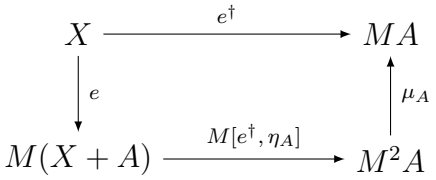
Recall that we assume that the base category $\mathbb{C}$ has finite coproducts. In this section, we assume that M is an idealised monad.

Definition 2.61. Let A and X be objects. A morphism $e : X \to M(X + A)$ is called a *guarded equation morphism* if it factors as follows:



We call X the object of *variables*, while A is called the object of *parameters*.

Definition 2.62. Let $e : X \to M(X + A)$ be a morphism. We call a morphism $e^\dagger : X \to MA$ a *solution* of e if the following diagram commutes:



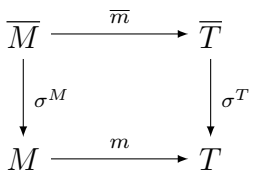
The monad M is *completely iterative* (is a *cim*) if every guarded equation morphism has a unique solution.

Example 2.63. For a signature Σ, the monad of possibly infinite Σ-terms on SET is completely iterative. In categorical terms, if for an endofunctor Σ, the final $(\Sigma(\text{-}) + A)$-coalgebras exist for all objects A, we define $\Sigma^\infty A = \nu X.\Sigma X + A$. It is a functor and a cim with respect to the module $\Sigma\Sigma^\infty$. It is a very important example of a cim—it is free, as discussed in the next subsection.

Example 2.64. The nomenclature of 'equations' and 'solutions' comes from the following example. Let Σ on SET be polynomial. We can view an equation morphism $e : X \to \Sigma^\infty(X + A)$ as a set of formal equations $\{x_i \approx t_i\}_{i \in I}$ for a (possibly infinite) set of indices I. The left-hand sides of the equations are variables from X, while the right-hand sides are (possibly infinite) terms with variables from the set X and 'constants' from the set A. A solution of such a set of equations is a substitution $e^\dagger : X \to \Sigma^\infty A$ with the property that it makes formal equations equalities, that is $x_i e^\dagger = t_i e^\dagger$ for all $i \in I$. This equality is exactly the diagram for solutions. One easily generalises this to any completely iterative monad M.

If M is ideal, the guardedness condition for e means that we do not consider sets of equations like $\{x_1 \approx x_1\}$ or $\{x_1 \approx x_2,\ x_2 \approx x_1\}$, which clearly do not have unique solutions, while M being idealised is a generalisation in which only certain operations in M guarantee that such solutions exist.

Definition 2.65. Let M and T be cims with respect to modules $\langle \overline{M}, \overline{\mu}^M \rangle$ and $\langle \overline{T}, \overline{\mu}^T \rangle$ respectively. We say that a monad morphism $m : M \to T$ is *coherent* if there exists a natural transformation $\overline{m} : \overline{M} \to \overline{T}$, such that the following diagram commutes:

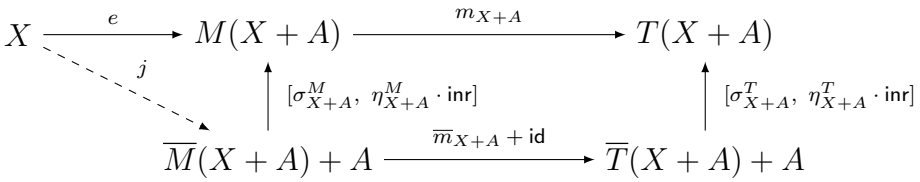


(It is the second condition for an idealised monad morphism in Definition 2.58.)

An important property of coherent monad morphism is that they preserve solutions:

Lemma 2.66 ([84])**.** *Let M, T, and m be as in the definition above. For a guarded equation morphism $e : X \to M(X + A)$, it is the case that $(m_{X+A} \cdot e)^\ddagger = m_A \cdot e^\dagger$, where $(\text{-})^\dagger$ and $(\text{-})^\ddagger$ denote solutions in M and T respectively.*

To be precise, we should note that in the lemma above the morphism $m_{X+A} \cdot e$ factors as follows:



(The triangle on the left is the guardedness diagram for e, while the square on the right follows from the definition of solution-preserving monad morphisms and properties of coproducts.) This means that $m_{X+A} \cdot e$ is guarded, so the solution $(m_{X+A} \cdot e)^\ddagger$ exists.

There are two ways in which we can define morphisms between cims. One preserves the entire structure of idealised monads, while in the other the only requirement is that they preserve solutions. Formally:

Definition 2.67 ([6])**.** The category with cims as objects and idealised monad morphisms as arrows is called iCim. The category with cims as objects and solution-preserving monad morphisms as arrows is called Cim.

Note that, quite obviously, every idealised monad morphism preserves solutions. Hence, the category iCim is a subcategory of Cim.

Remark 2.68. Adámek *et al.* use mostly the category iCim, while we concentrate on Cim. A conceptual reason for this is that the defining property of cims is the existence and uniqueness of solutions of guarded equation morphisms. The composition with a solution-preserving monad morphism preserves guardedness of equation morphisms, though not necessarily the 'way' a morphism is guarded (formally, the internal structure of the associated modules is not necessarily preserved), but this does not influence the solutions: we are not interested in what way an equation morphism factors through the module—we want the morphism j in Definition 2.61 *only* to exist; it is not a part of the data of a guarded equation morphism. A practical reason is that solution-preserving monad morphisms are better-behaved in some situations, as discussed in Section 5.5.

2.3.4 Completely iterative algebras and free cims

Definition 2.69 ([6])**.** Let H be an endofunctor. For two objects A and X, we call a morphism $e : X \to HX+A$ a *flat equation morphism*. We call a morphism $e^\dagger : X \to A$ a *solution* in an H-algebra $a : HA \to A$ if the following diagram commutes:

$$\begin{array}{ccc} X & \xrightarrow{e^\dagger} & A \\ \downarrow{\scriptstyle e} & & \uparrow{\scriptstyle [a,\mathsf{id}]} \\ HX + A & \xrightarrow{He^\dagger + \mathsf{id}} & HA + A \end{array}$$

The algebra $\langle A,\ a : HA \to A\rangle$ is called *completely iterative* (is a *cia*) if every flat equation morphism has a solution. By $\textsc{Cia}(H)$ we denote the full subcategory of $\textsc{Alg}(H)$ with completely iterative algebras as objects.

Definition 2.70 ([6])**.** A endofunctor H is called *iteratable* if the final coalgebra $\langle \nu X.HX + A,\ \xi^H_A\rangle$ exists for all A. We call the full subcategory of $\mathbb{C}^\mathbb{C}$ with iteratable endofunctors as objects $\textsc{Iter}$.

Theorem 2.71 ([6])**.** *Let H be an iteratable endofunctor. Then, the obvious forgetful functor $U^{\mathrm{CIA}} : \textsc{Cia}(H) \to \mathbb{C}$ has a left adjoint, which we call F^{CIA}. It is given on objects as $F^{\mathrm{CIA}}A = \langle \nu X.HX + A, \tau^H_A\rangle$, where τ^H is a natural transformation defined as:*

$$\tau^H_A = \left(H(\nu X.HX + A) \xrightarrow{\mathsf{inl}} H(\nu X.HX + A) + A \xrightarrow{(\xi^H_A)^{-1}} \nu X.HX + A\right)$$

Notation 2.72. We denote the monad induced by $F^{\mathrm{CIA}} \dashv U^{\mathrm{CIA}}$ as H^∞.

Note that $H^\infty A = \nu X.HX + A$ is similar to the free monad generated by H, but obtained with the greatest fixed point (ν) instead of the least one (μ). Thus, we can see it as a generalisation of possibly infinite H-terms. The monadic structure can be seen as embedding of variables and substitution. We denote it by η^∞ and μ^∞ (or η^{H^∞} and μ^{H^∞} when H is not obvious from the context). An important operation on H^∞ that we need is the following:

Lemma 2.73. *Given a natural transformation $f : H \to G$, consider the morphism $(fH^\infty + \mathsf{id}) \cdot \xi^H : H^\infty \to GH^\infty + \mathsf{Id}$. The morphism $[\![(fH^\infty + \mathsf{id}) \cdot \xi^H]\!] : H^\infty \to G^\infty$ is a monad morphism.*

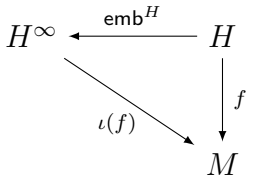
The fact that H^∞ is completely iterative was proven independently by Adámek *et al.* and (in a slightly different formulation) by Moss [91]:

Theorem 2.74. *The monad H^∞ is completely iterative with respect to the module $\langle HH^\infty, \tau^H \rangle$.*

Example 2.75. In a category with the initial object 0, the regular notion of coinduction can be retrieved from the complete iterativity of H^∞. Given a coalgebra $c : X \to HX$, it can be seen as a guarded equation morphism $\mathsf{emb} \cdot H\mathsf{inl} \cdot c : X \to H^\infty(X + 0)$, with the unique solution $(\mathsf{emb} \cdot H\mathsf{inl} \cdot c)^\dagger : X \to (H^\infty 0 \cong \nu H)$. In this case, the solution diagram instantiates to the coalgebra homomorphism diagram.

The monad H^∞ is the free object generated by H in the category of all cims:

Theorem 2.76 ([6])**.** *Let H be an iteratable endofunctor. Then, an ideal natural transformation $f : H \to M$ uniquely extends a monad morphism $\iota(f)$ that makes the following diagram commute in $\mathbb{C}^\mathbb{C}$:*



Moreover, the monad morphism $\iota(f)$ is coherent.

The morphism $\iota(f)$ can be defined directly as follows, where g is an auxiliary morphism. The guardedness of g follows from the fact that f is ideal.

$$g = \left(HH^\infty \xrightarrow{fH^\infty} MH^\infty \xrightarrow{M\xi} M(HH^\infty + \mathsf{Id}) \right)$$
$$g^\dagger : HH^\infty \to M$$
$$\iota(f) = \left(H^\infty \xrightarrow{\xi^H} HH^\infty + \mathsf{Id} \xrightarrow{[g^\dagger, \eta^M]} M \right)$$

Remark 2.77. For a natural transformation $f : H \to G$ the monad morphism $[\![(fH^\infty + \mathsf{id}) \cdot \xi^H]\!] : H^\infty \to G^\infty$ described in Lemma 2.73 can alternatively be given as $\iota(\mathsf{emb}^G \cdot f)$.

2.3.5 Elgot algebras

Another class of algebras with iteration are called Elgot algebras [15] introduced by Adámek *et al.* Elgot algebras are similar to cias, but instead of unique solutions, they come equipped with an assignment from equation morphisms to solutions subject to two axioms. In this dissertation they are used as a technical device: they correspond to Eilenberg–Moore algebras of free cims. First, we need the following:

Definition 2.78. For flat equation morphisms $e : X \to HX+Y$ and $f : Y \to HY+A$, and a morphism $h : Y \to Z$, we define two operations. *Renaming of parameters*:

$$h \circledcirc e = \Big(X \xrightarrow{e} HX + Y \xrightarrow{\mathsf{id}+h} HX + Z \Big)$$

Parallel composition:

$$f \boxdot e = \Big(X + Y \xrightarrow{[e,\mathsf{inr}]} HX + Y \xrightarrow{\mathsf{id}+f} HX + HY + A \xrightarrow{[H\mathsf{inl},H\mathsf{inr}]+\mathsf{id}} H(X+Y) + A \Big)$$

Definition 2.79 ([15])**.** For an endofunctor H, a *complete Elgot H-algebra* is a triple $\langle A, a : HA \to A, (\text{-})^\dagger \rangle$, where $(\text{-})^\dagger$ assigns to every flat equation morphism $e : X \to HX + A$ a solution $e^\dagger : X \to A$ such that the following two conditions hold:

- *Functoriality*: For two equation morphisms $e : X \to HX + A$ and $f : Y \to HY + A$ understood as $H(\text{-}) + A$ coalgebras, let $h : X \to Y$ be a coalgebra homomorphism, that is $f \cdot h = (Hh + \mathsf{id}_A) \cdot e$. Then, $e^\dagger = f^\dagger \cdot h$.
- *Compositionality:* For two equation morphisms $e : X \to HX + Y$ and $f : Y \to HY + A$ it is the case that $(f^\dagger \circledcirc e)^\dagger = (f \boxdot e)^\dagger \cdot \mathsf{inl}$.

Definition 2.80. For two complete Elgot H-algebras $\langle A, a, (\text{-})^\dagger \rangle$ and $\langle B, b, (\text{-})^\ddagger \rangle$, a morphism $h : A \to B$ is said to *preserve solutions* if for every flat equation morphism $e : X \to HX + A$ it is the case that $(h \circledcirc e)^\ddagger = h \cdot e^\dagger$. Complete Elgot H-algebras and solution-preserving morphisms form a category, which we denote as $\textsc{Elgot}(H)$.

Lemma 2.81 ([15])**.** *A solution-preserving morphism $h : \langle A, a, (\text{-})^\dagger \rangle \to \langle B, b, (\text{-})^\ddagger \rangle$ is a homomorphism of Σ-algebras $\langle A, a \rangle$ and $\langle B, b \rangle$.*

One can prove that completely iterative H-algebras enjoy functoriality and compositionality, hence they are also Elgot algebras. Moreover, completely iterative algebra homomorphisms preserve solutions. We denote the full embedding as $J^{\mathrm{Elg}} : \textsc{Cia}(H) \to \textsc{Elgot}(H)$.

Lemma 2.82 ([15])**.** *The obvious forgetful functor $U^{\mathrm{Elg}} : \textsc{Elgot}(H) \to \mathbb{C}$ has a left adjoint F^{Elg} given as the composition $F^{\mathrm{Elg}} = J^{\mathrm{Elg}} F^{\mathrm{CIA}}$, which means that the free cia is also the free Elgot algebra.*

Theorem 2.83 ([15])**.** *The adjunction $F^{\mathrm{Elg}} \dashv U^{\mathrm{Elg}}$ is monadic. This entails that there exists an isomorphism $\textsc{Elgot}(H) \cong \textsc{MAlg}(H^\infty)$.*

Chapter 3

Cims in semantics

The purpose of this chapter is twofold: First, we introduce a general construction of cims with adjunctions. Then, we use monads that result from this construction to give semantics to a generalised While language.

In detail, we introduce the *composition* theorem that addresses the composition of a monad with an adjunction (as described in Lemma 2.18). We show that if the inner monad is a cim (or even a weaker notion that we call a *relative cim*), then the monad that arises from the composition is completely iterative as well.

Second, we show how completely iterative and Elgot monads can be used in denotational semantics of programming languages. Our approach is axiomatic, which means that the (co)recursion in the modelled language is provided by a cim that is a parameter of the semantics. This way, we are able to study the semantics of (co)recursion without explicitly settling for any additional structure of the base category, like metric spaces or cpos.

Our running example is a generalised While language, parametrised with an underlying computational effect. In the ordinary While, the effect is mutable state, but other interesting and known constructions arise as different instances. We show that some known semantics for the ordinary While are concretisations of our generic semantics by an appropriate choice of the parametrising cim. Employing the composition theorem, we reconstruct two metric approaches: step-counting and trace semantics.

3.1 Composition theorem

Huber [63] noticed that given an adjunction $F \dashv U : \mathbb{C} \to \mathbb{D}$ and a monad M on $\mathbb{D}$, the composition UMF is also a monad (see Lemma 2.15). We prove that if additionally M is a cim, the resulting composition UMF is also a cim. In this

chapter, we instantiate this observation with the 'currying' adjunction $(\text{-}) \times Y \dashv (\text{-})^Y$ on two different categories: complete bounded ultrametric spaces and a cartesian closed category. In the former case we instantiate M with a 'delay' modality, which allows us to explicitly say that a stateful computation does not terminate. In the latter case, we instantiate M with a form of a stream that accumulate the intermediate states.

Theorem 3.1. *Let $\mathbb{C}$ and $\mathbb{D}$ be two categories with binary coproducts, $F \dashv U : \mathbb{C} \rightharpoonup \mathbb{D}$ be an adjunction with the associated natural isomorphism isomorphism φ, and T be a cim on $\mathbb{D}$ with respect to a module $\overline{T}$. Then, the induced monad UTF together with the module*

$$\langle U\overline{T},\ U\overline{T}FUTF \xrightarrow{U\overline{T}\varepsilon TF} U\overline{T}TF \xrightarrow{U\overline{\mu}^T F} U\overline{T}F\rangle$$

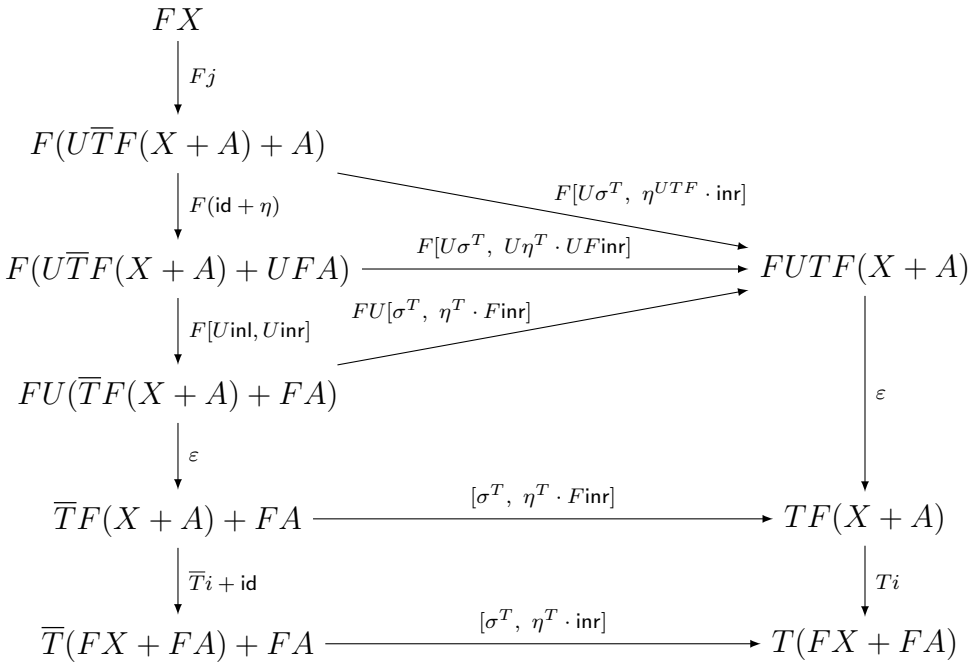
and the associated module morphism

$$\sigma^{UTF} = U\sigma^T F : U\overline{T}F \to UTF$$

is completely iterative with respect to $U\overline{T}F$. The solution of a guarded equation morphism $e : X \to UTF(X + A)$ is given by $\varphi((\varphi^{-1}e)^\dagger)$.

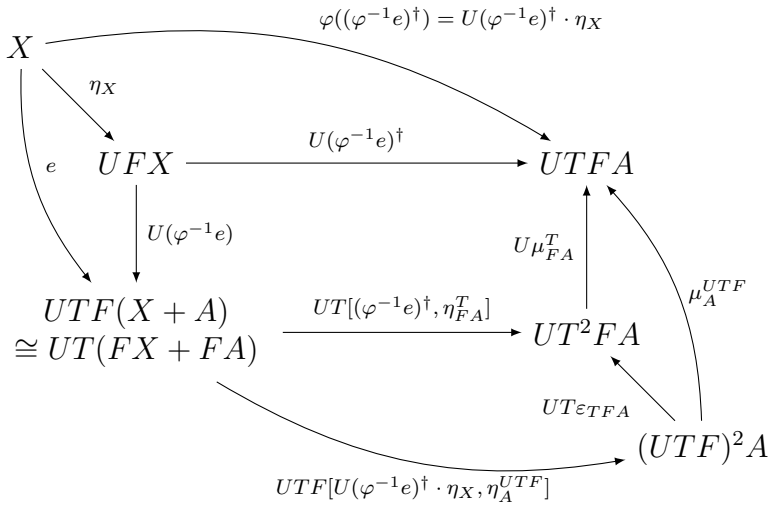
Proof. We need to show that: (i) $\varphi^{-1}e$ is a guarded equation morphism, hence $(\varphi^{-1}e)^\dagger$ exists, (ii) $\varphi((\varphi^{-1}e)^\dagger)$ is a solution in the monad UTF, and (iii) it is a unique solution.

(i) Left adjoints are cocontinuous, hence there exists a natural isomorphism $i_{A,B} : F(A+B) \to FA+FB$. Let $e : X \to UTF(X + A)$ be a guarded equation morphism in the monad UTF, that is, $e = [\sigma^{UTF}_{X+A},\ \eta^{UTF}_{X+A} \cdot \mathsf{inr}_{X+A}] \cdot j$ for a morphism $j : X \to U\overline{T}F(X + A) + A$. Consider the transposition of e, that is, the morphism $\varphi^{-1}(e) : FX \to TF(X + A)$, and the morphism $Ti \cdot \varphi^{-1}(e) : FX \to T(FX + FA)$. The following diagram commutes, which means that the latter morphism is a guarded equation morphism in T (the right-most path is equal to $Ti \cdot \varphi^{-1}(e)$ with $\varphi^{-1}(e)$ given via the counit, while the left–most path is an appropriate factorisation in T).



(The square at the bottom follows from the cocontinuity of F—more precisely, $i_{A,B}\cdot F\mathsf{inr}_{A,B}=\mathsf{inr}_{FA,FB}$. The rest is naturality.)

(ii) We show that $\varphi((\varphi^{-1}e)^\dagger)$ is a solution in the monad UTF. Since $Ti\cdot\varphi^{-1}e$ is guarded, the morphism $\varphi^{-1}e : FX\to T(FX+FA)$ has a unique solution $(\varphi^{-1}e)^\dagger : FX\to TFA$. We define the solution of e as $\varphi((\varphi^{-1}e)^\dagger)$. The following diagram commutes, which means that e has the solution property:



(The inner square is the U-image of the solution diagram for $(\varphi^{-1}e)^\dagger$. The outer triangles commute due to properties of adjunctions and the definition of μ^{UTF}.)

(iii) For uniqueness, let $g : X \to UTFA$ be a solution of e. Substitute $\varphi^{-1}g$ for $(\varphi^{-1}e)^\dagger$ in the above diagram. The outer square commutes, because $\varphi(\varphi^{-1}g) = g$ is a solution, and the triangles commute because of properties of adjunctions, hence the inner square precomposed with η_X also commutes. For all morphisms $f, f' : FY \to B$, if $Uf \cdot \eta_B = Uf' \cdot \eta_B$ then $f = f'$. Therefore, $\varphi^{-1}(g)$ is a solution of $\varphi^{-1}e$, so $\varphi^{-1}g = (\varphi^{-1}e)^\dagger$, hence $g = \varphi(\varphi^{-1}g) = \varphi((\varphi^{-1}e)^\dagger)$. □

Note that the assumption that $\mathbb{D}$ has all binary coproducts is rather strong. For example the category of Eilenberg-Moore algebras does not have all coproducts in general. On the other hand, in the proof of the theorem above, we exploit only solution morphisms in T of the shape $FX \to T(FX + FA)$, and the coproducts of the shape $FX + FA$ exist as F-images of $X + A$ (since F is cocontinuous). Thus, we can weaken our assumptions about $\mathbb{D}$ and T as follows:

Corollary 3.2. *Let $F \dashv U : \mathbb{C} \multimap \mathbb{D}$ be an adjunction. We call an idealised monad T on $\mathbb{D}$ an F-relative cim if every guarded equation morphism of the shape $FX \to T(FX+FA)$ has a unique solution. In such a setting, the monad UTF is a completely iterative with respect to the module $U\overline{T}F$.*

3.2 Generalised While

Now, we consider a generalised While language, similar to the one given by Rutten [105]. The syntax is parametrised with two signatures: C (intuitively, of statements or commands) and B (of boolean expressions), both consisting of (possibly infinitely many) nullary operations. For instance, in the ordinary While, the signature C has all possible assignment statements $x := e$, where x is a variable from a set X and e is an (arithmetic) expression that involves variables from X, and B is a set of boolean expressions that involve variables from X (they may have side-effects). The syntax of programs is given by the following grammar:

$$P \ ::= \ C \mid P; P \mid \textbf{if } B \textbf{ then } P \textbf{ else } P \mid \textbf{while } B \textbf{ do } P$$

An interpretation of the generalised While language consists of the following elements:

– a category $\mathbb{C}$ with countable coproducts and a terminal object 1,

– a cim M on $\mathbb{C}$,

– two *guards*, that is ideal natural transformations $\gamma^{\mathbf{if}}, \gamma^{\mathbf{while}} : \mathsf{Id}_{\mathbb{C}} \to M$,

– two interpretations of commands and boolean expressions, $[\![\text{-}]\!]_{\mathsf{C}} : C \to \mathbb{C}[1, M1]$ and $[\![\text{-}]\!]_{\mathsf{B}} : B \to \mathbb{C}[1, M(1+1)]$ respectively, where $\mathbb{C}[\text{-},\text{-}]$ is the hom-functor of $\mathbb{C}$.

The denotation of a program p is given by $[\![p]\!] : 1 \to M1$, defined as follows, where $\circ$ is the Kleisli composition:

$$\begin{aligned}
[\![c]\!] &= [\![c]\!]_{\mathsf{C}} \\
[\![p; q]\!] &= [\![q]\!] \circ [\![p]\!] \\
[\![\mathbf{if}\ b\ \mathbf{then}\ p\ \mathbf{else}\ q]\!] &= [[\![p]\!], [\![q]\!]] \circ [\![b]\!]_{\mathsf{B}} \circ \gamma_1^{\mathbf{if}} \\
[\![\mathbf{while}\ b\ \mathbf{do}\ p]\!] &= e^\dagger, \text{ where } e = ([\![p]\!] + \eta_1^M) \circ [\![b]\!]_{\mathsf{B}} \circ \gamma_1^{\mathbf{while}}
\end{aligned}$$

The denotation of a statement is given by the interpretation of the signature. The composition of two programs is the Kleisli composition of their respective denotations. The denotation of an **if** statement first performs the guard $\gamma^{\mathbf{if}}$, then b, and then the appropriate branch is chosen (we use the left component of $1 + 1$ to represent 'true'). The denotation of **while** first builds an equation morphism by composing the guard, the condition, and the choice between returning the right component of the coproduct (a constant, which means 'stop the iteration'), or performing the body, and left-injecting the result (which makes it a 'continue the iteration' variable).

The purpose of guards is to make sure that the morphism e is indeed guarded. We model loops as solutions to equation morphisms in a cim, so we need to make sure that the body of each **while** loop is guarded. For example, in the step-counting semantics, it is the guards that trigger the counting. That is why they are defined as ideal natural transformations. The 'if' guard can be used to specify an additional action upon entering an **if** statement, if one wants the equivalence of programs **while** b **do** p and **if** b **then** p; (**while** b **do** p) **else skip**.

The ordinary While is an imperative language, which means that commands (assignments) mutate global state. Usually, the state consists of a set of cells, but here, for brevity, we simply use an abstract set (object in general) S of all possible states. So, one could naively try to give the semantics to the language as follows. We set $\mathbb{C}$ to be Set, and M to be the state monad $\mathsf{St}\, A = (A \times S)^S$ for a set of states S (for now, we ignore the guards, that is, we assume $\gamma^{\mathbf{if}} = \gamma^{\mathbf{while}} = \eta^{\mathsf{St}}$). It is easy to calculate that the solution property of $e^\dagger$ instantiates to the following:

$$e^\dagger = \lambda(* \in 1).\lambda(s \in S).\begin{cases} e^\dagger(*)(s'') & \text{if } [\![b]\!]_{\mathsf{B}}(*)(s) = \langle \mathsf{inl}\, *, s' \rangle \text{ and } [\![p]\!]_{\mathsf{C}}(*)(s') = \langle *, s'' \rangle \\ \langle *, s' \rangle & \text{if } [\![b]\!]_{\mathsf{B}}(*)(s) = \langle \mathsf{inr}\, *, s' \rangle \end{cases}$$

It is (modulo (un)currying and the isomorphism $1 \times A \cong A$) the equation for the loop known from the domain-theoretic approach (see, for example, Nielson and Nielson's book [96]). Of course, it is not guaranteed to have a solution in Set. In the more domain-theoretic category Dcpo of pointed directed-complete posets, a solution exists, but it is not necessarily unique. Since we are interested in unique solutions, in the next section we discuss the metric approach, in which unique fixed points exist due to Banach's theorem.

3.3 The metric state monad and step-counting

In this section, we discuss a metric approach to semantics of While. We first show the usual way how a semantics for the ordinary While can be given using a step-counting monad on the category of ultrametric spaces. Then, we prove that this monad is a cim (by virtue of Theorem 3.1) and that the usual semantics is an instance of the proposed generic semantics.

Metric methods in semantics were initiated in the 1970s by Arnold and Nivat (see [21] for an overview) and later developed by Bakker and Zucker [32], and America and Rutten [19]. Here, we present an ultrametric approach used, for example, by Escardó [38] to model PCF, and Krishnaswami and Benton [74] to model functional reactive programming.

3.3.1 Metric semantics—the usual way

The results and notations presented in this subsection can be found in Escardó [38]. In the subsequent subsections we show that this semantics is an instance of our generic semantics.

Definition 3.3. A metric space $\langle A, d_A : A \times A \to \mathbb{R} \rangle$ is called an *ultrametric space* if it enjoys a stronger 'triangle inequality', namely $d_A(x, y) \leq \max\{d_A(x, z), d_A(z, y)\}$. It is *complete* if every Cauchy sequence has a limit in A. For a real $t > 1$, the metric space A is *t-bounded* if for all $x, y \in A$, $d_A(x, y) \leq t$.

For two metric spaces $\langle A, d_A \rangle$ and $\langle B, d_B \rangle$, we say that a function $f : A \to B$ is *non-expansive* if $d_B(f(x), f(y)) \leq d_A(x, y)$. Additionally, we call f *contractive* if there exists a non-negative $k < 1$ such that $d_B(f(x), f(y)) \leq k \cdot d_A(x, y)$.

Definition 3.4. By Cbums we denote the category of complete 1-bounded ultrametric spaces as objects and non-expansive functions as morphisms.

Theorem 3.5 (see [109]). *The category* Cbums *is cartesian closed and it is equipped with coproducts.*

In detail, the product of $\langle A, d_A\rangle$ and $\langle B, d_B\rangle$ is given by $\langle A \times B, d_{A\times B}\rangle$, where

$$d_{A\times B}(\langle x_1, y_1\rangle, \langle x_2, y_2\rangle) = \max\{d_A(x_1, x_2), d_B(y_1, y_2)\}.$$

The exponential object is equal to $\langle B^A, d_{A\Rightarrow B}\rangle$ where

$$d_{A\Rightarrow B}(f, g) = \sup\{d_B(f(x), g(x)) \mid x \in A\}.$$

The coproduct is given by $\langle A + B, d_{A+B}\rangle$, where

$$d_{A+B}(p, q) = \begin{cases} d_A(x_1, x_2) & \text{if } p = \mathsf{inl}\, x_1 \text{ and } q = \mathsf{inl}\, x_2 \\ d_B(y_1, y_2) & \text{if } p = \mathsf{inr}\, y_1 \text{ and } q = \mathsf{inr}\, y_2 \\ 1 & \text{otherwise} \end{cases}$$

Each set A can be lifted to Cbums using the discrete metric, that is, for all $x, y \in A$, the following holds:

$$\begin{aligned} d_A(x, x) &= 0 \\ d_A(x, y) &= 1 \qquad \text{if } x \neq y \end{aligned}$$

We use this lifting implicitly, that is, whenever a set A is mentioned as an object of Cbums without specifying the distance function, we mean the discrete lifting of A.

Lemma 3.6 (Banach's theorem). *A contractive function $f : A \to A$ on a non-empty complete metric space A has a unique fixed point denoted as $\mathsf{fix}(f)$.*

We interpret While programs in the following 'step-counting lifting' of metric spaces. The counting of steps necessary to complete the computation guarantees that appropriate functions are contractive.

Definition 3.7. We set a non-negative real $r < 1$. We define an endofunctor L : Cbums $\to$ Cbums. For an object $\langle A, d_A\rangle$ in Cbums, we define $LA = (A \times \mathbb{N}) \cup \{\infty\}$, and denote a pair $\langle x, n\rangle \in A \times \mathbb{N}$ as $x^{(n)}$. We define the distance function d_{LA} : $LA \times LA \to \mathbb{R}$ as follows:

$$\begin{aligned} d_{LA}(\infty, \infty) &= 0 \\ d_{LA}(x^{(k)}, \infty) &= d_{LA}(\infty, x^{(k)}) = r^k \\ d_{LA}(x^{(k)}, y^{(k)}) &= r^k \cdot d_A(x, y) \\ d_{LA}(x^{(k)}, y^{(t)}) &= r^{\min\{k,t\}} \qquad \text{if } k \neq t \end{aligned}$$

The morphism part is given as follows:

$$Lf(a^{(k)}) = f(a)^{(k)}$$
$$Lf(\infty) = \infty$$

Intuitively, two computations are closer together, if it takes more steps to distinguish between them. Note that two computations that result in the same value but obtained in different number of steps are considered different.

Definition 3.8. We define a natural transformation $\mathsf{delay}_A : LA \to LA$ as follows:

$$\mathsf{delay}_A(x^{(k)}) = x^{(k+1)}$$
$$\mathsf{delay}_A(\infty) = \infty$$

Lemma 3.9. *The endofunctor L is a monad with the monadic structure given as $\eta^L_A(x) = x^{(0)}$ and $\mu^L_A((x^{(k)})^{(t)}) = x^{(k+t)}$.*

We assume that the execution of a While program mutates the global state, represented as elements of a set $S = \{s, s_0, s_1, \ldots\}$. Given the interpretation of commands $[\![c]\!]'_{\mathsf{C}} : S \to S$ and boolean expressions $[\![b]\!]'_{\mathsf{B}} : S \to (1+1) \times S$, which return a modified state and a boolean value, we can define the semantics of While programs as $[\![p]\!]' : S \to LS$ as follows, where $\circ$ denotes the Kleisli composition of the monad L:

$$[\![c]\!]'(s) = \eta^L_S([\![c]\!]'_{\mathsf{C}}(s))$$
$$[\![p; q]\!]'(s) = ([\![q]\!]' \circ [\![p]\!]')(s)$$
$$[\![\textbf{if } b \textbf{ then } p \textbf{ else } q]\!]'(s_0) = \begin{cases} [\![p]\!]'(s_1) & \text{if } [\![b]\!]'_{\mathsf{B}}(s_0) = \langle \mathsf{inl}\, *,\ s_1 \rangle \\ [\![q]\!]'(s_1) & \text{if } [\![b]\!]'_{\mathsf{B}}(s_0) = \langle \mathsf{inr}\, *,\ s_1 \rangle \end{cases}$$
$$[\![\textbf{while } b \textbf{ do } p]\!]'(s) = \mathsf{fix}(\psi)(s),$$

where $\psi : (S \to LS) \to (S \to LS)$ is given as:

$$\psi(f)(s_0) = \mathsf{delay}_S \begin{cases} (f \circ [\![p]\!]')(s_1) & \text{if } [\![b]\!]'_{\mathsf{B}}(s_0) = \langle \mathsf{inl}\, *,\ s_1 \rangle \\ \eta^L_S(s_1) & \text{if } [\![b]\!]'_{\mathsf{B}}(s_0) = \langle \mathsf{inr}\, *,\ s_1 \rangle \end{cases}$$

One can prove that ψ is contractive, hence $\mathsf{fix}(\psi)$ is well-defined.

3.3.2 The cim-based semantics

Now, we show that the semantics above is an instance of the generic semantics given in the Section 3.2. First, we establish that L is a cim by means of existence of unique fixed points.

Theorem 3.10. *The monad L is completely iterative with respect to $\langle L, \mu^L \rangle$. The morphism $\sigma^L : L \to L$ is given by* delay.

Proof. It is easy to verify that L is idealised and that $\langle \mathsf{id}, \mathsf{delay} \rangle$ is a module morphism. Thus, we need to check that a guarded equation morphism $e : X \to L(X + A)$ has a unique solution. The fact that e is guarded means that it factors as follows:

$$X \xrightarrow{j} L(X + A) + A \xrightarrow{[\mathsf{delay}_{X+A},\ \eta^L_{X+A} \cdot \mathsf{inr}_{X,A}]} L(X + A)$$

We define the morphism $e^\dagger : X \to LA$ as the unique fixed point of the following function $\psi : (X \to LA) \to (X \to LA)$:

$$\begin{aligned} \psi(f) &= \mu^L_A \cdot L[f, \eta^L_A] \cdot e \\ &= \mu^L_A \cdot L[f, \eta^L_A] \cdot [\mathsf{delay}_{X+A},\ \eta^L_{X+A} \cdot \mathsf{inr}_{X,A}] \cdot j \end{aligned}$$

One can easily see that any fixed point of ψ is a solution (in the sense of cims), and that the uniqueness of such a fixed point gives us that e has a unique solution. We use Banach's theorem to achieve both.

By Banach's theorem, it is enough to show that ψ is contractive, that is, there exists a non-negative real $c < 1$ such that for two maps $f, f' : X \to LA$, the following holds:

$$d_{X \Rightarrow LA}(\psi(f), \psi(f')) \leq c \cdot d_{X \Rightarrow LA}(f, f') \tag{3.1}$$

The left-hand side of the equation (3.1) is equal to:

$$d_{X \Rightarrow LA}(\psi(f), \psi(f')) = \sup\{d_{LA}(\psi(f)(x), \psi(f')(x)) \mid x \in X\}$$

In turn, the right-hand side is as follows:

$$\begin{aligned} c \cdot d_{X \Rightarrow LA}(f, f') &= c \cdot \sup\{d_{LA}(f(x), f'(x)) \mid x \in X\} \\ &= \sup\{c \cdot d_{LA}(f(x), f'(x)) \mid x \in X\} \end{aligned}$$

Thus, it is enough to show that for all $x \in X$, there exists $y \in X$ such that:

$$d_{LA}(\psi(f)(x), \psi(f')(x)) \leq c \cdot d_{LA}(f(y), f'(y))$$

We show this for $c = r$. We analyse three cases:

Case 1. $j(x) = \mathsf{inr}\, a$, for some $a \in A$. It is easy to see that $\psi(f)(x) = \psi(f')(x) = \eta^L_A(a) = a^{(0)}$, so $d_{LA}(\psi(f)(x), \psi(f')(x)) = d_{LA}(a^{(0)}, a^{(0)}) = 0 \leq r \cdot d_{LA}(f(x), f'(x))$.

Case 2. $j(x) = \mathsf{inl}\,(\mathsf{inr}\,a)^{(k)}$, for some $a \in A$ and $k \in \mathbb{N}$. We calculate:

$$\begin{aligned}\psi(f)(x) &= \mu^L_A(L[f, \eta^L_A](\mathsf{delay}_{X+A}((\mathsf{inr}\,a)^{(k)})))\\ &= \mu^L_A(L[f, \eta^L_A]((\mathsf{inr}\,a)^{(k+1)}))\\ &= \mu^L_A(([f, \eta^L_A](\mathsf{inr}\,a))^{(k+1)})\\ &= \mu^L_A((\eta^L_A(a))^{(k+1)})\\ &= a^{(k+1)}\end{aligned}$$

Similarly, $\psi(f')(x) = a^{(k+1)}$, so $d_{LA}(\psi(f)(x), \psi(f')(x)) = d_{LA}(a^{(k+1)}, a^{(k+1)}) = 0 \leq r \cdot d_{LA}(f(x), f'(x))$.

Case 3. $j(x) = \mathsf{inl}\,(\mathsf{inl}\,y)^{(k)}$, for some $y \in X$ and $j \in \mathbb{N}$. We calculate:

$$\begin{aligned}\psi(f)(x) &= \mu^L_A(L[f, \eta^L_A](\mathsf{delay}_{X+A}((\mathsf{inl}\,y)^{(k)})))\\ &= \mu^L_A(L[f, \eta^L_A]((\mathsf{inl}\,y)^{(k+1)}))\\ &= \mu^L_A(([f, \eta^L_A](\mathsf{inl}\,y))^{(k+1)})\\ &= \mu^L_A((f(y))^{(k+1)})\end{aligned}$$

Let $f(y) = a^{(t)}$ and $f'(y) = b^{(h)}$, for some $t, h \in \mathbb{N}$ and $a, b \in A$. In such a case:

$$\begin{aligned}d_{LA}(\psi(f)(x), \psi(f')(x)) &= d_{LA}(\mu^L_A((f(y))^{(k+1)}), \mu^L_A((f(y))^{(k+1)}))\\ &= d_{LA}(\mu^L_A((a^{(t)})^{(k+1)}), \mu^L_A((b^{(h)})^{(k+1)}))\\ &= d_{LA}(a^{(k+1+t)}, b^{(k+1+h)})\\ &= r^{k+1} \cdot d_{LA}(a^{(t)}, b^{(h)})\\ &= r \cdot r^k \cdot d_{LA}(a^{(t)}, b^{(h)})\\ &\leq r \cdot d_{LA}(a^{(t)}, b^{(h)})\\ &= r \cdot d_{LA}(f(y), f'(y))\end{aligned}$$

□

Since CBUMS is cartesian closed, for the set S there is an adjunction $\text{-} \times S \dashv (\text{-})^S$, which induces the state monad $\mathsf{St} = (\text{-} \times S)^S$. The composition theorem (Theorem 3.1) immediately gives us the following:

Corollary 3.11 (the metric state monad)**.** *The monad* $\mathsf{MSt}\,X = (L(X \times S))^S$ *is completely iterative with respect to the module* $\langle \mathsf{MSt}, \mu^{\mathsf{MSt}} \rangle$ *and the associated morphism* $\sigma^{\mathsf{MSt}}_A = (\mathsf{delay}_{(A \times S)})^S$.

Using the obvious isomorphisms, one can express the interpretations from the metric semantics $[\![c]\!]'_{\mathsf{C}}$ and $[\![b]\!]'_{\mathsf{B}}$ as a computation in the state monad: $[\![c]\!]'_{\mathsf{C}} : 1 \to$

$(1 \times S)^S$ and $[\![b]\!]'_{\mathsf{B}} : 1 \to ((1+1) \times S)^S$. They easily lift to the monad MSt as $(\eta^L_{1\times S})^S \cdot [\![c]\!]'_{\mathsf{C}} : 1 \to (L(1 \times S))^S$ and $(\eta^L_{(1+1)\times S})^S \cdot [\![b]\!]'_{\mathsf{B}} : 1 \to (L((1+1) \times S))^S$. To complete an instance of the generic semantics, we need to make sure that the equation morphism in the denotation of the loop is guarded. To achieve this, we set $\gamma^{\mathbf{while}} = (\mathsf{delay}_{1\times S})^S \cdot \eta_1^{\mathsf{MSt}} : 1 \to \mathsf{MSt}\, 1$.

3.3.3 Equivalence of the two semantics

The two presented models—metric and cim-based—use the same building blocks: the L monad and state transforming functions. The difference is that the latter is more categorical, in the sense that it identifies state transformation as a monad and the whole semantic universe as a standard composition of two monads. In terms of the denotations, they are isomorphic:

Theorem 3.12. *The metric and cim-based models are equivalent, that is, for any program p, an initial state s_0, and a final state s_1, it holds that:*

$$[\![p]\!](*)(s_0) = \langle *, s_1\rangle^{(k)} \text{ if and only if } [\![p]\!]'(s_0) = s_1^{(k)}$$
$$[\![p]\!](*)(s_0) = \infty \text{ if and only if } [\![p]\!]'(s_0) = \infty$$

Proof. We proceed by induction over the structure of p. The only non-trivial case is $p =$ **while** b **do** q. In the following, $(\text{-})^\ddagger$ denotes the solution in MSt, while $(\text{-})^\dagger$ denotes the solution in L. We uses $\circ^M$ for the Kleisli composition of a monad M.

In the cim-based semantics, it holds that $[\![\mathbf{while}\ b\ \mathbf{do}\ q]\!] = e^\ddagger$, where $e = ([\![q]\!] + \eta_1^M) \circ^{\mathsf{MSt}} ((\eta^L_{(1+1)\times S})^S \cdot [\![b]\!]'_{\mathsf{B}}) \circ^{\mathsf{MSt}} (\mathsf{delay}_{1\times S})^S \circ^{\mathsf{MSt}} \eta_1^{\mathsf{MSt}}$. By the composition theorem (Theorem 3.1), $e^\ddagger = \varphi((\varphi^{-1}e)^\dagger)$, where the adjuncts $\varphi(\text{-})$ and $\varphi^{-1}(\text{-})$ of $\text{-} \times S \dashv (\text{-})^S$ are currying and uncurrying respectively. To clarify, the types are as follows:

$$\begin{aligned}
&e : 1 \to (L((1+1) \times S))^S \\
&\varphi^{-1}e : 1 \times S \to L((1+1) \times S) \quad \cong \quad 1 \times S \to L((1 \times S) + (1 \times S)) \\
&\varphi^{-1}e^\dagger : 1 \times S \to L(1 \times S) \quad \cong \quad S \to LS \\
&\varphi((\varphi^{-1}e)^\dagger) : 1 \to (L(1 \times S))^S
\end{aligned}$$

The morphism $(\varphi^{-1}e)^\dagger$ is given by $\mathsf{fix}(\psi)$, where $\psi : (1 \times S \to L(1 \times S)) \to (1 \times S \to L(1 \times S))$ is given by $\psi(f) = \mu^L_{1\times S} \cdot L[f, \eta^L_{1\times S}] \cdot \varphi^{-1}e$, that is, $\psi(f) = [f, \eta^L_{1\times S}] \circ^L \varphi^{-1}e$. Equivalently, we exploit the isomorphism $\mathsf{outr} : 1 \times S \cong S : \langle !_S, \mathsf{id}_S\rangle$ (where $!_A : A \to 1$ is the unique morphism to the terminal object), and define $(\varphi^{-1}e)^\dagger$ as $L\langle !_S, \mathsf{id}_S\rangle \cdot \mathsf{fix}(\chi) \cdot \mathsf{outr}$ for $\chi : (S \to LS) \to (S \to LS)$ given as follows:

$$\chi(f : S \to LS) = L\mathsf{outr} \cdot \psi(L\langle !_S, \mathsf{id}_S\rangle \cdot f \cdot \mathsf{outr}) \cdot \langle !_S, \mathsf{id}_S\rangle$$

By elementary calculation, one can show that $\chi = \psi$, so $\mathsf{fix}(\chi) = \mathsf{fix}(\psi)$.

Let $i : A \to B$ be an isomorphism between two ultrametric spaces. For a contractive function $f : B \to B$, it is easy to show that $i^{-1} \cdot f \cdot i : A \to A$ is contractive, and that $\mathsf{fix}(i^{-1} \cdot f \cdot i) = i^{-1}(\mathsf{fix}(f)) \in A$. We sum up:

$$\begin{aligned}
[\![\textbf{while } b \textbf{ do } q]\!](*)(s) &= e^{\ddagger}(*)(s) && \text{(def.)}\\
&= \varphi((\varphi^{-1}e)^{\dagger})(*)(s) && \text{(composition theorem (Theorem 3.1))}\\
&= (\varphi^{-1}e)^{\dagger}(*, s) && \text{(uncurrying)}\\
&= \mathsf{fix}(\psi)(*, s) && \text{(def.)}\\
&= (L\langle !_S, \mathsf{id}_S\rangle \cdot \mathsf{fix}(\chi))(s) && \text{(isomorphism)}\\
&= (L\langle !_S, \mathsf{id}_S\rangle \cdot \mathsf{fix}(\psi))(s) && \text{(the above)}\\
&= L\langle !_S, \mathsf{id}_S\rangle(\mathsf{fix}(\psi)(s)) && \text{(funct. composition)}\\
&= L\langle !_S, \mathsf{id}_S\rangle([\![\textbf{while } b \textbf{ do } q]\!]'(s)) && \text{(def.)}
\end{aligned}$$

□

3.4 The states monad and trace semantics

In this section, we give another variation of the state monad that arises from the composition theorem: the *states* monad. It is based on the existence of certain final coalgebras—free cims in particular—rather than any structure of the base category. We use it to instantiate the generic semantics for While to obtain a trace semantics: for an initial state, the meaning of a program is given by a (possibly infinite) stream of intermediate states.

We start with the following observation: Consider the category Set and a set S. The functor $\text{-} \times S$ is iteratable, which means that the free cim $(\text{-} \times S)^{\infty} = \nu X.(X \times S) + (\text{-})$ exists. We denote it as $\overrightarrow{S}A = (\text{-} \times S)^{\infty}A$. The instance $\overrightarrow{S}A$ can be regarded as a set of possibly infinite streams that consist of elements of S, possibly (if the stream is of finite length) terminated with a value $a \in A$. Traces of computations with final values from the set A are modelled as members of $\overrightarrow{S}A$.

Generalising this to any cartesian closed category with streams for an object S, we use the composition theorem to define the states monad:

Definition 3.13. In a cartesian closed category, let S be an object such that $\overrightarrow{S}$ exists. We define the *states monad* as the composition $\mathsf{Sts}\,A = (\overrightarrow{S}(A \times S))^S$.

One can easily see that it is similar to the metric state monad from the previous section with $\overrightarrow{S}$ substituted for L. Both are cims that 'upgrade' stateful computations with iteration.

Intuitively, we think of $\mathsf{Sts}\,A$ as a set of functions that take an initial state and return a stream of intermediate states S. If the stream is of finite length, it is always terminated with a final state paired with a final value. The final state is not necessarily a part of the trace, though. This means that the monadic composition does not preserve it by default. Consider the following example, in which we denote streams by juxtaposition:

$$\begin{aligned}&\mu_A^{\mathsf{Sts}}(\lambda(s \in S).\, s_0 \dots s_n s'(\lambda(t \in S).\, (t_0(t)) \dots (t_{k(t)}(t))\, (t'(t))\, a(s')))\\ &= \lambda(s \in S).\, s_0 \dots s_n (t_0(s')) \dots (t_{k(s')}(s'))\, (t'(s'))\, a(s')\end{aligned}$$

Though the value and number of values t depend on s' (the final state of the outer computation), s' does not appear in the stream after the composition. This means that to log the current state in the stream, we need to duplicate it. To capture this, we define the following operation dupl:

$$\mathsf{dupl}_A = \left(A \times S \xrightarrow{\langle \mathsf{id}, \mathsf{outr}\rangle} (A \times S) \times S \xrightarrow{\mathsf{emb}_{A\times S}} \overrightarrow{S}(A \times S)\right)$$

Thus, we can save the current state with the following operation:

$$\mathsf{save}_A = \left((\overrightarrow{S}(A \times S))^S \xrightarrow{(\overrightarrow{S}\mathsf{dupl})^S} (\overrightarrow{S}\,\overrightarrow{S}(A \times S))^S \xrightarrow{(\mu^{\overrightarrow{S}}_{A\times S})^S} (\overrightarrow{S}(A \times S))^S\right)$$

The composition theorem (Theorem 3.1) entails complete iterativity of Sts:

Corollary 3.14. *The monad* Sts *is completely iterative with respect to the module* $(S\overrightarrow{S}(A \times S))^S$ *together with the associated module morphism* $(\sigma^{\overrightarrow{S}}_{A\times S})^S : (S\overrightarrow{S}(A \times S))^S \to (\overrightarrow{S}(A \times S))^S$.

Chapter 4

Modules over monads

In this chapter, we present some new results regarding modules over monads (defined in Section 2.3.2). Modules play two important roles in this dissertation: they are used in the definition of idealised monads to capture the notion of the 'iterative core' of a cim, but they also provide general building blocks for other constructions—inductive and coinductive resumptions. Though this chapter does not deal with monads with iteration, it can be seen as a prelude to the next chapter, in which we transfer most of the theory presented here to the completely iterative setting.

First, we introduce the general theory of modules over monads, extending some results by Dubuc [33] (scaled down to the 1-categorical setting; see Mac Lane [78, Ch. VII, Sec. 4]). The results closely follow the basic theory of monads presented in Chapter 2. In detail:

- We lift the connection between monads and adjunctions to a connection between modules and adjunctions paired with a functor.
- We introduce distributive laws of monads over M-modules and M-modules over monads, together with liftings of modules to Eilenberg–Moore and Kleisli categories. Just as in the case of their monadic counterparts, the respective notions are in 1-1 correspondence.
- We construct a monadic structure on the composition MS^*, provided that S^* (that is, the free monad generated by S as an endofunctor) exists. The monadic structure is induced by a distributive law between monads $S^*M \to MS^*$.
- We introduce algebras for modules. For an M-module S, such an algebra consists of an Eilenberg–Moore algebra for M and an algebra for S as an endofunctor that satisfy a certain coherence condition. We prove that algebras for the

module S coincide with Eilenberg–Moore algebras for the monad MS^*, which makes algebras for modules a useful technical device.

– We prove that MS^* is freely generated by the M-module S.

From the perspective of programming and semantics, the monad MS^* freely generated by the module S can be seen as a generalisation of Moggi's inductive resumption monad $M(\Sigma M)^*$ for an endofunctor Σ. (The original monad is obviously obtained when $S = \Sigma M$.) The freeness entails a number of properties, like an elimination principle (a fold) coherent with the monadic structure of M, which subsumes some known results, for example the fact due to Hyland, Plotkin, and Power [65] (already stated in Theorem 2.52) that $M(\Sigma M)^*$ is the coproduct of M and Σ^* in the category of monads, or Filinski and Støvring's Σ-and-M-algebras [39] used to model effectful data types. We also show interesting instances of our monad in which $S \neq \Sigma M$.

Some results about modules are extended to the case of idealised monads. Such technical results are needed in the subsequent chapters to study the notion of guardedness for cims obtained via distributive laws.

4.1 Liftings and distributive laws

In this section, we show two types of distributivity: a monad over a module and a module over a monad. They turn out to correspond to obvious extensions of the notions of Eilenberg–Moore and Kleisli liftings.

4.1.1 Eilenberg–Moore liftings and distributive laws

Definition 4.1. A *distributive law* of a monad T over an M-module S is a pair $\langle \lambda : TM \to MT,\ \overline{\lambda} : TS \to ST \rangle$, where λ is a distributive law between monads and $\overline{\lambda}$ is a distributive law of a monad over an endofunctor, such that the following diagram commutes:

$$\begin{array}{ccccc} TSM & \xrightarrow{\overline{\lambda}M} & STM & \xrightarrow{S\lambda} & SMT \\ \downarrow{\scriptstyle T\overline{\mu}^S} & & & & \downarrow{\scriptstyle \overline{\mu}^S T} \\ TS & & \xrightarrow{\quad\overline{\lambda}\quad} & & ST \end{array} \tag{4.1}$$

Definition 4.2. Let T be a monad and S be a right M-module. An *Eilenberg–Moore lifting* of S as a module is a triple $\langle \lceil M \rceil, \lceil S \rceil, \overline{\mu}^{\lceil S \rceil} \rangle$, where $\lceil M \rceil : \textsc{Malg}(T) \to \textsc{Malg}(T)$ is an Eilenberg–Moore lifting of M, $\lceil S \rceil : \textsc{Malg}(T) \to \textsc{Malg}(T)$ is a

lifting of S as a functor, and $\overline{\mu}^{\lceil S \rceil} : \lceil S \rceil \lceil M \rceil \to \lceil S \rceil$ is a natural transformation such that $\langle \lceil S \rceil, \overline{\mu}^{\lceil S \rceil} \rangle$ is a right $\lceil M \rceil$-module and the following condition holds:

$$U^{\mathrm{EM}} \overline{\mu}^{\lceil S \rceil} = \overline{\mu}^{S} U^{\mathrm{EM}} \tag{4.2}$$

We need the following technical lemma in the proof of the next theorem:

Lemma 4.3. *Let T be a monad, and let G be an endofunctor with a lifting to* $\mathrm{MALG}(T)$ *denoted as $\lceil G \rceil$. Consider $\lceil G \rceil$-images of free Eilenberg–Moore algebras:*

$$\lceil G \rceil F^{\mathrm{EM}} A = \lceil G \rceil \langle TA, \mu^T_A \rangle = \langle GTA,\, p^G_A : TGTA \to GTA \rangle$$

The family of morphisms p^G_A is natural in A. Moreover, it is universal in the sense that for any Eilenberg–Moore algebra $\langle B,\, b : TB \to B \rangle$, the value of $\lceil G \rceil \langle B,\, b : TB \to B \rangle$ can be given in terms of p^G as follows:

$$\lceil G \rceil \langle B,\, b : TB \to B \rangle = \langle GB,\, TGB \xrightarrow{TG\eta^T_B} TGTB \xrightarrow{p^G_B} GTB \xrightarrow{Gb} GB \rangle$$

Proof. See Tanaka's PhD thesis [113, Ch. 3]. □

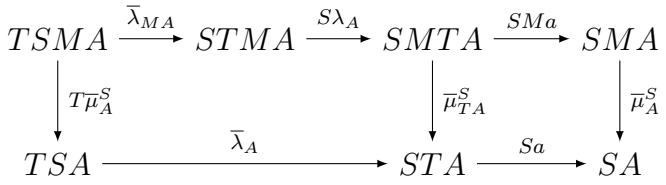
Theorem 4.4. *Distributive laws of a monad over an M-module and Eilenberg–Moore liftings of the module are in 1-1 correspondence.*

Proof. First, we notice that the condition (4.2) determines $\overline{\mu}^{\lceil S \rceil}$ uniquely. So, we can equivalently define a lifting of a modules as a pair $\langle \lceil M \rceil, \lceil S \rceil \rangle$ such that for all Eilenberg–Moore T-algebras $\langle A, a \rangle$, the morphism $\overline{\mu}^S_A$ is a homomorphism of the type $\lceil S \rceil \lceil M \rceil \langle A, a \rangle \to \lceil S \rceil \langle A, a \rangle$. Since distributive laws between monads are in 1-1 correspondence with Eilenberg–Moore liftings of monads, and the distributive laws of monads over endofunctors are in 1-1 correspondence with Eilenebrg–Moore lifting of functors (Lemma 2.37), it is left to prove that the additional conditions given in the definitions above also correspond.

Let $\langle \lambda : TM \to MT,\ \overline{\lambda} : TS \to ST \rangle$ be a distributive law of a monad T over an M-module S. By Lemma 2.37, λ corresponds to a lifting of a monad $\lceil M \rceil : \mathrm{MALG}(T) \to \mathrm{MALG}(T)$, while $\overline{\lambda}$ corresponds to a lifting of an endofunctor $\lceil S \rceil : \mathrm{MALG}(T) \to \mathrm{MALG}(T)$. We need to show that $\langle \lceil M \rceil, \lceil S \rceil \rangle$ is a lifting of a module, that is, that the following morphism $\overline{\mu}^{\lceil S \rceil} : \lceil S \rceil \lceil M \rceil \to \lceil S \rceil$ is indeed an algebra homomorphism:

$$\begin{aligned}
&\overline{\mu}^{\lceil S \rceil}_{\langle A,\, a : TA \to A \rangle} : \langle SMA,\ TSMA \xrightarrow{\overline{\lambda}_{MA}} STMA \xrightarrow{S\lambda_A} SMTA \xrightarrow{SMa} SMA \rangle \\
&\qquad \to \langle SA,\ TSA \xrightarrow{\overline{\lambda}_A} STA \xrightarrow{Sa} SA \rangle \\
&\overline{\mu}^{\lceil S \rceil} = \overline{\mu}^S
\end{aligned}$$

Let $\langle A, a : TA \to A\rangle$ be an Eilenberg–Moore T-algebra. The fact that $\overline{\mu}^{\lceil S\rceil}_{\langle A,\, a:TA\to A\rangle}$ is an algebra homomorphism can be read from the following diagram:



(The part on the left-hand side is the diagram (4.1), the square on the right-hand side is the naturality diagram of $\overline{\mu}^S$.)

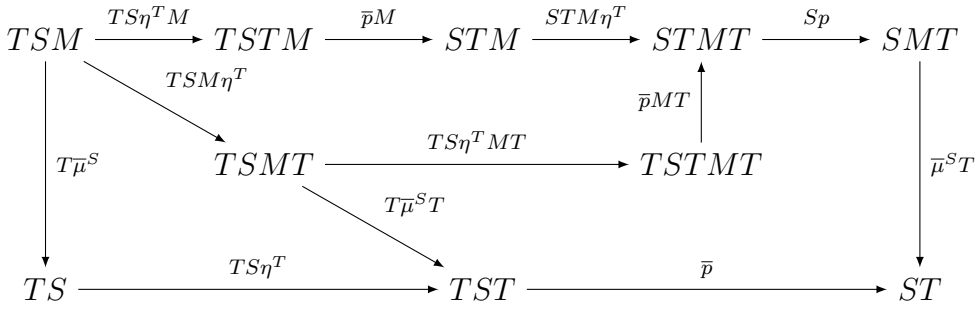
Conversely, let $\langle \lceil M\rceil, \lceil S\rceil, \overline{\mu}^{\lceil S\rceil}\rangle$ be a lifting of an M-module S to $\textsc{Malg}(T)$. By $\lambda : TM \to MT$ and $\overline{\lambda} : TS \to ST$ we denote the corresponding distributive laws (Lemma 2.37). We show that $\langle \lambda, \overline{\lambda}\rangle$ is a distributive law of T over S as a module, that is, that the diagram (4.1) from Definition 4.1 commutes. We unfold the definition of λ as given by Lemma 2.44:

$$\lambda = \left(TM \xrightarrow{TM\eta^T} TMT \xrightarrow{p} MT\right)$$

where $\lceil M\rceil\langle TA, \mu^T_A\rangle = \langle MTA,\ p_A : TMTA \to MTA\rangle$. Similarly:

$$\overline{\lambda} = \left(TS \xrightarrow{TS\eta^T} TST \xrightarrow{\overline{p}} ST\right)$$

where $\lceil S\rceil\langle TA, \mu^T_A\rangle = \langle STA,\ \overline{p}_A : TSTA \to STA\rangle$. We unfold these definitions in the diagram (4.1) and obtain the following diagram:



The only non-trivial part is the hexagon in the bottom-right corner. To see that it commutes, consider the type of $\overline{\mu}^{\lceil S\rceil}$ as given by the universality specified in

Lemma 4.3:

$$\begin{aligned}
&\overline{\mu}^{\lceil S\rceil}_{F^{\mathrm{EM}}A} : \lceil S\rceil\lceil M\rceil F^{\mathrm{EM}}A \\
&\quad = \lceil S\rceil\langle MTA, p_A\rangle \\
&\quad = \langle SMTA,\, TSMTA \xrightarrow{TS\eta^T_{MTA}} TSTMTA \xrightarrow{\overline{p}_{MTA}} STMTA \xrightarrow{Sp_A} SMTA\rangle \\
&\quad \to \lceil S\rceil F^{\mathrm{EM}}A \\
&\quad = \langle STA,\, TSTA \xrightarrow{\overline{p}_A} STA\rangle
\end{aligned}$$

The condition (4.2) states that $\overline{\mu}^{\lceil S\rceil} = \overline{\mu}^S$ as $\mathbb{C}$-morphisms, thus the bottom-right hexagon is the algebra homomorphism diagram for $\overline{\mu}^{\lceil S\rceil}_{F^{\mathrm{EM}}A}$, therefore it commutes. □

We extend the definitions above to the case of idealised monads, by requiring additional coherence with the module homomorphism σ.

Definition 4.5. Let T be a monad, $\langle M, \overline{M}, \sigma^M\rangle$ be an idealised monad, and $\langle \lambda : TM \to MT,\ \overline{\lambda} : T\overline{M} \to \overline{M}T\rangle$ be a distributive law of the monad T over the module $\overline{M}$. We say that $\langle\lambda, \overline{\lambda}\rangle$ is a *distributive law* of the monad T over the idealised monad $\langle M, \overline{M}, \sigma^M\rangle$ if it is the case that $\lambda \cdot T\sigma^M = \sigma^M T \cdot \overline{\lambda}$.

Definition 4.6. An *Eilenberg–Moore lifting* of an idealised monad $\langle M, \overline{M}, \sigma^M\rangle$ to $\mathrm{MAlg}(T)$ is a lifting $\langle\lceil M\rceil, \lceil\overline{M}\rceil, \overline{\mu}^{\lceil\overline{M}\rceil}\rangle$ of $\overline{M}$ as a module to $\mathrm{MAlg}(T)$ such that $\lceil M\rceil$ is idealised with $\lceil\overline{M}\rceil$ and the associated module morphism $\sigma^{\lceil M\rceil} : \lceil\overline{M}\rceil \to \lceil M\rceil$ satisfies the following condition:

$$U^{\mathrm{EM}}\sigma^{\lceil M\rceil} = \sigma^M U^{\mathrm{EM}} \tag{4.3}$$

Theorem 4.7. *The notions of distributive laws of monads over idealised monads and Eilenberg–Moore liftings of idealised monads are in 1-1 correspondence.*

Proof. First, we notice that $\sigma^{\lceil M\rceil}$, if it exists, is uniquely defined by its property $U^{\mathrm{EM}}\sigma^{\lceil M\rceil} = \sigma^M U^{\mathrm{EM}}$. Theorem 4.4 states that distributive laws of monads over M-modules and Eilenberg–Moore liftings of modules are in 1-1 correspondence, so it is enough to show that the additional conditions imply each other.

In one direction, assume that $\langle\lambda, \overline{\lambda}\rangle$ is a distributive law of a monad over an idealised monad. We need to prove that for every Eilenberg–Moore T-algebra $\langle A,\ a :$

$TA \to A\rangle$, the morphism σ^M_A is a homomorphism $\lceil\overline{M}\rceil\langle A, a\rangle \to \lceil M\rceil\langle A, a\rangle$ between the induced liftings. It can be read from the following diagram:

$$\begin{array}{ccccc} T\overline{M}A & \xrightarrow{\overline{\lambda}_A} & \overline{M}TA & \xrightarrow{\overline{M}a} & \overline{M}A \\ \downarrow{\scriptstyle T\sigma^M_A} & & \downarrow{\scriptstyle \sigma^M_{TA}} & & \downarrow{\scriptstyle \sigma^M_A} \\ TMA & \xrightarrow{\lambda_A} & MTA & \xrightarrow{Ma} & MA \end{array}$$

(The square on the left follows from the fact that $\langle\lambda, \overline{\lambda}\rangle$ is a distributive law of a monad over an idealised monad. The square on the right follows from the naturality of σ^M.)

In the other direction, let $\langle\lceil M\rceil, \lceil\overline{M}\rceil, \overline{\mu}^{\lceil M\rceil}\rangle$ be a lifting of an idealised monad to $\textsc{Malg}(T)$. We need to prove that the condition $\lambda \cdot T\sigma^M = \sigma^M T \cdot \overline{\lambda}$ holds for the induced distributive law $\langle\lambda, \overline{\lambda}\rangle$. We unfold the definitions of λ and $\overline{\lambda}$ as in the proof of Theorem 4.4:

$$\lambda = \left(TM \xrightarrow{TM\eta^T} TMT \xrightarrow{p} MT\right)$$

where $\lceil M\rceil\langle TA, \mu^T_A\rangle = \langle MTA,\ p_A : TMTA \to MTA\rangle$. Similarly:

$$\overline{\lambda} = \left(T\overline{M} \xrightarrow{T\overline{M}\eta^T} T\overline{M}T \xrightarrow{\overline{p}} \overline{M}T\right)$$

where $\lceil\overline{M}\rceil\langle TA, \mu^T_A\rangle = \langle\overline{M}TA,\ \overline{p}_A : T\overline{M}TA \to \overline{M}TA\rangle$. Therefore, it is enough to show that the following diagram commutes:

$$\begin{array}{ccccc} T\overline{M} & \xrightarrow{T\overline{M}\eta^T} & T\overline{M}T & \xrightarrow{\overline{p}} & \overline{M}T \\ \downarrow{\scriptstyle T\sigma^M} & & \downarrow{\scriptstyle T\sigma^M T} & & \downarrow{\scriptstyle \sigma^M T} \\ TM & \xrightarrow{TM\eta^T} & TMT & \xrightarrow{p} & MT \end{array} \tag{4.4}$$

The square on the left-hand side follows from the naturality of σ^M. To show that the square on the right-hand side commutes, we continue as follows.

The condition (4.3) states that the module morphism $\sigma^M : \overline{M} \to M$ is an algebra homomorphism $\lceil\overline{M}\rceil\langle T, a\rangle \to \lceil M\rceil\langle T, a\rangle$ for every Eilenberg–Moore T-algebra $\langle A,\ a : TA \to A\rangle$. Unfolding $\lceil\overline{M}\rceil$ and $\lceil M\rceil$ as in Lemma 4.3 yields that this means that the following diagram commutes for every $\langle A, a\rangle$ in $\textsc{Malg}(T)$:

$$\begin{array}{ccccccc} T\overline{M}A & \xrightarrow{T\overline{M}\eta^T_A} & T\overline{M}TA & \xrightarrow{\overline{p}_A} & \overline{M}TA & \xrightarrow{\overline{M}a} & \overline{M}A \\ \downarrow{\scriptstyle T\sigma^M_A} & & & & & & \downarrow{\scriptstyle \sigma^M_A} \\ TMA & \xrightarrow{TM\eta^T_A} & TMTA & \xrightarrow{p_A} & MTA & \xrightarrow{Ma} & MA \end{array} \tag{4.5}$$

By instantiating the diagram (4.5) with $\langle TA,\ \mu_A^T : TTA \to TA\rangle$ we obtain that the following diagram commutes:

$$\begin{array}{ccccccc} T\overline{M}TA & \xrightarrow{T\overline{M}\eta^T_{TA}} & T\overline{M}TTA & \xrightarrow{\overline{p}_{TA}} & \overline{M}TTA & \xrightarrow{\overline{M}\mu^T_A} & \overline{M}TA \\ \downarrow{\scriptstyle T\sigma^M_{TA}} & & & & & & \downarrow{\scriptstyle \sigma^M_{TA}} \\ TMTA & \xrightarrow{TM\eta^T_{TA}} & TMTTA & \xrightarrow{p_{TA}} & MTTA & \xrightarrow{M\mu^T_A} & MTA \end{array} \tag{4.6}$$

On the other hand, Lemma 4.3 gives us that:

$$\begin{aligned} \langle MTA,\ TMTA \xrightarrow{p_A} MTA\rangle &= \lceil M\rceil\langle TA, \mu_A^T\rangle \\ &= \langle MTA,\ TMTA \xrightarrow{TM\eta^T_{TA}} TMTTA \xrightarrow{p_{TA}} MTTA \xrightarrow{M\mu^T_A} MTA\rangle \end{aligned}$$

Similarly for $\overline{p}$:

$$\begin{aligned} \langle \overline{M}TA,\ T\overline{M}TA \xrightarrow{\overline{p}_A} \overline{M}TA\rangle &= \lceil \overline{M}\rceil\langle TA, \mu_A^T\rangle \\ &= \langle \overline{M}TA,\ T\overline{M}TA \xrightarrow{T\overline{M}\eta^T_{TA}} TMTTA \xrightarrow{\overline{p}_{TA}} \overline{M}TTA \xrightarrow{\overline{M}\mu^T_A} \overline{M}TA\rangle \end{aligned}$$

This means that the top edge of the diagram (4.6) is equal to $\overline{p}_A$, while the bottom edge is equal to p_A. In other words, the diagram (4.6) is the missing bit of the diagram (4.4), so the latter commutes. □

4.1.2 Kleisli liftings and distributive laws

In this section, we introduce the definition of a distributive law of an M-module over a monad. It mirrors the definition of a distributive law in the other direction (Definition 4.1). In the obvious way, we define a lifting of an M-module to the Kleisli category of a monad.

Definition 4.8. A *distributive law* of an M-module S over a monad T is a pair $\langle \lambda : MT \to TM,\ \overline{\lambda} : ST \to TS\rangle$, where λ is a distributive law between monads and $\overline{\lambda}$ is a distributive law of a functor over a monad, such that the following diagram commutes:

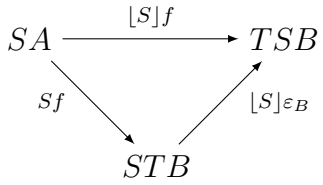
$$\begin{array}{ccccc} SMT & \xrightarrow{S\lambda} & STM & \xrightarrow{\overline{\lambda}M} & TSM \\ \downarrow{\scriptstyle \overline{\mu}^S T} & & & & \downarrow{\scriptstyle T\overline{\mu}^S} \\ ST & & \xrightarrow{\quad\overline{\lambda}\quad} & & TS \end{array} \tag{4.7}$$

Definition 4.9. Let T be a monad and S be a right M-module. A *Kleisli lifting* of S as a module is a triple $\langle \lfloor M \rfloor, \lfloor S \rfloor, \overline{\mu}^{\lfloor S \rfloor} \rangle$, where $\lfloor M \rfloor : \textsc{Kleisli}(T) \to \textsc{Kleisli}(T)$ is a Kleisli lifting of M as a monad, $\lfloor S \rfloor : \textsc{Kleisli}(T) \to \textsc{Kleisli}(T)$ is a Kleisli lifting of S as a functor, and $\overline{\mu}^{\lfloor S \rfloor} : \lfloor S \rfloor \lfloor M \rfloor \to \lfloor S \rfloor$ is a natural transformation such that $\langle \lfloor S \rfloor, \overline{\mu}^{\lfloor S \rfloor} \rangle$ is a $\lfloor M \rfloor$-module and the following condition holds:

$$\overline{\mu}^{\lfloor S \rfloor} F^{\mathrm{Kl}} = F^{\mathrm{Kl}} \overline{\mu}^{S} \tag{4.8}$$

We need an auxiliary lemma. The unit ε of the Kleisli adjunction $F^{\mathrm{Kl}} \dashv U^{\mathrm{Kl}}$ is universal in the sense that every lifting has the following factorisation:

Lemma 4.10. *For a monad $T : \mathbb{C} \to \mathbb{C}$, a Kleisli lifting $\lfloor S \rfloor : \textsc{Kleisli}(T) \to \textsc{Kleisli}(T)$, and a $\textsc{Kleisli}(T)$-morphism $f : A \to B$, the following diagram commutes in $\mathbb{C}$:*



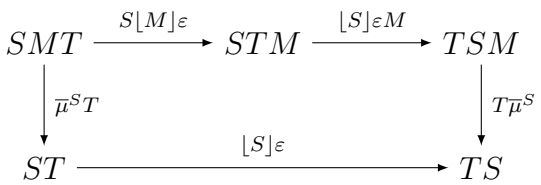
Proof. See Tanaka's PhD thesis [113, Lemma 4.8]. □

We now prove the main result about Kleisli liftings and distributive laws of M-modules over monads:

Theorem 4.11. *Distributive laws of an M-module over a monad and Kleisli liftings of the module are in 1-1 correspondence.*

Proof. We proceed similarly to the proof of Theorem 4.4. Distributive laws are in 1-1 correspondence with the appropriate liftings (Lemma 2.44), so we just need to check that the side conditions imply each other and that the natural transformation $\overline{\mu}^{\lfloor S \rfloor}$ is uniquely determined.

In one direction, let $\langle \lfloor M \rfloor, \lfloor S \rfloor, \overline{\mu}^{\lfloor S \rfloor} \rangle$ be a lifting of an M-module S to $\textsc{Kleisli}(T)$. We show that the diagram (4.7) commutes. Applying the definition of a distributive laws via the counit ε (Lemma 2.44), the diagram in question becomes as follows:



Due to Lemma 4.10, the horizontal path on top of the diagram is equal to $\lfloor S \rfloor \lfloor M \rfloor \varepsilon$. Thus, one can easily see that the diagram in question corresponds to the following diagram in $\textsc{Kleisli}(T)$, where the equalities are instances of the condition (4.8):

$$\begin{array}{ccc} SMT & \xrightarrow{\lfloor M \rfloor \lfloor S \rfloor \varepsilon} & SM \\ {\scriptstyle F^{\mathrm{Kl}}\overline{\mu}^S T = \overline{\mu}^{\lfloor S \rfloor T}}\downarrow & & \downarrow{\scriptstyle F^{\mathrm{Kl}}\overline{\mu}^S = \overline{\mu}^{\lfloor S \rfloor}} \\ ST & \xrightarrow{\lfloor S \rfloor \varepsilon} & S \end{array}$$

This diagram is simply the naturality condition for $\overline{\mu}^{\lfloor S \rfloor}$, hence it commutes.

In the other direction, let $\langle \lambda : MT \to TM,\ \overline{\lambda} : ST \to TS \rangle$ be a distributive law of S as a module over M. The functor F^{EM} is an identity on objects, so the condition (4.8) is actually a definition. Thus, the natural transformation $\overline{\mu}^{\lfloor S \rfloor}$ is uniquely determined as $F^{\mathrm{Kl}}\overline{\mu}^S = \eta^T S \cdot \mu^S$. □

We also introduce the notion of a distributive law of an idealised monad over a monad.

Definition 4.12. Let T be a monad, $\langle M, \overline{M}, \sigma^M \rangle$ be an idealised monad, and $\langle \lambda : MT \to TM,\ \overline{\lambda} : \overline{M}T \to T\overline{M} \rangle$ be a distributive law of the module $\overline{M}$ over T. We say that $\langle \lambda, \overline{\lambda} \rangle$ is a *distributive law* of the idealised monad $\langle M, \overline{M}, \sigma^M \rangle$ over a monad T if it is the case that $\lambda \cdot \sigma^M T = T\sigma^M \cdot \overline{\lambda}$.

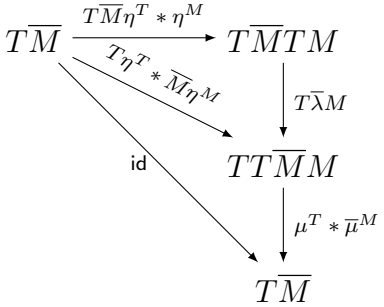
Such distributive laws are useful when we want to study idealised monads induced by distributive laws:

Theorem 4.13. *Let $\langle \lambda, \overline{\lambda} \rangle$ be a distributive law of an idealised monad $\langle M, \overline{M}, \sigma^M \rangle$ over a monad T. Then, the monad TM induced by λ is idealised with $T\overline{M}$.*

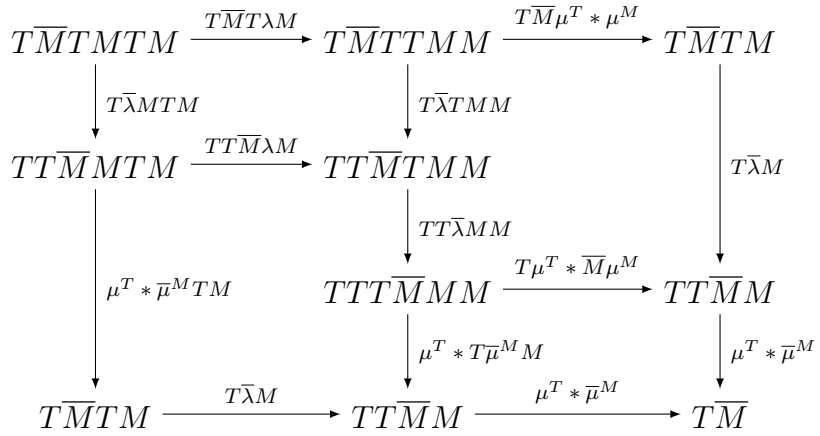
Proof. First, we show that $T\overline{M}$ together with the action

$$\overline{\mu}^{TM} = \left(T\overline{M}TM \xrightarrow{T\overline{\lambda}M} TT\overline{M}M \xrightarrow{\mu^T * \overline{\mu}^M} T\overline{M} \right)$$

is a module of TM. Indeed, the following diagrams commute. The unit law for modules:



(The triangle on top follows from the definition of distributive laws, the triangle in the bottom follows from monad laws for T and module laws for $\overline{M}$.) The multiplication law for modules:

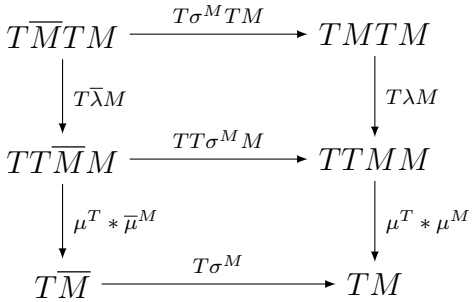


(Top-left corner: naturality of $\overline{\lambda}$, bottom-left corner: distributive law of modules over monads, top-right corner: distributive law of endofunctors over monads, bottom-right corner: monad laws for T and module laws for $\overline{M}$.)

We define the associated module morphism as follows:

$$\sigma^{TM} = \left(T\overline{M} \xrightarrow{T\sigma^M} TM\right)$$

We verify that it is a module morphism:



(Top square: distributive law of an idealised monad over a monad, bottom square: naturality of μ^T and the fact that σ^M is a module morphism.) □

4.2 Free monads generated by modules and their algebras

In the following, let M be a monad and S be a right M-module such that S^* (that is, the free monad generated by the endofunctor S) exists. In this section, we show that

the composition MS^* is a monad. Moreover, it is canonical in the sense that it is the free monad generated by S understood as a module. Then, we introduce *algebras for the module* S, which coincide with the Eilenberg–Moore algebras for the monad MS^*.

4.2.1 Monadic structure

We give the monadic structure of MS^* via a distributive law $\lambda : S^*M \to MS^*$. Our construction is an obvious adaptation of Hyland, Plotkin, and Power's proof [65] that the inductive resumptions $M(\Sigma M)^*$ form a monad (Theorem 2.50).

Theorem 4.14. *Let $\langle S, \overline{\mu}^S \rangle$ be an M-module. Consider the following natural transformation:*

$$\delta = \left(SM \xrightarrow{\overline{\mu}^S} S \xrightarrow{\eta^M S} MS \right)$$

It is a distributive law of the functor S over the monad M.

The following theorem immediately follows from the correspondence between distributive laws and liftings (compare the proof of Theorem 2.50):

Theorem 4.15. *The monad M has a lifting to $\textsc{Alg}(S)$ given as:*

$$\lceil M \rceil \langle A,\, a : SA \to A \rangle = \langle MA,\, SMA \xrightarrow{\delta_A} MSA \xrightarrow{Ma} A \rangle$$
$$\lceil M \rceil f = Mf$$

The monadic structure is inherited from M.

Theorem 4.16. *The composition MS^* is a monad.*

Proof. Since $\textsc{Alg}(S)$ is isomorphic to $\textsc{MAlg}(S^*)$, by the above theorem there exists a lifting of M to $\textsc{MAlg}(S^*)$. It induces a distributive law $\lambda : S^*M \to MS^*$, which gives us the monadic structure on MS^*. □

Remark 4.17. Using the definitions of the isomorphisms and with some calculation, we can read the direct definition of the monadic structure in terms of a fold, that is the unique algebra homomorphisms from the initial $(S(\text{-}) + MA)$-algebra to the algebra

$$\langle MS^*A,\, [\mu^M_{S^*A} \cdot \beta^S_A \cdot \overline{\mu}^S_{S^*A},\, M\eta^*_A] : SMS^*A + MA \to MS^*A \rangle,$$

where $\beta^S_A : SS^*A \to S^*A$ is the action of the free S-algebra generated by the object A. That is,

$$\lambda_A = (\!| \mu^M_{S^*A} \cdot \beta^S_A \cdot \overline{\mu}^S_{S^*A},\, M\eta^*_A |\!) : S^*MA \to MS^*A$$

For a more detailed account, see Section A.2 in the appendix.

The monad MS^* can be seen as a generalisation of Moggi's resumption monad $M(\Sigma M)^*A \cong \mu X.M(\Sigma X + A)$ for an endofunctor Σ. Thus, in this dissertation, we call it *the inductive resumption monad*, opposed to the *coinductive* resumption monad MS^∞ discussed in the next chapter. Moggi's monad arises for the special case $S = \Sigma M$—it follows from Example 2.57 that ΣM is an M-module. A distinctive feature of this generalisation is that in general MS^* is not given by the family of carriers of initial algebras, like in the following example:

Example 4.18. In a cartesian closed category, we can define a version of the state monad that keeps track of the intermediate states. It is similar but not identical to 'states' given in Definition 3.13 (compare also Ahman and Uustalu's update monads [17]). Fix an object of states A, and consider the reader monad $\mathsf{R}X = X^A$. The writer $\mathsf{W}X = X \times A$ is an R-module. The action can be given explicitly as $\langle \mathsf{ev}^{\mathsf{A}}_{\mathsf{X}}, \mathsf{outr} \rangle : \mathsf{WR}X = X^A \times A \to X \times A = \mathsf{W}X$, where ev is the evaluation morphism of the exponential object, outr is the right projection, and $\langle \text{-}, \text{-} \rangle$ is the product mediator. Intuitively, for an initial state, the monad $\mathsf{RW}^*X = ((\text{-} \times A)^*X)^A$ produces a (finite) sequence of intermediate states W^*. The sequence is terminated with a final value X.

4.2.2 Algebras for modules

Definition 4.19. An algebra for an M-module S is a triple $\langle A, f : MA \to A, g : SA \to A \rangle$ such that the following conditions hold:

1. The morphism f is an Eilenberg–Moore M-algebra.
2. The morphism g is an S-algebra.
3. Coherence: the following diagram commutes:

$$\begin{array}{ccc} SMA & \xrightarrow{Sf} & SA \\ \downarrow{\scriptstyle \overline{\mu}^S_A} & & \downarrow{\scriptstyle g} \\ SA & \xrightarrow{g} & A \end{array}$$

A morphism between two algebras for modules is a pair of appropriate algebra homomorphisms. We denote the category of algebras for an M-module S as $\textsc{ModAlg}(M, S)$.

Theorem 4.20. *Let S be an M-module. If S^* exists, the obvious forgetful functor $U^{\mathrm{ModAlg}} : \textsc{ModAlg}(M, S) \to \mathbb{C}$ has a left adjoint given by:*

$$\begin{aligned} F^{\mathrm{ModAlg}}A &= \langle MS^*A,\ f,\ g\rangle, \text{ where} \\ f &= MMS^*A \xrightarrow{\mu^M_{S^*A}} MS^*A \\ g &= SMS^*A \xrightarrow{\overline{\mu}^S_{S^*A}} SS^*A \xrightarrow{\beta^S_A} S^*A \xrightarrow{\eta^M_{S^*A}} MS^*A \\ F^{\mathrm{ModAlg}}k &= MS^*k \end{aligned}$$

The monad induced by this adjunction is equal to MS^*.

Proof. We show that the proposed adjunction is in fact a composition (see Lemma 2.18) of two simpler adjunctions. First, consider the adjunction $F^{\mathrm{Alg}} \dashv U^{\mathrm{Alg}} : \mathbb{C} \rightharpoonup \textsc{Alg}(S)$. The lifting $\lceil M \rceil$ from Theorem 4.15 can be seen as a monad on $\textsc{Alg}(S)$. It gives rise to the Eilenberg–Moore adjunction $F^{\mathrm{EM}} \dashv U^{\mathrm{EM}} : \textsc{Alg}(S) \rightharpoonup \textsc{MAlg}(\lceil M \rceil)$. This gives us the composite adjunction $F^{\mathrm{EM}}F^{\mathrm{Alg}} \dashv U^{\mathrm{Alg}}U^{\mathrm{EM}} : \mathbb{C} \rightharpoonup \textsc{MAlg}(\lceil M \rceil)$. We show that the categories $\textsc{MAlg}(\lceil M \rceil)$ and $\textsc{ModAlg}(M, S)$ are isomorphic and that $F^{\mathrm{ModAlg}} = F^{\mathrm{EM}}F^{\mathrm{Alg}}$ and $U^{\mathrm{ModAlg}} = U^{\mathrm{Alg}}U^{\mathrm{EM}}$ (modulo the isomorphism of categories).

To show the isomorphism of categories, we notice that the algebras in $\textsc{MAlg}(\lceil M \rceil)$ are of the following shape:

$$\langle \langle A, g : SA \to A\rangle,\ f : \lceil M \rceil \langle A, g\rangle \to \langle A, g\rangle \rangle$$

Unfolding the definitions of Eilenberg–Moore algebras and $\lceil M \rceil$, we obtain that the pair above is an Eilenberg–Moore algebra of $\lceil M \rceil$ if and only if the following conditions are met:

- The morphism g is an S-algebra (obviously).
- Since $\lceil M \rceil$ inherits its monadic structure from M, the morphism $f : MA \to A$ understood as a $\mathbb{C}$-morphism has the Eilenberg–Moore property for M.
- The morphism f is an algebra homomorphism between $\lceil M \rceil \langle A, g\rangle = \langle MA,\ SMA \xrightarrow{\overline{\mu}^S_A} SA \xrightarrow{g} A \xrightarrow{\eta^M_A} MA\rangle$ and $\langle A, g\rangle$. The homomorphism diagram is as follows:

$$\begin{array}{ccccccc} SMA & \xrightarrow{\overline{\mu}^S_A} & SA & \xrightarrow{g} & A & \xrightarrow{\eta^M_A} & MA \\ \downarrow{\scriptstyle Sf} & & & & & \overset{=}{\searrow} & \downarrow{\scriptstyle f} \\ SA & & & \xrightarrow{\quad g \quad} & & & A \end{array}$$

  Since f has the Eilenberg–Moore property, it is the case that $f \cdot \eta^M_A = \mathsf{id}_A$ (as indicated by the dashed arrow).

These conditions are the same as the conditions for $\langle A, f, g\rangle$ being an algebra for the module S, which means that a triple $\langle A, f, g\rangle$ is an algebra for the module S precisely when $\langle\langle A, g\rangle, f\rangle$ is an Eilenberg–Moore algebra for the monad $\lceil M\rceil$. Moreover, a morphism in $\textsc{MAlg}(\lceil M\rceil)$ between $\langle\langle A, g\rangle, f\rangle$ and $\langle\langle B, g'\rangle, f'\rangle$ is given by a $\mathbb{C}$-morphism $k : A \to B$ such that:

- It is a morphism in $\textsc{Alg}(S)$, that is, $k \cdot g = g' \cdot Sk$.
- It is an algebra homomorphism, that is, $k \cdot f = f' \cdot Mk$.

Again, these conditions are the same as the conditions for k being a morphism between the corresponding algebras for S as a module $\langle A, f, g\rangle$ and $\langle B, f', g'\rangle$. This gives us an isomorphism of categories $\textsc{MAlg}(\lceil M\rceil) \cong \textsc{ModAlg}(M, S)$.

By composing the adjunctions $F^{\mathrm{Alg}} \dashv U^{\mathrm{Alg}} : \mathbb{C} \rightharpoonup \textsc{Alg}(S)$ and $F^{\mathrm{EM}} \dashv U^{\mathrm{EM}} : \textsc{Alg}(S) \rightharpoonup \textsc{MAlg}(\lceil M\rceil)$, we get the following:

$$F^{\mathrm{EM}}F^{\mathrm{Alg}} \dashv U^{\mathrm{Alg}}U^{\mathrm{EM}} : \mathbb{C} \rightharpoonup \textsc{MAlg}(\lceil M\rceil) \cong \textsc{ModAlg}(M, S)$$

Now, it is left to verify that this adjunction is as defined in the theorem. The right adjoints:

$$\begin{aligned} U^{\mathrm{Alg}}U^{\mathrm{EM}}\langle\langle A, g\rangle, f\rangle &= U^{\mathrm{Alg}}\langle A, g\rangle = A = U^{\mathrm{ModAlg}}\langle A, f, g\rangle \\ U^{\mathrm{Alg}}U^{\mathrm{EM}}k &= U^{\mathrm{Alg}}k = k = U^{\mathrm{ModAlg}}k \end{aligned}$$

The left adjoints (where $\beta_A : SS^*A \to S^*A$ is the action of the free S-algebra generated by an object A):

$$\begin{aligned} F^{\mathrm{EM}}F^{\mathrm{Alg}}A &= F^{\mathrm{EM}}\langle S^*A, \beta_A\rangle \\ &= \langle \lceil M\rceil\langle S^*A, \beta_A\rangle,\ \mu^{\lceil M\rceil}_{\langle S^*A, \beta_A\rangle}\rangle \\ &= \langle\langle MS^*A,\ SMS^*A \xrightarrow{\delta_{S^*A}} MSS^*A \xrightarrow{M\beta_A} MS^*A\rangle,\ \mu^{\lceil M\rceil}_{\langle S^*A, \beta_A\rangle}\rangle \\ &= \langle\langle MS^*A,\ SMS^*A \xrightarrow{\eta^M_{S^*A}\cdot\beta^S_A\cdot\overline{\mu}^S_{S^*A}} MS^*A\rangle,\ MMS^*A \xrightarrow{\mu^M_{S^*A}} MS^*A\rangle \\ &= F^{\mathrm{ModAlg}}A \end{aligned}$$

$$F^{\mathrm{EM}}F^{\mathrm{Alg}}k = F^{\mathrm{EM}}S^*k = MS^*k = F^{\mathrm{ModAlg}}k$$ □

Theorem 4.21. *If S^* exists, the functor U^{ModAlg} is strictly monadic. This entails that the category $\textsc{ModAlg}(M, S)$ is isomorphic to $\textsc{MAlg}(MS^*)$.*

Proof. We use the strict version of Beck's monadicity theorem. We have already shown that U^{ModAlg} is a right adjoint, so it is left to show that it creates coequalisers for those parallel h_0, h_1 in $\textsc{ModAlg}(M,S)$ for which $U^{\mathrm{ModAlg}}h_0$ and $U^{\mathrm{ModAlg}}h_1$ have a split coequaliser in $\mathbb{C}$.

Let $h_0, h_1 : \langle A, f^A, g^A\rangle \to \langle B, f^B, g^B\rangle$ be such a pair. Let c be a split coequaliser of $U^{\mathrm{ModAlg}}h_0$ and $U^{\mathrm{ModAlg}}h_1$. In other words, there exist morphisms s and t such that the following diagram commutes in $\mathbb{C}$ and in which the two horizontal compositions are the identities:

$$\begin{array}{ccccc} B & \xrightarrow{t} & A & \xrightarrow{h_0} & B \\ \downarrow{\scriptstyle c} & & \downarrow{\scriptstyle h_1} & & \downarrow{\scriptstyle c} \\ C & \xrightarrow{s} & B & \xrightarrow{c} & C \end{array} \tag{4.9}$$

We need to show that there exist unique $f^C : MC \to C$ and $g^C : SA \to A$ such that $\langle C, f^C, g^C\rangle$ is an algebra for a module, and $c : \langle B, f^B, g^B\rangle \to \langle C, f^C, g^C\rangle$ is a homomorphism and a coequaliser of h_0 and h_1. From the monadicity of the forgetful functors $U^{\mathrm{EM}} : \textsc{MAlg}(M) \to \mathbb{C}$ and $U^{\mathrm{Alg}} : \textsc{Alg}(S) \to \mathbb{B}$ (see Theorem 2.49), we obtain that there exist a unique Eilenberg–Moore algebra $\langle C, f^C\rangle$ and a unique S-algebra $\langle C, g^C\rangle$ with such properties, with the actions defined respectively as:

$$f^C = \left(MC \xrightarrow{Ms} MB \xrightarrow{f^B} B \xrightarrow{c} C\right)$$
$$g^C = \left(SC \xrightarrow{Ss} SB \xrightarrow{g^B} B \xrightarrow{c} C\right)$$

It is left to check that the two put together form an algebra for a module, that is, that the tuple $\langle C, f^C, g^C\rangle$ satisfy the coherence condition from the definition of algebras for modules:

$$\begin{aligned} g^C \cdot Sf^C &= c \cdot g^B \cdot Ss \cdot Sc \cdot Sf^B \cdot SMs && \text{(def.)} \\ &= c \cdot g^B \cdot Sh_1 \cdot St \cdot Sf^B \cdot SMs && \text{(diag. (4.9))} \\ &= c \cdot h_1 \cdot g^A \cdot St \cdot Sf^B \cdot SMs && (h_1 \text{ homomorph.}) \\ &= c \cdot h_0 \cdot g^A \cdot St \cdot Sf^B \cdot SMs && (c \text{ coequaliser}) \\ &= c \cdot g^B \cdot Sh_0 \cdot St \cdot Sf^B \cdot SMs && (h_0 \text{ homomorph.}) \\ &= c \cdot g^B \cdot Sf^B \cdot SMs && \text{(diag. (4.9))} \\ &= c \cdot g^B \cdot \overline{\mu}^S_B \cdot SMs && \text{(coherence)} \\ &= c \cdot g^B \cdot Ss \cdot \overline{\mu}^S_C && (\overline{\mu}^S \text{ nat.}) \\ &= g^C \cdot \overline{\mu}^S_C && \text{(def.)} \end{aligned}$$

□

Lemma 4.22. *Let Λ denote the comparison functor* $\textsc{ModAlg}(M,S) \to \textsc{MAlg}(MS^*)$. *Its inverse is given as follows:*

$$\Lambda^{-1}\langle A,\ a : MS^*A \to A\rangle = \langle A,\ MA \xrightarrow{M\eta^{S^*}_A} MS^*A \xrightarrow{a} A,\\ SA \xrightarrow{\mathsf{emb}_A} S^*A \xrightarrow{\eta^M_{S^*A}} MS^*A \xrightarrow{a} A\rangle$$

$$\Lambda^{-1}k = k$$

Proof. We need to see that Λ^{-1} is a left and a right inverse of Λ. Since Λ is an isomorphism (Theorem 4.21), it is enough to show that Λ^{-1} is a left inverse. We unfold the definitions, where ε^{S^*} and ε^M are the counits of the $F^{\mathrm{Alg}} \dashv U^{\mathrm{Alg}} : \mathbb{C} \to \textsc{Alg}(S)$ and $F^{\mathrm{EM}} \dashv U^{\mathrm{EM}} : \textsc{Alg}(S) \to \textsc{MAlg}(\lceil M \rceil)$ adjunctions respectively (compare the proof of Theorem 4.20):

$$\begin{aligned}
&\Lambda^{-1}\Lambda\langle A,\ MA \xrightarrow{f} A,\ SA \xrightarrow{g} A\rangle \\
&= \Lambda^{-1}\langle A,\ MS^*A \xrightarrow{M\varepsilon^{S^*}_{\langle A,g\rangle}} MA \xrightarrow{\varepsilon^M_{\langle A,f\rangle} = f} A\rangle \\
&= \langle A,\ MA \xrightarrow{M\eta^{S^*}_A} MS^*A \xrightarrow{M\varepsilon^{S^*}_{\langle A,g\rangle}} MA \xrightarrow{f} A,\\
&\qquad SA \xrightarrow{\mathsf{emb}_A} S^*A \xrightarrow{\eta^M_{S^*A}} MS^*A \xrightarrow{M\varepsilon^{S^*}_{\langle A,g\rangle}} MA \xrightarrow{f} A\rangle
\end{aligned}$$

We verify that each action is equal. The equality $f \cdot M\varepsilon^{S^*}_{\langle A,g\rangle} \cdot M\eta^{S^*}_A = f$ follows from the properties of adjunctions (zig-zag equalities). The remaining equality can be proven with the following calculation, where $\beta_A : SS^*A \to S^*A$ is the free S-algebra action:

$$\begin{aligned}
f \cdot M\varepsilon^{S^*}_{\langle A,g\rangle} \cdot \eta^M_{S^*A} \cdot \mathsf{emb}_A &= f \cdot \eta^M_A \cdot \varepsilon^{S^*}_{\langle A,g\rangle} \cdot \mathsf{emb}_A && \text{(naturality of } \eta^M\text{)} \\
&= \varepsilon^{S^*}_{\langle A,g\rangle} \cdot \mathsf{emb}_A && (f \text{ is Eilenberg–Moore}) \\
&= \varepsilon^{S^*}_{\langle A,g\rangle} \cdot \beta_A \cdot S\eta^{S^*}_A && \text{(def. of } \mathsf{emb}\text{)} \\
&= g \cdot S\varepsilon^{S^*}_{\langle A,g\rangle} \cdot S\eta^{S^*}_A && (\beta \text{ is free}) \\
&= g && \text{(zig–zag)}
\end{aligned}$$

□

Example 4.23. Algebras for modules allows us to provide a simpler and more abstract proof that $M(\Sigma M)^*$ is a coporduct of M and Σ^* in $\textsc{Mnd}$. First, for two monads M and T, we define an $\langle M,T\rangle$-bialgebra as a triple $\langle A,\ f : MA \to A,\ g : TA \to A\rangle$, where f and g are Eilenberg–Moore algebra actions. All $\langle M,T\rangle$-bialgebras form a category, $\textsc{BiAlg}(M,T)$, with morphisms given by pairs of algebra homomorphisms.

As shown by Kelly [71], in a category with coproducts, if the obvious forgetful functor from $\textsc{BiAlg}(M, T)$ to the base category has a left adjoint, the induced monad is a coproduct of M and T in $\textsc{Mnd}$. Indeed, for an M-module ΣM, one can prove that the category $\textsc{ModAlg}(M, \Sigma M)$ is isomorphic to $\textsc{BiAlg}(M, \Sigma^*)$ as follows:

First, by virtue of Theorem 2.49, $\textsc{MAlg}(\Sigma^*) \cong \textsc{Alg}(\Sigma)$, so we can work with Σ-algebras (instead of Eilenberg–Moore Σ^*-algebras) in the third component of bialgebras. Given an algebra for a module $\langle A,\, f : MA \to A,\, g : \Sigma MA \to A\rangle$, we define the corresponding bialgebra as $\langle A,\, f,\, g \cdot \Sigma\eta^M_A : \Sigma A \to A\rangle$. Given a bialgebra $\langle A,\, f : MA \to A,\, g : \Sigma A \to A\rangle$, we define the corresponding algebra for a module as $\langle A,\, f,\, g \cdot \Sigma f : \Sigma MA \to A\rangle$. The coherence condition holds easily from the fact that f is an Eilenberg–Moore algebra action. Simple calculation reveals that that the two transformations are mutual inverses. It is also easy to verify that a morphism between two algebras for a module is also a morphism between the corresponding bialgebras and *vice versa*.

Theorem 4.20 characterises the left adjoint to U^{ModAlg} (and so, to the forgetful functor $\textsc{BiAlg}(M, \Sigma^*)$). The induced monad is indeed the free monad generated by the module ΣM, that is $M(\Sigma M)^*$.

Example 4.24. For a monad M and an endofunctor Σ on the category $\mathbb{C}$, we define a Σ-and-M-algebra as a triple $\langle A,\, m : MA \to A,\, f : \Sigma A \to A\rangle$, where f is a morphism, and m is an Eilenberg–Moore algebra action. Morphisms between two Σ-and-M-algebras are pairs of algebra homomorphisms. Filinski and Støvring [39] use the initial Σ-and-M-algebra (whose carrier is given by $\mu M\Sigma \cong M(\mu\Sigma M)$) to model effectful datatypes (see also Atkey *et al.* [22]). Employing the isomorphism $\textsc{MAlg}(\Sigma^*) \cong \textsc{Alg}(\Sigma)$, one can easily see that the category of Σ-and-M-algebras is isomorphic to $\textsc{BiAlg}(M, \Sigma^*)$, and so, as described in the previous example, isomorphic to $\textsc{ModAlg}(M, \Sigma M)$. Since $F^{\mathrm{ModAlg}} : \mathbb{C} \to \textsc{ModAlg}(M, \Sigma M)$ is cocontinuous (since it is a left adjoint), the initial Σ-and-M-algebra can be obviously reconstructed as $F^{\mathrm{ModAlg}}0$, where 0 is the initial object of $\mathbb{C}$.

Example 4.25. Another reason to study algebras for modules is the codensity monad construction, which (via the coend formula for right Kan extensions), gives a 'CPS-ed' version of a data structure; see Hinze [59] for an overview. This can be seen as a unification of two approaches to resumptions in programming: the (co)data type approach à la Haskell [54] and the continuation-based à la LISP [34, 57]

4.2.3 Freeness

Every monad M can be seen as its own module with $\overline{\mu}^M = \mu^M$. Moreover, this trivial construction is functorial:

Definition 4.26. We define a functor $U^{\mathrm{Mod}} : \textsc{Mnd} \to \textsc{Mod}$ as follows:

$$U^{\mathrm{Mod}} M = \langle M, M \rangle$$
$$U^{\mathrm{Mod}} f = \langle f, f \rangle$$

For a monad M, the module $U^{\mathrm{Mod}} M$ is called a *tautological* module by Hirschowitz and Maggesi [61], while Mac Lane [78] refers to it as *right regular representation* of M, since it resembles a construction with the same name in the theory of group representation. In this section, we prove that the monad MS^* is free with respect to this functor. But first, we need some auxiliary definitions:

Definition 4.27. Let S be an M-module. We define the two following forgetful functors:

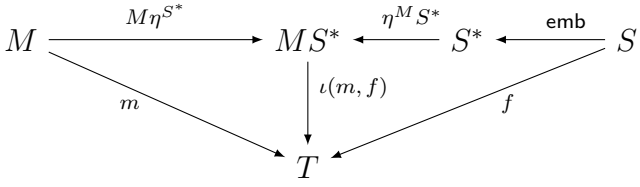
$$U^M : \textsc{ModAlg}(M,S) \to \textsc{MAlg}(M) \qquad U^S : \textsc{ModAlg}(M,S) \to \textsc{Alg}(S)$$
$$U^M \langle A, f, g \rangle = \langle A, f \rangle \qquad U^S \langle A, f, g \rangle = \langle A, g \rangle$$
$$U^M k = k \qquad U^S k = k$$

Recall the isomorphism of categories $\Delta : \textsc{Mnd} \to \mathrm{EM}^{\mathsf{op}}$ (Lemma 2.26), where $\mathrm{EM}^{\mathsf{op}}$ denotes the category of all Eilenberg–Moore categories of monads on the base category and functors that preserve carriers. Since both $\textsc{ModAlg}(M,S)$ and $\textsc{Alg}(S)$ are monadic, the functors U^M and U^S can be seen as (carrier-preserving) functors between Eilenberg–Moore categories, that is, morphisms in EM. Their Δ^{-1}-images are as follows, where $\beta_A : SS^*A \to S^*A$ is the natural family of free algebra actions:

$$\begin{aligned}
\Delta^{-1} U^M &= \left(M \xrightarrow{M\eta^{MS^*}} MMS^* \xrightarrow{p} MS^* \right), \text{ where } U^M F^{\mathrm{ModAlg}} A = \langle MS^*A, p_A \rangle \\
&= \left(M \xrightarrow{M\eta^{MS^*}} MMS^* \xrightarrow{\mu^M S^*} MS^* \right) = \left(M \xrightarrow{M\eta^{S^*}} MS^* \right)
\end{aligned}$$

$$\begin{aligned}
\Delta^{-1} U^S &= \left(S \xrightarrow{S\eta^{MS^*}} SMS^* \xrightarrow{q} MS^* \right), \text{ where } U^S F^{\mathrm{Alg}} A = \langle S^*A, q_A \rangle \\
&= \left(S \xrightarrow{S\eta^{MS^*}} SMS^* \xrightarrow{\overline{\mu}^S S^*} SS^* \xrightarrow{\beta} S^* \xrightarrow{\eta^M S^*} MS^* \right) \\
&= \left(S \xrightarrow{S\eta^{S^*}} SS^* \xrightarrow{\beta} S^* \xrightarrow{\eta^M S^*} MS^* \right) = \left(S \xrightarrow{\mathsf{emb}} S^* \xrightarrow{\eta^M S^*} MS^* \right)
\end{aligned}$$

Theorem 4.28. *The monad MS^* is the free object in the category* Mnd *generated by S with respect to the functor U^{Mod}. More precisely, this means that for monads M and T, an M-module S, and a module morphism $\langle m, f\rangle : \langle M, S\rangle \to U^{\mathrm{Mod}}T$, there exists a unique monad morphism $\iota(m,f) : MS^* \to T$ such that the following diagram commutes:*

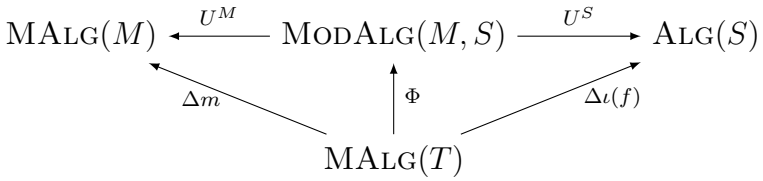


(4.10)

Proof. We define a functor $\Phi : \mathrm{MAlg}(T) \to \mathrm{ModAlg}(M,S)$ as follows:

$$\Phi\langle A,\ a : TA \to A\rangle = \langle A,\ MA \xrightarrow{m_A} TA \xrightarrow{a} A,\ SA \xrightarrow{f_A} TA \xrightarrow{a} A\rangle$$
$$\Phi k = k$$

We define $\iota(m,f) = \Delta^{-1}\Phi$. To see that it makes the diagram (4.10) commute, it is enough (via the isomorphism Δ) to show that the following diagram commutes:



(4.11)

The left-hand side can be proven as follows:

$$\begin{aligned} U^M\Phi\langle A, TA \xrightarrow{a} A\rangle &= U^M\langle A,\ MA \xrightarrow{m_A} TA \xrightarrow{a} A,\ SA \xrightarrow{f_A} TA \xrightarrow{a} A\rangle && \text{(def. of } \Phi) \\ &= \langle A,\ MA \xrightarrow{m_A} TA \xrightarrow{a} A\rangle && \text{(def. of } U^M) \\ &= (\Delta m)\langle A, TA \xrightarrow{a} A\rangle && \text{(def. of } \Delta) \end{aligned}$$

For the right-hand side, we use the isomorphism $\mathrm{Alg}(S) \cong \mathrm{MAlg}(S^*)$:

$$\begin{aligned} U^S\Phi\langle A, TA \xrightarrow{a} A\rangle &= U^S\langle A,\ MA \xrightarrow{m_A} TA \xrightarrow{a} A,\ SA \xrightarrow{f_A} TA \xrightarrow{a} A\rangle && \text{(def. of } \Phi) \\ &= \langle A,\ SA \xrightarrow{f_A} TA \xrightarrow{a} A\rangle \in \mathrm{Alg}(S) && \text{(def. of } U^S) \\ &= \langle A,\ S^\infty A \xrightarrow{\iota(f)_A} TA \xrightarrow{a} A\rangle \in \mathrm{MAlg}(S^*) && \text{(monadicity)} \\ &= (\Delta\iota(f))\langle A, TA \xrightarrow{a} A\rangle && \text{(def. of } \Delta) \end{aligned}$$

To see that Φ is a unique morphism that makes the diagram (4.11) commute, consider a monad morphism $r : MS^* \to T$ such that the diagram (4.10) commutes if we substitute r for $\iota(m,f)$. Since $\eta^M S^* : S^* \to MS^*$ is a monad morphism (since MS^*

is induced by a distributive law, see Lemma 2.33), the composition $r \cdot \eta^M S^* : S^* \to T$ is a monad morphism, hence, from the freeness of S^*, we obtain the following:

$$r \cdot \eta^M S^* = \iota(f) \tag{4.12}$$

We calculate:

$$\begin{aligned}
r &= r \cdot \mu^{MS^*} \cdot \eta^{MS^*} MS^* && \text{(monads)} \\
&= r \cdot (\mu^M * \mu^{S^*}) \cdot M\lambda S^* \cdot (\eta^M * \eta^{S^*} MS^*) && \text{(def.)} \\
&= r \cdot (\mu^M * \mu^{S^*}) \cdot (\eta^M * M\eta^{S^*} S^*) && \text{(distr. law)} \\
&= r \cdot (\mu^M * \mu^{S^*}) \cdot (M\eta^M * \eta^{S^*} S^*) && \text{(monads)} \\
&= r \cdot (\mu^M * \mu^{S^*}) \cdot M\lambda S^* \cdot (M\eta^{S^*} * \eta^M S^*) && \text{(distr. law)} \\
&= r \cdot \mu^{MS^*} \cdot (M\eta^{S^*} * \eta^M S^*) && \text{(def. of } \mu^{MS^*}\text{)} \\
&= \mu^T \cdot (r * r) \cdot (M\eta^{S^*} * \eta^M S^*) && \text{(monad morphism)} \\
&= \mu^T \cdot (m * \iota(f)) && \text{(LHS of (4.10) and (4.12))}
\end{aligned}$$

Thus, it is the case that:

$$(\Delta r)\langle A, TA \xrightarrow{a} A\rangle = \langle A,\ MS^*A \xrightarrow{m*\iota(f)} TTA \xrightarrow{\mu^T_A} TA \xrightarrow{a} A\rangle \tag{4.13}$$

The right-hand side of (4.13) is (by Lemma 4.22) the following algebra for the module S:

$$\begin{aligned}
\langle A,\ MA &\xrightarrow{M\eta^{S^*}_A} MS^*A \xrightarrow{m*\iota(f)} TTA \xrightarrow{\mu^T_A} TA \xrightarrow{a} A, \\
SA &\xrightarrow{\mathsf{emb}_A} S^*A \xrightarrow{\eta^M_{S^*A}} MS^*A \xrightarrow{m*\iota(f)} TTA \xrightarrow{\mu^T_A} TA \xrightarrow{a} A\rangle
\end{aligned}$$

We simplify each component. The first action:

$$\begin{aligned}
a \cdot \mu^T_A \cdot (m * \iota(f)) \cdot M\eta^{S^*}_A &= a \cdot \mu^T_A \cdot (m * \eta^T_A) && (\iota(f) \text{ is a monad morph.}) \\
&= a \cdot m && \text{(monad laws)}
\end{aligned}$$

The second action:

$$\begin{aligned}
a \cdot \mu^T_A \cdot (m * \iota(f)) \cdot \eta^M_{S^*A} \cdot \mathsf{emb}_A &= a \cdot \mu^T_A \cdot (\eta^T * \iota(f)) \cdot \mathsf{emb}_A && \text{(monad morphism)} \\
&= a \cdot \iota(f) \cdot \mathsf{emb}_A && \text{(monad laws)} \\
&= a \cdot f && \text{(free monad)}
\end{aligned}$$

Thus,

$$(\Delta r)\langle A,\ TA \xrightarrow{a} A\rangle = \langle A,\ MA \xrightarrow{m_A} TA \xrightarrow{a} A,\ SA \xrightarrow{f_A} TA \xrightarrow{a} A\rangle = \Phi\langle A,\ TA \xrightarrow{a} A\rangle$$

This means that $\Delta r = \Phi$, and so $r = \Delta^{-1}\Phi = \iota(m, f)$. □

Chapter 5

Coinductive resumptions

Resumptions (also known as *transducers*) were introduced by Milner [87] to model communicating processes. Abramsky [2] presented them in the language of coalgebra as elements of the carrier of the final coalgebra $\nu X.(O \times X)^I$ in $\textsc{Set}$, where I and O are sets of input and output symbols respectively. Intuitively, such a resumption is a function that consumes a piece of input, and returns a piece of output together with a new resumption, representing the behaviour of a process in a step-wise manner. Since resumptions are given by final coalgebras, the consuming and producing can be iterated indefinitely. A process can have a *local* effect, such as finite non-determinism, described by a monad M. In such a case, Abramsky's definition can be generalised to $\nu X.M(O \times X)^I$ (Hasuo and Jacobs [55]).

Moggi [88] introduced the resumption monad $M(\Sigma M)^*$ (on any category with coproducts and enough initial algebras), where M is the local effect, while Σ is a generalisation of the 'signature' of communication (given in Abramsky's resumptions by $(O \times (\text{-}))^I$). Since Moggi's construction is a monad, it can be used in the monadic semantics of the computational λ-calculus [89].

However, Moggi's monad is given via initial algebra, so it is 'inductive'. Thus, it is natural to ask for the 'completely iterative' version. Goncharov and Schröder [50] introduced a monad $MM^\infty \cong \nu X.M(X + (\text{-}))$. In this chapter, we generalise this construction to MS^∞, for any right M-module S (where S as an endofunctor is iteratable), mirroring the inductive resumption monad MS^* given in the previous chapter. More precisely, we show the following:

– The composition MS^∞ is a monad. Similarly to the construction of the monadic structure of MS^*, we adapt Hyland, Plotkin, and Power's [65] proof for Moggi's resumptions. We use the isomorphism between Eilenberg–Moore algebras for S^∞ and complete Elgot S-algebras (see Theorem 2.83).

– We show that MS^∞ is completely iterative. If M is also a cim, the monad MS^∞ allows to unfold both the structure of S^∞ and M simultaneously. Technically, this means that MS^∞ is a cim with respect to the module $\overline{M}S^\infty + MSS^\infty$.

– We introduce *resumption algebras*. They are similar to algebras for modules (Definition 4.19), but the S-algebra part is a complete Elgot algebra and additional coherence is required. They correspond to Eilenberg–Moore algebras for MS^∞.

– We characterise the monad MS^∞ with a universal property similar to the universal property of MS^* given in the previous chapter.

– We introduce *two-sided cims*, which are similar to cims, but have a slightly different guardedness condition. We show that every cim freely extends to a two-sided cims. This allows us to prove that the free two-sided extension of the monad $M(\Sigma M)^\infty$ for an endofunctor Σ is the coproduct in the category of two-sided cims of a two-sided cim M and Σ^∞ (which happens to be two-sided from the start).

5.1 Monadic structure

Let $\langle S, \overline{\mu}^S \rangle$ be a right M-module such that S^∞ exists. We give a monadic structure to MS^∞ via a distributive law [27], similarly to MS^* detailed in the previous chapter. In this case, our construction is again an adaptation of Hyland, Plotkin, and Power's proof [65] that the inductive resumptions $M(\Sigma M)^*$ form a monad (Theorem 2.50). We use the fact due to Adámek, Milius, and Velebil [15] that the category of complete Elgot algebras is strictly monadic over the base category $\mathbb{C}$ (see Theorem 2.83). Note that we cannot employ Uustalu's construction on parametrised monads [116] (successfully used by Goncharov and Schröder [50]), since MS^∞ is not in general given by the carrier of a final coalgebra; moreover, we make extensive use of the distributive law in the subsequent developments.

Definition 5.1. For a monad M and a right M-module S, we define the following two morphisms, both natural in A and B.

$$\mathsf{flatl} = \big(S(MA+B) \xrightarrow{S(\mathsf{id}+\eta^M)} S(MA+MB) \xrightarrow{S[M\mathsf{inl},M\mathsf{inr}]} SM(A+B) \xrightarrow{\overline{\mu}^S} S(A+B)\big)$$

$$\mathsf{flatr} = \big(S(A+MB) \xrightarrow{S(\eta^M+\mathsf{id})} S(MA+MB) \xrightarrow{S[M\mathsf{inl},M\mathsf{inr}]} SM(A+B) \xrightarrow{\overline{\mu}^S} S(A+B)\big)$$

Definition 5.2. Recall that $\langle S, \overline{\mu}^S \rangle$ is a right M-module. For a complete Elgot algebra $\langle A, a : SA \to A, (\text{-})^\dagger \rangle$, we define an algebra $\langle MA,\, a' : SMA \to MA,\, (\text{-})^\ddagger \rangle$. The action is defined as the following composition:

$$a' = \left(SMA \xrightarrow{\overline{\mu}^S_A} SA \xrightarrow{a} A \xrightarrow{\eta^M_A} MA \right)$$

For a flat equation morphism $e : X \to SX + MA$, we define an auxiliary morphism $|e|$ and the solution $e^\ddagger$:

$$|e| = \left(SX + A \xrightarrow{Se+\mathsf{id}} S(SX + MA) + A \xrightarrow{\mathsf{flatr}_{SX,A}+\mathsf{id}} S(SX + A) + A \right)$$
$$e^\ddagger = \left(X \xrightarrow{e} SX + MA \xrightarrow{\mathsf{inl}+\mathsf{id}} SX + A + MA \xrightarrow{|e|^\dagger+\mathsf{id}} A + MA \xrightarrow{[\eta^M_A,\mathsf{id}]} MA \right)$$

Lemma 5.3. *The triple $\langle MA, a', (\text{-})^\ddagger \rangle$ from Definition 5.2 is a complete Elgot algebra. Moreover, the assignment $\lceil M \rceil \langle A, a, (\text{-})^\dagger \rangle = \langle MA, a', (\text{-})^\ddagger \rangle$ on objects and $\lceil M \rceil f = Mf$ on morphisms is an endofunctor on $\textsc{Elgot}(S)$ with a monadic structure given by the monadic structure of M.*

Proof. The proof amounts to mundane verification that the necessary axioms of Elgot algebras hold. See Section A.3 in the appendix for the details. □

Theorem 5.4. *Given a monad M and an M-module S such that the free cim S^∞ exists, the composition MS^∞ can be given a monadic structure via a distributive law between monads $\lambda : S^\infty M \to MS^\infty$.*

Proof. The assignment from Lemma 5.3 is a monad, so it is a lifting of M to $\textsc{Elgot}(S)$ with respect to U^{Elg}. Thus, by Theorem 2.83, it is a lifting of M to $\textsc{Malg}(S^\infty)$. This induces a distributive law between monads $\lambda : S^\infty M \to MS^\infty$, which gives a monadic structure to MS^∞. (Compare proofs of Theorems 2.50 and 4.16.) □

Example 5.5. Let $\mathbb{C}$ be $\textsc{Set}$, D be the monad of discrete probability distributions, $O = \{\mathsf{a}, \mathsf{b}, \ldots\}$ be a set, and $\Sigma X = O \times X$ be an endofunctor. An element of the carrier of the monad of the monad $\mathsf{D}(\Sigma\mathsf{D})^\infty X$ is a countably branching, possibly infinite decision tree in which the nodes (except for the root) are labelled with elements of the set O, edges with probabilities, and leaves with elements of X. Such a structure can be understood as a denotation of a possibly non-terminating, probabilistic process that produces a stream of elements of O as its output.

From the technical perspective, it is important that the root has no label – we take a probabilistic step before generating a label and a probabilistic step before reaching a leaf. This way, when we substitute a tree for a variable, we take two probabilistic

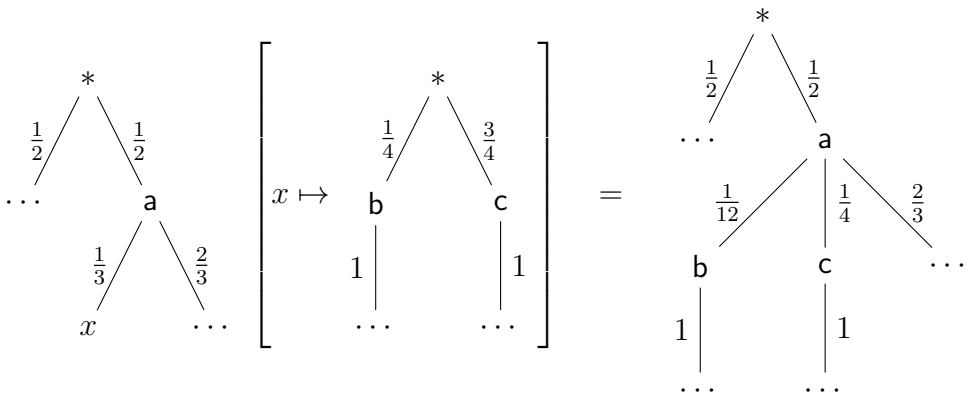


Figure 5.1: Example of a substitution in the $\mathsf{D}(O \times \mathsf{D}(\text{-}))^\infty$ monad.

steps before generating a label or reaching a leaf. The monadic structure of $\mathsf{D}(\Sigma\mathsf{D})^\infty X$ takes care of flattening these to one probabilistic step by multiplying out the adjacent distributions; see Figure 5.1 for an example.

Example 5.6. a) We can consider a coinductive version of the logging state monad given in Example 4.18. It is defined as $\mathsf{RW}^\infty X = ((\text{-} \times A)^\infty X)^A$. Given an initial state, it unfolds a possibly infinite stream of states that the program runs through during the execution.

b) The logging state monad is a bit different than the states monad defined in Section 3.4. In the coinductive logging state monad, every new state is saved in the stream, and the last element (if it exists) is the 'current state'. In the states monad, there is a separate 'current state' element, which can be saved (attached to the end of the stream), and which is overwritten when it is composed with another computation. The states monad, however, turns out to be an instance of the coinductive resumption monad. Since $\mathsf{W} \dashv \mathsf{R}$, Example 2.57 tells us that W is a module of RW. We instantiate the coinductive resumption monad with this data, and obtain the monad RWW^∞, which is isomorphic to states.

The direct definition of the distributive law can be given in terms of an unfold, that is, as a unique homomorphism to a final coalgebra:

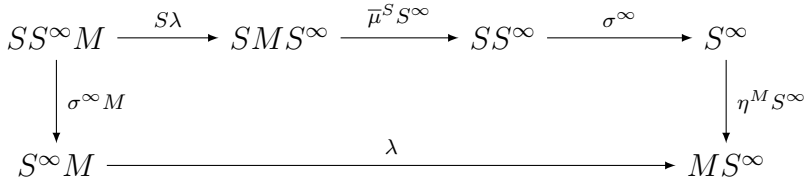
$$\lambda_A = \Big(S^\infty MA \xrightarrow{\xi_{MA}} SS^\infty MA + MA \xrightarrow{\mathsf{inl}+\mathsf{id}} SS^\infty MA + A + MA \xrightarrow{[\![|\xi_{MA}|]\!]+\mathsf{id}} S^\infty A + MA \xrightarrow{[\eta^M_{S^\infty A}, M\eta^\infty_A]} MS^\infty A\Big),$$

where $[\![\,|\xi_{MA}|\,]\!]$ is the unique coalgebra homomorphism to the final coalgebra $\langle S^\infty A, \xi_A\rangle$ from the coalgebra $\langle SS^\infty MA + A, |\xi_{MA}|\rangle$, whose action is given by the following composition:

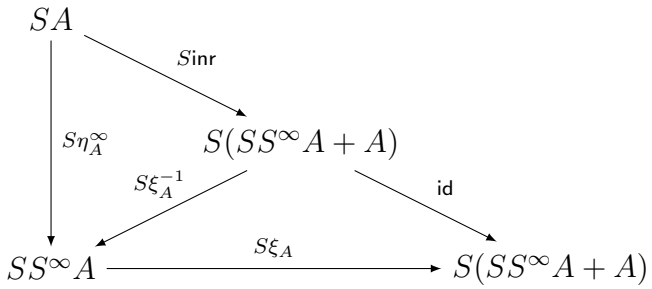
$$|\xi_{MA}| = \Big(SS^\infty MA + A \xrightarrow{S\xi_{MA}+\mathsf{id}} S(SS^\infty MA + MA) + A \xrightarrow{\mathsf{flatr}+\mathsf{id}} S(SS^\infty MA + A) + A\Big)$$

Consult Section A.4 in the appendix for the details. This characterisation can be applied to prove the following lemma:

Lemma 5.7. *The following diagram commutes:*



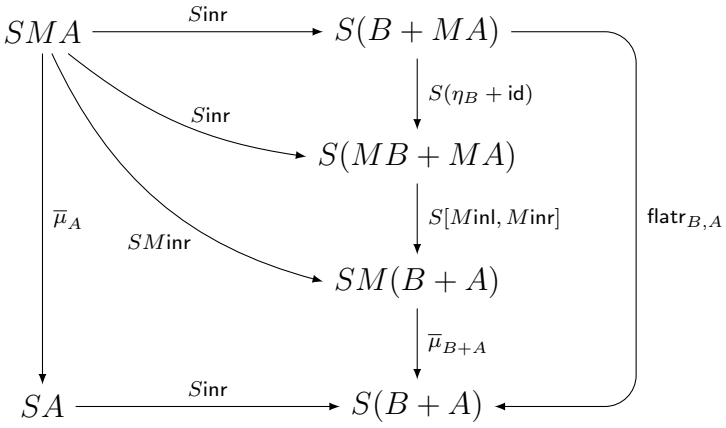
Proof. The following diagrams commutes for all A:



(5.1)

(The triangle on the left follows from the definition of η^∞, the triangle on the right follows from the fact that ξ is an isomorphism.)

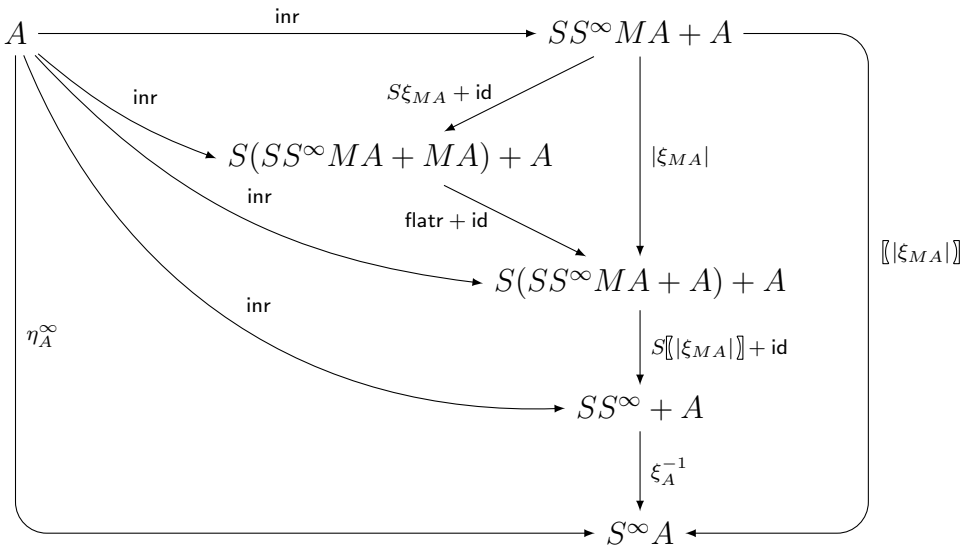
The following diagram commutes for all A and B:



(5.2)

(The two triangles follow from the properties of coproducts, the square in the bottom-left corner follows from the naturality of $\overline{\mu}$, the square on the right is the definition of flatr)

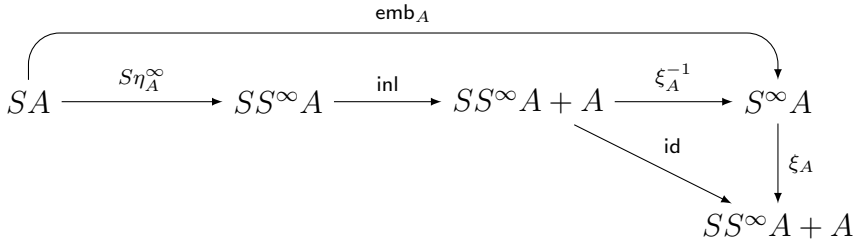
The following diagram commutes for all A:



(5.3)

(From the definitions of $[\![|\xi_{MA}|]\!]$, $|\xi_{MA}|$, and η^∞; and properties of coproducts.)

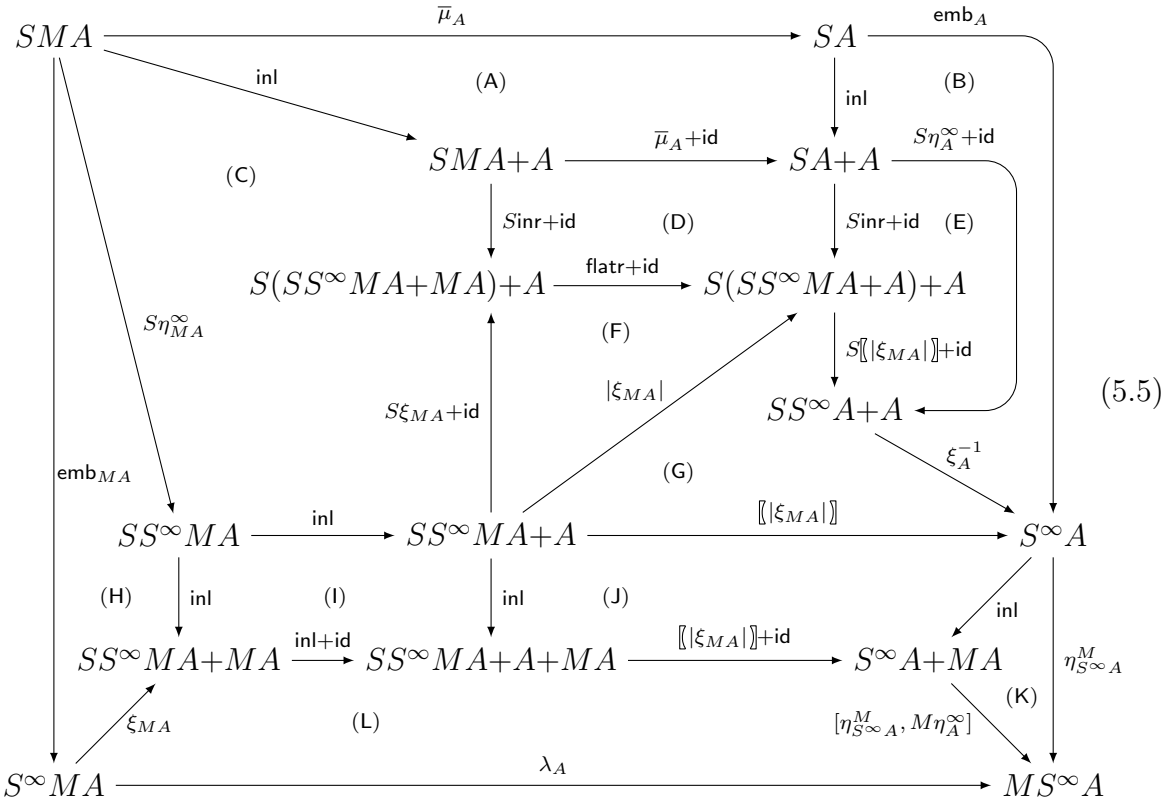
The following diagram commutes for all A:



(5.4)

(From the definition of emb and the fact that ξ is an isomorphism.)

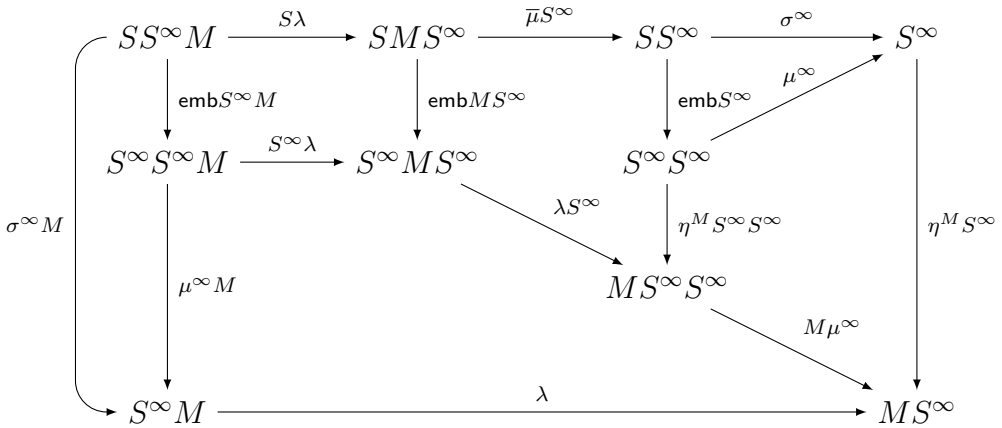
The following diagram commutes for all A:



(5.5)

((A) naturality of inl, (B) definition of emb, (C) diagram (5.1) and naturality of inl, (D) diagram (5.2), (E) diagram (5.3), (F) definition of $|\xi_{MA}|$, (G) $[\![\text{-}]\!]$ is a morphism between coalgebras, (H) diagram (5.4), (I) naturality of inl, (J) naturality of inl, (K) coproduct, (L) direct definition of λ.)

The following diagram commutes. Its perimeter is the desired diagram.



(The triangle on the left and in the top-right corner follow from the definition of σ^∞. The square in the top-left corner follows from the naturality of emb. The pentagon in

the middle is the diagram (5.5). The square on the right follows from the naturality of η^M. The bottom pentagon follows from the fact that λ is a distributive law.) $\square$

The lemma above is used in the proof of the following theorem, which states that the distributive law preserves the module SS^∞ of S^∞. The corollary is that the monad MS^∞ is idealised with $\overline{M}S^\infty + MSS^\infty$ (whenever M is, possibly trivially, idealised with $\overline{M}$), and, later on, that it is completely iterative with respect to this module.

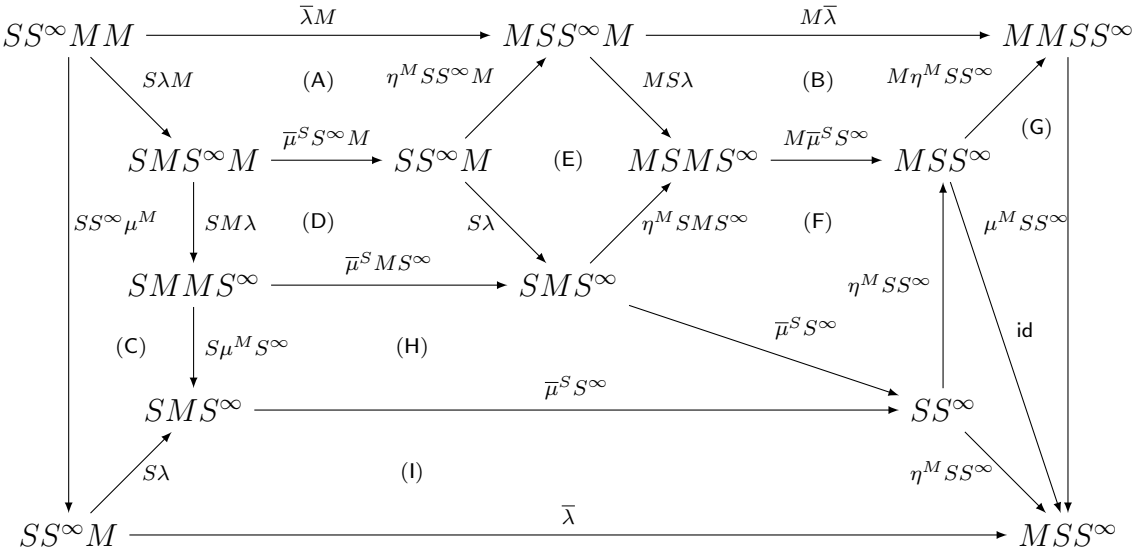
Theorem 5.8. *We define the following natural transformation:*

$$\overline{\lambda} = \left(SS^\infty M \xrightarrow{S\lambda} SMS^\infty \xrightarrow{\overline{\mu}^S S^\infty} SS^\infty \xrightarrow{\eta^M SS^\infty} MSS^\infty \right)$$

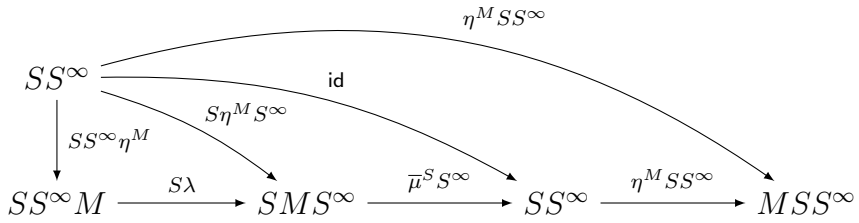
The pair $\langle \lambda, \overline{\lambda} \rangle$ is a distributive law of the idealised monad $\langle S^\infty, SS^\infty, \sigma^\infty \rangle$ over the monad M.

Proof. Unpacking Definition 4.12, we need to check: (a) that $\overline{\lambda}$ is a distributive law of SS^∞ as an endofunctor over the monad M, (b) a coherence condition saying that $\langle \lambda, \overline{\lambda} \rangle$ is a distributive law of the S^∞-module SS^∞ over M (Definition 4.8), and (c) that $\langle \lambda, \overline{\lambda} \rangle$ commutes with σ^∞.

(a) We show that the two desired diagrams commute:

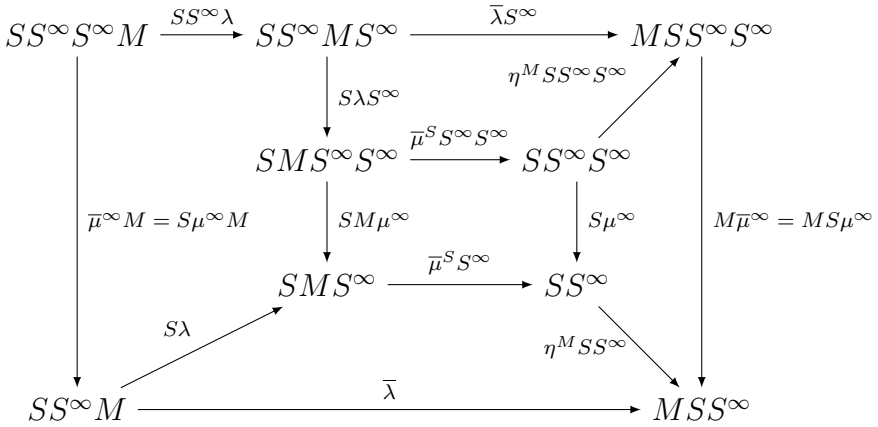


((A) and (B) definition of $\overline{\lambda}$, (C) λ is a distributive law of S^∞ over M, (D) naturality of $\overline{\mu}$, (E) and (F) naturality of η^M, (G) right-unit monad law, (H) S is a module, (I) definition of $\overline{\lambda}$.)



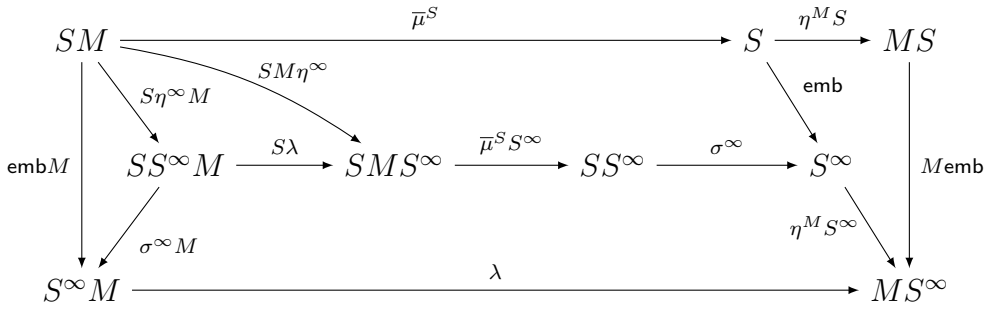
(From left to right: λ is a distributive law, $\overline{\mu}^S$ is a module, identity.)

(b) To see that $\langle \lambda, \overline{\lambda} \rangle$ is a distributive law of a module over a monad, it is left to verify the coherence condition given in Definition 4.8:



(Clockwise starting with the pentagon on the left: λ is a distributive law, definition of $\overline{\lambda}$, naturality of η^M, definition of $\overline{\lambda}$. The square in the centre follows from the naturality of $\overline{\mu}^S$.)

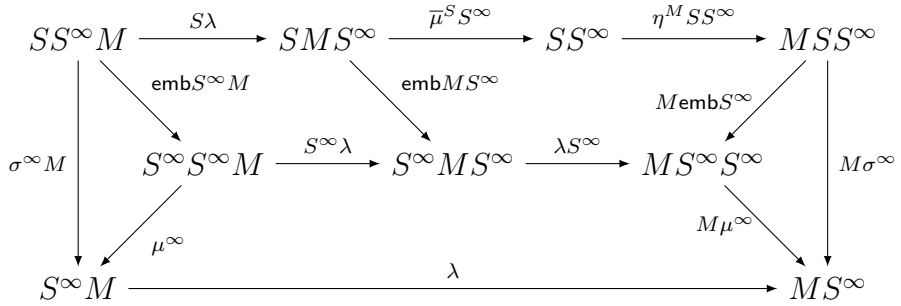
(c) It is left to check that $\langle \lambda, \overline{\lambda} \rangle$ also commutes with σ^∞ (Definition 4.12). Consider the following diagram:



(5.6)

It commutes: the only non-trivial part is the hexagon on the bottom, which is given

by Lemma 5.7. We use it to show that the desired diagram commutes:



(The pentagon at the top-right corner is the diagram (5.6) composed with S^∞, the pentagon at the bottom follows from the fact that λ is a distributive law, the rest is trivial.) □

Corollary 5.9. *Assume that M is idealised. Then, the monad MS^∞ is idealised with $\overline{M}S^\infty$ (by Example 2.60 (1)), MSS^∞ (Lemma 4.13 together with the lemma above), and $\overline{M}S^\infty + MSS^\infty$ (by Example 2.60 (2)).*

5.2 Complete iterativity

We show that the monad MS^∞ defined in the previous section is completely iterative. We consider two cases:

- M is any monad. In this case, MS^∞ is completely iterative with respect to the module MSS^∞. Informally, it means that a guarded equation morphism adds at least one 'level' to the structure of S^∞.
- M is a cim. In this case, the monad MS^∞ is completely iterative with respect to the module $\overline{M}S^\infty + MSS^\infty$, which means that we do not have to add structure to S^∞, as long as we perform a guarded computation in M.

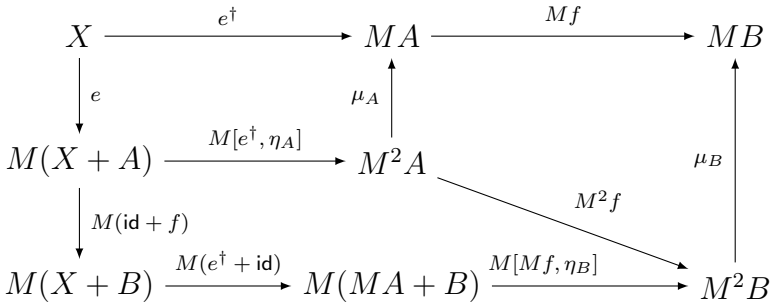
But first, we need a number of auxiliary lemmata.

5.2.1 More properties of cims

We give a few technical results concerning completely iterative monads. First, we show that taking solutions of guarded equation morphisms is functorial in parameters. We need the following technical lemma:

Lemma 5.10. *Let $f : A \to B$ be a morphism, M be a cim, and $e : X \to M(X + A)$ be a guarded equation morphism. Then, $Mf \cdot e^\dagger = (M(\mathsf{id} + f) \cdot e)^\dagger$.*

Proof. The diagram below states that $Mf \cdot e^\dagger$ is a solution of $M(\mathsf{id} + f) \cdot e$. The lemma follows from the uniqueness of solutions.

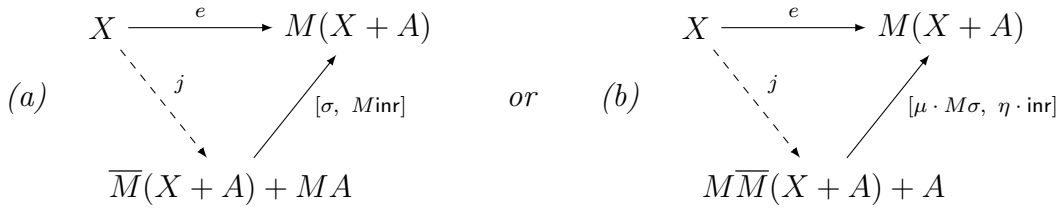


(The top-left square is the solution property of $e^\dagger$. The square on the right is the naturality of μ. The bottom pentagon follows from properties of coproducts.) □

We can construct a category GEqMor with guarded equation morphisms as objects. A morphism between $e : X \to M(X + A)$ and $s : X \to M(X + B)$ is a $\mathbb{C}$-morphism $f : A \to B$ such that $s = M(\mathsf{id} + f) \cdot e$. There are no morphisms between equation morphisms with different objects of variables (although one can easily find a similar parametricity for variables—we leave the details to the reader). In the light of the lemma above, the operation $(\text{-})^\dagger$ ('solving') can be extended to a functor $(\text{-})^\dagger : \text{GEqMor} \to \text{Arr}(\text{Kleisli}(M))$, where $\text{Arr}(\text{Kleisli}(M))$ is the arrow category of the Kleisli category of the monad M. The functor takes a GEqMor-morphism $f : A \to B$ to $F^{\text{Kl}} f = \eta_B \cdot f$.

Next, we show some alternative formulations of the notion of complete iterativity: slightly different guardedness conditions and additional monadic structure in the parameter.

Lemma 5.11. *Let M be completely iterative with respect to a module $\overline{M}$. Let $e : X \to M(X + A)$ factor as one of the following two:*



Then, the morphism e has a unique solution.

Proof. (a) To construct the solution of e, we first define the following morphism:

$$\widetilde{e} = \Big(\overline{M}(X+A) \xrightarrow{\overline{M}(j+\mathsf{id})} \overline{M}(\overline{M}(X+A)+MA+A) \xrightarrow{\overline{M}(\mathsf{id}+[\mathsf{id},\eta_A])} \overline{M}(\overline{M}(X+A)+MA) \xrightarrow{\mathsf{flatr}} \overline{M}(\overline{M}(X+A)+A) \xrightarrow{\sigma} M(\overline{M}(X+A)+A)\Big)$$

The morphism $\widetilde{e}$ is a guarded equation morphism in M, since the lattermost σ can be factored as $[\sigma,\ \eta\cdot\mathsf{inr}]\cdot\mathsf{inl}$. This means that $\widetilde{e}$ has a unique solution $\widetilde{e}^\dagger : \overline{M}(X+A) \to MA$. We define the solution of e as:

$$e^\ddagger = \Big(X \xrightarrow{j} \overline{M}(X+A)+MA \xrightarrow{[\widetilde{e}^\dagger,\mathsf{id}]} MA\Big)$$

It is left to verify that $e^\ddagger$ is indeed a solution of e and that it is unique. The proof amounts to mundane calculation. The details are given in Section A.5 in the appendix.

(b) We define the following morphism:

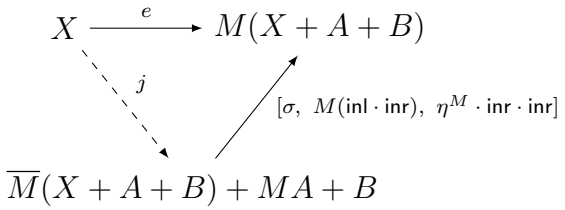
$$f = \Big(\overline{M}(X+A) \xrightarrow{\overline{M}(j+\mathsf{id})} \overline{M}(M\overline{M}(X+A)+A+A) \xrightarrow{\overline{M}(\mathsf{id}+[\mathsf{id},\mathsf{id}])} \overline{M}(M\overline{M}(X+A)+A) \xrightarrow{\mathsf{flatl}} \overline{M}(\overline{M}(X+A)+A) \xrightarrow{\sigma} M(\overline{M}(X+A)+A)\Big)$$

Using the same argument as for $\widetilde{e}$ in (a), f is guarded, so it has a unique solution $f^\dagger : \overline{M}(X+A) \to MA$. We define the solution of e as follows:

$$e^\ddagger = \Big(X \xrightarrow{j} M\overline{M}(X+A)+A \xrightarrow{Mf^\dagger+\mathsf{id}} MMA+A \xrightarrow{[\mu_A,\eta_A]} MA\Big)$$

The proof that $e^\ddagger$ is a unique solution of e is shown in Section A.6 in the appendix. □

Corollary 5.12. *Let M be completely iterative with respect to a module $\overline{M}$. Let $e : X \to M(X+A+B)$ factor as follows:*



Then, the morphism e has a unique solution.

Proof. Using properties of coproducts and the naturality of η, the morphism e can be rewritten as follows:

$$e = [\sigma, M\mathsf{inr}]\cdot(\mathsf{id}+[M\mathsf{inl},\eta\cdot\mathsf{inr}])\cdot j$$

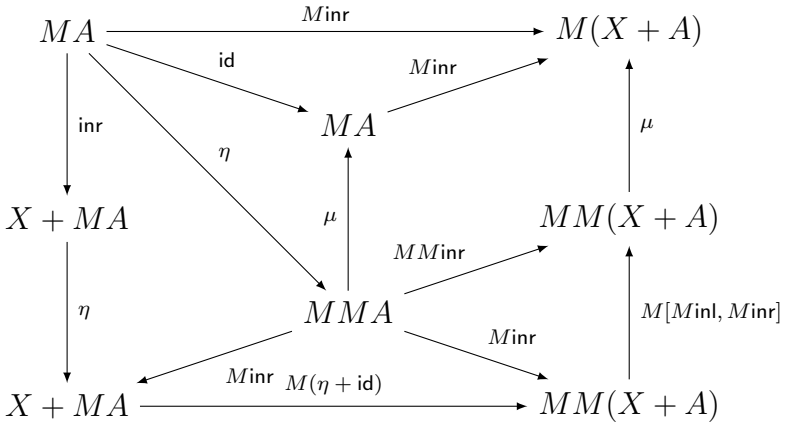
Thus, it factors as in Lemma 5.11 (a), so it has a unique solution. □

Lemma 5.13. *Let M be a cim. Let $e : X \to M(X + MA)$ be a guarded equation morphism. Then, there exists a unique solution to the morphism $\mathsf{flatr}_{X,A} \cdot e : X \to M(X + A)$ given by $\mu_A \cdot e^\dagger$.*

Proof. First, we show that the following diagram commutes:

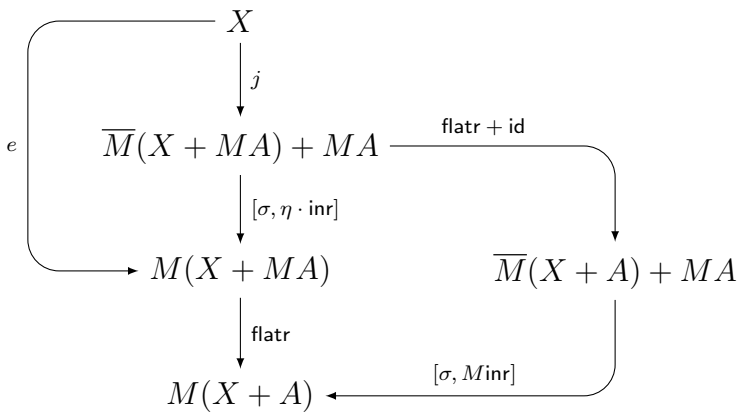
$$\begin{array}{ccc} \overline{M}(X + MA) + MA & \xrightarrow{\mathsf{flatr} + \mathsf{id}} & \overline{M}(X + A) + MA \\ \downarrow{\scriptstyle [\sigma,\ \eta \cdot \mathsf{inr}]} & & \downarrow{\scriptstyle [\sigma, M\mathsf{inr}]} \\ M(X + MA) & \xrightarrow{\mathsf{flatr}} & M(X + A) \end{array} \tag{5.7}$$

It is enough to prove this for each component of the coproduct. The left-hand side component is trivial, since both paths are equal. For the right-hand component consider the following diagram, whose bottom–right path is equal to flatr:



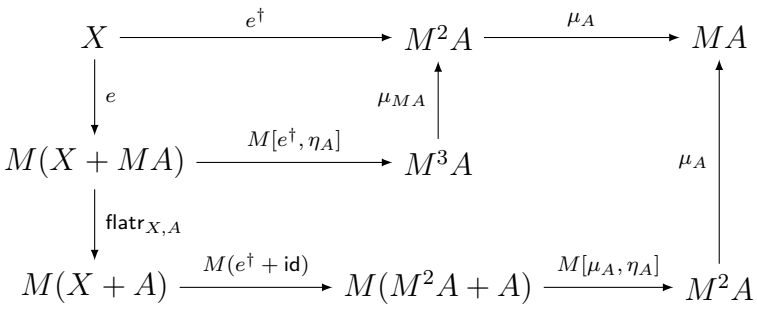
(Each part commutes due to naturality, monad laws, or properties of coproducts.)

The morphism e is guarded, which means that there exists a morphism $j : X \to \overline{M}(X + A) + A$ such that $e = [\sigma,\ \eta \cdot \mathsf{inr}] \cdot j$. The following diagram commutes:

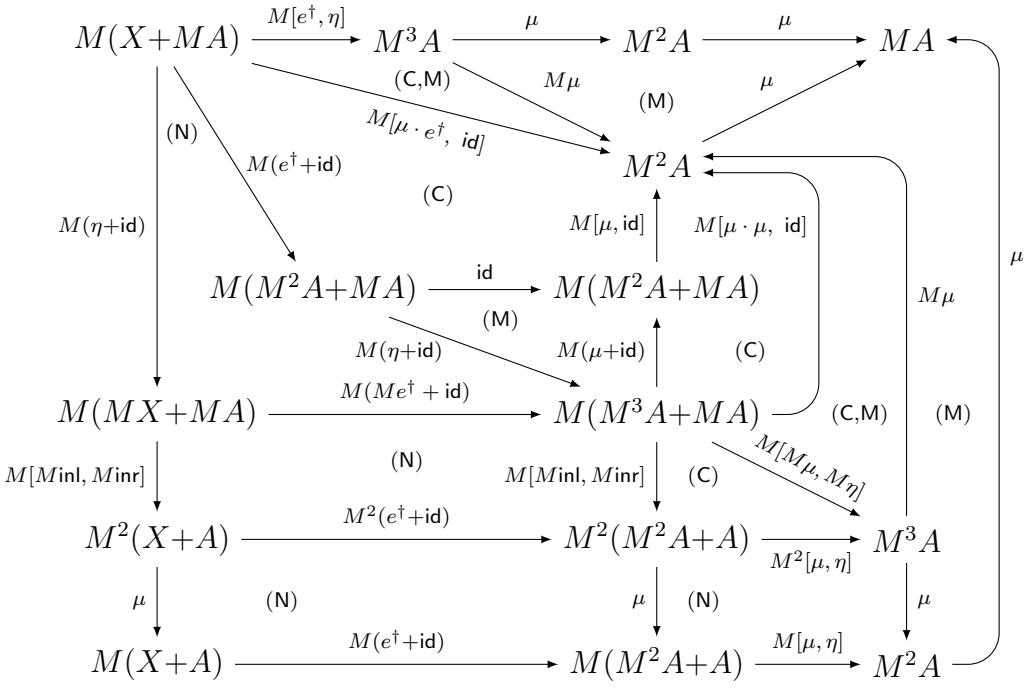


(The square in the bottom-right is the diagram (5.7).)

Following the path on the right-hand side of the diagram, we read that the morphism $\mathsf{flatr}_{X,A} \cdot e$ factors as in Lemma 5.11 (a), which means that the morphism $\mathsf{flatr}_{X,A} \cdot e$ has a unique solution. The following diagram commutes, which means that this solution is equal to $\mu_A \cdot e^\dagger$.



(The square in the top-left corner is the solution diagram of $e^\dagger$. The heptagon is detailed in the following diagram, where the left-most edge is equal to flatr.)



((N) naturality, (M) monad laws, (C) properties of coproducts.) □

5.2.2 An alternative definition of the Kleisli category

Below, we propose an alternative definition of the Kleisli category for M, which is less customary but isomorphic to the usual one. Though the new definition makes descriptions of some constructions a bit more cumbersome (for example, coproducts),

its advantage is a straightforward correspondence between diagrams in KLEISLI and the base category $\mathbb{C}$.

Definition 5.14. For a monad M on the base category $\mathbb{C}$, we define its *Kleisli category*, denoted as KLEISLI$'$. It has the same objects as $\mathbb{C}$. For two objects A and B, an arrow $A \to\!\!\!\!\triangleright\ B$ in KLEISLI$'$ is an arrow $f : MA \to MB$ in $\mathbb{C}$ such that $\mu \cdot Mf = f \cdot \mu$ (that is, it behaves like a homomorphism between two free Eilenberg-Moore algebras).

The categories $\mathbb{C}$ and KLEISLI$'$ share the collection of objects, so, to avoid confusion, we use a different notation for morphisms in KLEISLI$'$, namely $\to\!\!\!\!\triangleright$.

Theorem 5.15. *The categories* KLEISLI$'$ *and* KLEISLI *are isomorphic.*

Remark 5.16. The definition above is similar to the definition of the full subcategory of MALG(M) with free Eilenberg-Moore algebras as objects (which is equivalent to KLEISLI$'$—see, for example, Lambek and Scott's textbook [76, Part 0, Corollary 6.9]) with the difference that the algebras additionally keep the original $\mathbb{C}$-object, instead of its M-image only. This observation can be formalised by saying that KLEISLI$'$ (and so, of course, KLEISLI) is the pullback in CAT of the functors $M : \mathbb{C} \to \mathbb{C}$ and $U^{\mathrm{FEMA}} : \mathrm{FEMA}(M) \to \mathbb{C}$, where FEMA$(M)$ is the mentioned category of free Eilenberg-Moore algebras for M, and U^{FEMA} is the obvious restriction of the forgetful functor $U^{\mathrm{EM}} : \mathrm{MALG}(M) \to \mathbb{C}$.

Lemma 5.17. *The category* $\mathbb{C}$ *has coproducts, which means that* KLEISLI$'$ *also has coproducts. The coproduct of* A *and* B *is given by* $M(A + B)$*. The injections are given by the following morphisms in* $\mathbb{C}$*:*

$$MA \xrightarrow{M\mathsf{inl}} M(A + B) \xleftarrow{M\mathsf{inr}} MB$$

For two KLEISLI$'$*-morphisms* $f : A \to\!\!\!\!\triangleright\ C$ *and* $g : B \to\!\!\!\!\triangleright\ C$*, their mediator is given as the following composition in* $\mathbb{C}$*:*

$$M(X + A) \xrightarrow{M(\eta^M + \eta^M)} M(MX + MA) \xrightarrow{M[f,g]} MMC \xrightarrow{\mu^M} MC$$

Remark 5.18. For two $\mathbb{C}$-morphisms $f : A \to C$ and $g : B \to C$, and a KLEISLI$'$-morphism $k : D \to\!\!\!\!\triangleright\ C$, the coproduct mediators in KLEISLI$'$ simplify to the following

compositions in $\mathbb{C}$:

$$[Mf, k] = \Big(M(A+D) \xrightarrow{M(\mathsf{id}+\eta^M)} M(A+MD) \xrightarrow{M(f+k)} M(C+MC) \xrightarrow{\mathsf{flatr}} M(C+C) \xrightarrow{M[\mathsf{id},\mathsf{id}]} MC\Big)$$

$$[k, Mg] = \Big(M(D+B) \xrightarrow{M(\eta^M+\mathsf{id})} M(MD+B) \xrightarrow{M(k+f)} M(MC+C) \xrightarrow{\mathsf{flatl}} M(C+C) \xrightarrow{M[\mathsf{id},\mathsf{id}]} MC\Big)$$

$$[Mf, Mg] = \Big(M(A+B) \xrightarrow{M[f,g]} MC\Big)$$

5.2.3 Lifting free cims to Kleisli categories

The distributive law $\lambda : S^\infty M \to MS^\infty$ gives rise to the lifting $\lfloor S^\infty \rfloor$ in Kleisli$'$. It is given by:

$$\lfloor S^\infty \rfloor A = S^\infty A$$
$$\lfloor S^\infty \rfloor (f : A \nrightarrow B) =$$
$$\Big(MS^\infty A \xrightarrow{MS^\infty \eta^M} MS^\infty MA \xrightarrow{MS^\infty f} MS^\infty MB \xrightarrow{M\lambda} MMS^\infty B \xrightarrow{\mu^M} MS^\infty B\Big)$$

Since λ is a distributive law between monads, $\lfloor S^\infty \rfloor$ is also a monad. Its structure is given by:

$$\mu^{\lfloor S^\infty \rfloor} = \Big(MS^\infty S^\infty \xrightarrow{M\mu^\infty} MS^\infty\Big)$$
$$\eta^{\lfloor S^\infty \rfloor} = \Big(M \xrightarrow{M\eta^\infty} MS^\infty\Big)$$

Theorem 5.19. *Let M be a monad and $\langle S, \overline{\mu}^S \rangle$ be an M-module. For a morphism $f : A \nrightarrow B$ in* Kleisli$'$*, we define the following morphism in $\mathbb{C}$:*

$$\widehat{f} = \Big(SA \xrightarrow{S\eta^M} SMA \xrightarrow{Sf} SMB \xrightarrow{\overline{\mu}^S} SB\Big)$$

We define an endofunctor G on Kleisli$'$ *as follows:*

$$GA = SA$$
$$G(f : A \nrightarrow B) = \Big(MSA \xrightarrow{M\widehat{f}} MSB\Big)$$

The monad $\lfloor S^\infty \rfloor$ is a cim with respect to the module $G\lfloor S^\infty \rfloor$. Moreover, it is the free cim generated by G in Kleisli$'$.

Proof. We show that for all objects A in KLEISLI$'$, the object $\lfloor S^\infty \rfloor A$ is given by the carrier of the final $(G(\text{-})\oplus A)$-coalgebra, and that the action of $\lfloor S^\infty \rfloor$ on morphisms and its monadic structure is as in Adámek *et al.*'s construction (Theorem 2.71). Hence, by Theorem 2.74, it is the free cim generated by G.

Recall that by ξ_A we denote the action of the final $(S(\text{-}) + A)$-coalgebra. We first show that for an object A, the tuple

$$\langle MS^\infty A,\ M\xi_A : MS^\infty A \to M(SS^\infty A + A)\rangle$$

is the final $(G(\text{-}) \oplus A)$-coalgebra. Let $f : X \nrightarrow GX \oplus A$ be a morphism in KLEISLI$'$. It has the type $MX \to M(SX + A)$ in $\mathbb{C}$. Consider $\widehat{f} : SX \to S(SX + A)$. It is an $S((\text{-}) + A)$-coalgebra, so it induces a unique homomorphism $SX \to \nu Y.S(Y + A)$. By the rolling rule, it holds that $\nu Y.S(Y + A) \cong SS^\infty A$, which means that there exists a unique morphism $[\![\widehat{f}]\!]$ such that the following diagram commutes in $\mathbb{C}$:

$$\begin{array}{ccc} SS^\infty A & \xrightarrow{S\xi_A} & S(SS^\infty A + A) \\ {\scriptstyle [\![\widehat{f}]\!]}\uparrow & & \uparrow{\scriptstyle S([\![\widehat{f}]\!] + \mathsf{id})} \\ SX & \xrightarrow{\widehat{f}} & S(SX + A) \end{array} \tag{5.8}$$

We define a morphism:

$$[\![f]\!] = \left(MX \xrightarrow{f} M(SX + A) \xrightarrow{M([\![\widehat{f}]\!] + \mathsf{id})} M(SS^\infty A + A) \xrightarrow{M\xi_A^{-1}} MS^\infty A \right)$$

The diagram (5.8) entails that $[\![\widehat{f}]\!] = \widehat{[\![f]\!]}$. Substituting $\widehat{[\![f]\!]}$ for $[\![\widehat{f}]\!]$ in the definition of $[\![f]\!]$ and composing with $M\xi_A^{-1}$ yields that the following diagram commutes in $\mathbb{C}$:

$$\begin{array}{ccc} MS^\infty A & \xrightarrow{M\xi_A} & M(SS^\infty A + A) \\ {\scriptstyle [\![f]\!]}\uparrow & & \uparrow{\scriptstyle M(\widehat{[\![f]\!]} + \mathsf{id})} \\ MX & \xrightarrow{f} & M(SX + A) \end{array} \tag{5.9}$$

All the components of the above diagram are morphisms in KLEISLI$'$, so, applying the appropriate definitions, the diagram corresponds to the following composition in KLEISLI$'$:

$$\begin{array}{ccc} \lfloor S^\infty \rfloor A & \xrightarrow{M\xi_A} & G\lfloor S^\infty \rfloor A \oplus A \\ {\scriptstyle [\![f]\!]}\uparrow & & \uparrow{\scriptstyle G[\![f]\!] \oplus \mathsf{id}} \\ X & \xrightarrow{f} & GX \oplus A \end{array} \tag{5.10}$$

This means that $[\![f]\!]$ is indeed a homomorphism $\langle X, f\rangle \to \langle \lfloor S^\infty \rfloor A, M\xi_A\rangle$.

We need to show that $[\![f]\!]$ is a unique morphism in Kleisli$'$ that makes the diagram (5.10) commute. Assume that r is another morphism of this type, which means that the following diagram commutes in Kleisli$'$:

$$\begin{array}{ccc} \lfloor S^\infty \rfloor A & \xrightarrow{M\xi_A} & G\lfloor S^\infty \rfloor A \oplus A \\ {\scriptstyle r}\uparrow & & \uparrow{\scriptstyle Gr \oplus \mathsf{id}} \\ X & \xrightarrow{f} & GX \oplus A \end{array} \tag{5.11}$$

Unfolding the appropriate definitions and applying Lambek's lemma, the diagram above corresponds to the following diagram in $\mathbb{C}$:

$$\begin{array}{ccccccc} MS^\infty A & & \xleftarrow{\quad M\xi_A^{-1} \quad} & & & & M(SS^\infty A + A) \\ {\scriptstyle r}\uparrow & & & & & & \uparrow{\scriptstyle M(\overline{\mu} + \mathsf{id})} \\ MX & \xrightarrow{f} & M(SX + A) & \xrightarrow{M(S\eta + \mathsf{id})} & M(SMX + A) & \xrightarrow{M(Sr + \mathsf{id})} & M(SMS^\infty + A) \end{array} \tag{5.12}$$

As a result, the following diagram commutes in $\mathbb{C}$:

$$\begin{array}{ccccccc} SX & \xrightarrow{S\eta} & SMX & \xrightarrow{Sr} & SMS^\infty A & \xrightarrow{\overline{\mu}} & SS^\infty A \\ {\scriptstyle S\eta}\downarrow & & & & \uparrow{\scriptstyle SM(\xi_A^{-1})} & & \uparrow{\scriptstyle S\xi_A^{-1}} \\ SMX & & \overset{Sf}{\searrow} & & SM(SS^\infty A + A) & & \\ {\scriptstyle Sf}\downarrow & & & & \uparrow{\scriptstyle SM(\overline{\mu} + \mathsf{id})} & & \\ SM(SX + A) & \xrightarrow{SM(S\eta + \mathsf{id})} & SM(SMX + A) & \xrightarrow{SM(Sr + \mathsf{id})} & SM(SMS^\infty A + A) & & \\ {\scriptstyle \overline{\mu}}\downarrow & & & & & & \\ S(SX + A) & \xrightarrow{S(S\eta + \mathsf{id})} & S(SMX + A) & \xrightarrow{S(Sr + \mathsf{id})} & S(SMS^\infty A + A) & \xrightarrow{S(\overline{\mu} + \mathsf{id})} & S(SS^\infty A + A) \end{array} \tag{5.13}$$

(The triangle in the top-left corner is trivial. The hexagon next to it is the S-image of the diagram (5.12). The remaining part is the naturality of $\overline{\mu}$.) The top edge of the diagram (5.13) above is equal to $\widehat{r}$, the left-most edge is equal to $\widehat{f}$, while the bottom edge is equal to $S(\widehat{r} + \mathsf{id})$. Thus, the diagram (5.13) is the following equality:

$$\widehat{r} = \left(SX \xrightarrow{\widehat{f}} S(SX + A) \xrightarrow{S(\widehat{r} + \mathsf{id})} S(SS^\infty A + A) \xrightarrow{S\xi_A^{-1}} SS^\infty A\right)$$

This means that $\widehat{r}$ is an $S((\text{-}) + A)$-coalgebra morphism from $\langle SX, \widehat{f}\rangle$ to $\langle SS^\infty A, S\xi_A\rangle$. Due to the uniqueness of $[\![\widehat{f}]\!]$ in the diagram (5.8), we obtain that $\widehat{r} = [\![\widehat{f}]\!]$. Together

with the fact that r and $[\![\widehat{f}]\!]$ are coalgebra morphisms, this gives us that $r = M\xi_A^{-1} \cdot M(\widehat{r} + \mathsf{id}) \cdot f = M\xi_A^{-1} \cdot M([\![\widehat{f}]\!] + \mathsf{id}) \cdot f = [\![f]\!]$.

The above means that $\langle \lfloor S^\infty \rfloor A,\ M\xi_A \rangle$ is the final $(G(\text{-}) \oplus A)$-coalgebra in KLEISLI$'$, which entails that G is iteratable (Definition 2.70) and it admits a free cim (Theorem 2.74). We denote this free cim by $G^\Diamond$. To complete the proof, we need to show that $\lfloor S^\infty \rfloor$ and $G^\Diamond$ agree as functors and monads.

Obviously, $\lfloor S^\infty \rfloor$ is equal to $G^\Diamond$ on objects. For morphisms, let $\mathbb{C}$ be a category and $F : \mathbb{C} \to \mathbb{C}$ an endofunctor that generates the free cim F^∞ with the final coalgebra map denoted as ζ. For a morphism $f : A \to B$ in $\mathbb{C}$, its F^∞-image is given by a unique morphism $k : F^\infty A \to F^\infty B$ such that the following condition holds:

$$k = \left(F^\infty A \xrightarrow{\zeta_A} FF^\infty A + A \xrightarrow{Fk+f} FF^\infty B + B \xrightarrow{\zeta_B^{-1}} F^\infty B \right)$$

Let $f : A \nrightarrow B$ be a morphism in KLEISLI$'$. In the case of $G^\Diamond$ the condition above specialises to the following composition in $\mathbb{C}$:

$$k = \Big(MS^\infty A \xrightarrow{M\xi_A} M(SS^\infty A + A) \xrightarrow{M(\widehat{k} + (f \cdot \eta^M))} M(SS^\infty B + MB) \xrightarrow{\mathsf{flatr}} M(SS^\infty B + B) \xrightarrow{M\xi_B^{-1}} MS^\infty B \Big)$$

We check this condition for $\lfloor S^\infty \rfloor f$. First, unfolding one step of the distributive law λ, we obtain the following:

$$\lambda = \left(S^\infty M \xrightarrow{\xi} SS^\infty M + M \xrightarrow{S\lambda + \mathsf{id}} SMS^\infty + M \xrightarrow{\overline{\mu}^S + \mathsf{id}} SS^\infty + M \xrightarrow{\eta^M} M(SS^\infty + M) \xrightarrow{\mathsf{flatr}} M(SS^\infty + \mathsf{Id}) \xrightarrow{M\xi^{-1}} MS^\infty \right)$$

For a KLEISLI$'$ morphism $f : A \nrightarrow B$, we first calculate:

$$\begin{aligned}
& \widehat{\lfloor S^\infty \rfloor A f} \\
= \quad & \text{(definition)} \\
& \overline{\mu}^S_B \cdot S \lfloor S^\infty \rfloor f \cdot S\eta^M_{S^\infty A} \\
= \quad & \text{(definition)} \\
& \overline{\mu}^S_B \cdot S\mu^M_{S^\infty B} \cdot SM\lambda_B \cdot SMS^\infty f \cdot SMS^\infty \eta^M_A \cdot S\eta^M_{S^\infty A} \\
= \quad & \text{(naturality and monad laws)} \\
& \overline{\mu}^S_B \cdot S\lambda_B \cdot SS^\infty f \cdot SS^\infty \eta^M_A
\end{aligned}$$

We use this to prove the desired property of $\lfloor S^\infty \rfloor$:

$$
\begin{aligned}
& \lfloor S^\infty \rfloor f \\
= \quad & \text{(lifting)} \\
& \mu^M \cdot M\lambda \cdot MS^\infty(f \cdot \eta^M) \\
= \quad & \text{(one step of } \lambda \text{)} \\
& \mu^M \cdot M^2\xi^{-1} \cdot M\mathsf{flatr} \cdot M\eta^M \cdot M(\overline{\mu}^S + \mathsf{id}) \cdot M(S\lambda + \mathsf{id}) \cdot M\xi \cdot MS^\infty(f \cdot \eta^M) \\
= \quad & \text{(naturality and monad laws)} \\
& M\xi^{-1} \cdot \mathsf{flatr} \cdot M(\overline{\mu}^S + \mathsf{id}) \cdot M(S\lambda + \mathsf{id}) \cdot M\xi \cdot MS^\infty(f \cdot \eta^M) \\
= \quad & \text{(naturality)} \\
& M\xi^{-1} \cdot \mathsf{flatr} \cdot M(\overline{\mu}^S + \mathsf{id}) \cdot M(S\lambda + \mathsf{id}) \cdot M(SS^\infty(f \cdot \eta^M) + (f \cdot \eta^M)) \cdot M\xi \\
= \quad & \text{((the above))} \\
& M\xi_B^{-1} \cdot \mathsf{flatr} \cdot M(\widehat{\lfloor S^\infty \rfloor f} + (f \cdot \eta^M)) \cdot M\xi_A
\end{aligned}
$$

To show that the monadic structure of $\lfloor S^\infty \rfloor$ given by the lifting and $G^\Diamond$ as the free cim are equal, we first notice that the multiplication of $\mu^\infty_A : F^\infty F^\infty A \to F^\infty A$ can be described as a unique morphism with the following property:

$$
\begin{aligned}
\mu^\infty_A = \Big(F^\infty F^\infty A \xrightarrow{\zeta} FF^\infty F^\infty A + F^\infty A \xrightarrow{F\mu^\infty_A + \zeta} FF^\infty A + FF^\infty A + A \\
\xrightarrow{[\mathsf{id},\mathsf{id}]+\mathsf{id}} FF^\infty A + A \xrightarrow{\xi^{-1}} F^\infty A \Big)
\end{aligned}
\tag{5.14}
$$

We also notice the following:

$$
\widehat{\mu^{\lfloor S^\infty \rfloor}} = S\mu^\infty \tag{5.15}
$$

It can be proven as follows:

$$
\begin{aligned}
& \widehat{\mu^{\lfloor S^\infty \rfloor}_A} \\
= \quad & \text{(def. of } \mu^{\lfloor S^\infty \rfloor} \text{)} \\
& \widehat{M\mu^\infty_A} \\
= \quad & \text{(def. of } \widehat{(\text{-})} \text{)} \\
& \overline{\mu}^S_{S^\infty A} \cdot SM\mu^\infty_A \cdot S\eta^M_{S^\infty S^\infty A} \\
= \quad & \text{(naturality of } \overline{\mu} \text{)} \\
& S\mu^\infty_A \cdot \overline{\mu}^S_{S^\infty S^\infty A} \cdot S\eta^M_{S^\infty S^\infty A} \\
= \quad & \text{(modules)} \\
& S\mu^\infty_A
\end{aligned}
$$

To show that $\mu^{\lfloor S^\infty \rfloor}$ satisfies the property (5.14), we calculate:

$$\begin{aligned}
& \mu_A^{\lfloor S^\infty \rfloor} \\
= \quad & \text{(def.)} \\
& M\mu_A^\infty \\
= \quad & \text{(property (5.14) for } \mu^\infty \text{)} \\
& M\xi_A^{-1} \cdot M([\mathsf{id}, \mathsf{id}] + \mathsf{id}) \cdot M[S\mu_A^\infty, \xi_A] \cdot M\xi_{S^\infty A} \\
= \quad & \text{((5.15))} \\
& M\xi_A^{-1} \cdot M([\mathsf{id}, \mathsf{id}] + \mathsf{id}) \cdot M[\widehat{\mu^{\lfloor S^\infty \rfloor}}, \xi_A] \cdot M\xi_{S^\infty A}
\end{aligned}$$

Applying the appropriate definitions, the ultimate morphism is equal to the following composition in Kleisli′:

$$\lfloor S^\infty \rfloor \lfloor S^\infty \rfloor A \xrightarrow{M\xi_A} G\lfloor S^\infty \rfloor \lfloor S^\infty \rfloor A \oplus \lfloor S^\infty \rfloor A \xrightarrow{G\mu_A^{\lfloor S^\infty \rfloor} \oplus M\xi_A} G\lfloor S^\infty \rfloor A \oplus G\lfloor S^\infty \rfloor A \oplus A \xrightarrow{[\![\mathsf{id}, \mathsf{id}]\!] \oplus \mathsf{id}} G\lfloor S^\infty \rfloor A \oplus A \xrightarrow{M\xi_A^{-1}} \lfloor S^\infty \rfloor A$$

Therefore, $\mu^{\lfloor S^\infty \rfloor}$ satisfies the property (5.14), hence $\mu^{\lfloor S^\infty \rfloor} = \mu^{G^\Diamond}$. The case $\eta^{\lfloor S^\infty \rfloor} = \eta^{G^\Diamond}$ follows trivially by unfolding of the definitions. □

5.2.4 Resumptions over monads

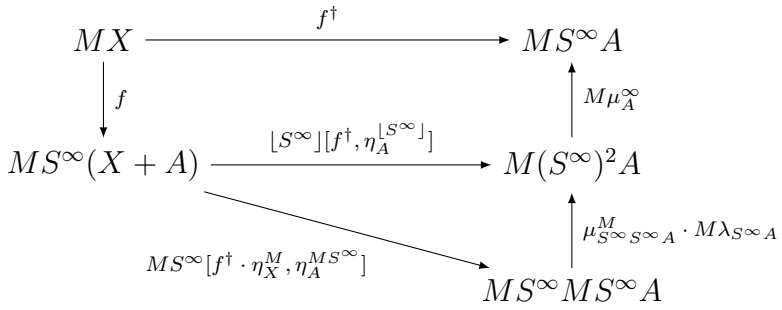
Lemma 5.20. *Let S be an iteratable module over a monad M, and let $f : X \to \lfloor S^\infty \rfloor (X + A)$ be a guarded equation morphism in the monad $\lfloor S^\infty \rfloor$ with a solution $f^\dagger : X \to \lfloor S^\infty \rfloor A$. Then, $f^\dagger$ seen as a $\mathbb{C}$-morphism is the unique morphism in $\mathbb{C}$ such that the following diagram commutes:*

$$\begin{array}{ccc}
MX & \xrightarrow{f^\dagger} & MS^\infty A \\
{\scriptstyle f}\downarrow & & \uparrow{\scriptstyle \mu_A^{MS^\infty}} \\
MS^\infty(X + A) & \xrightarrow{MS^\infty[f^\dagger \cdot \eta_A^M,\ \eta_A^{MS^\infty}]} & MS^\infty MS^\infty A
\end{array}$$

Proof. By Lemma 5.19 there exists a unique morphism $f^\dagger : X \to \lfloor S^\infty \rfloor A$ such that the following diagram commutes in Kleisli′:

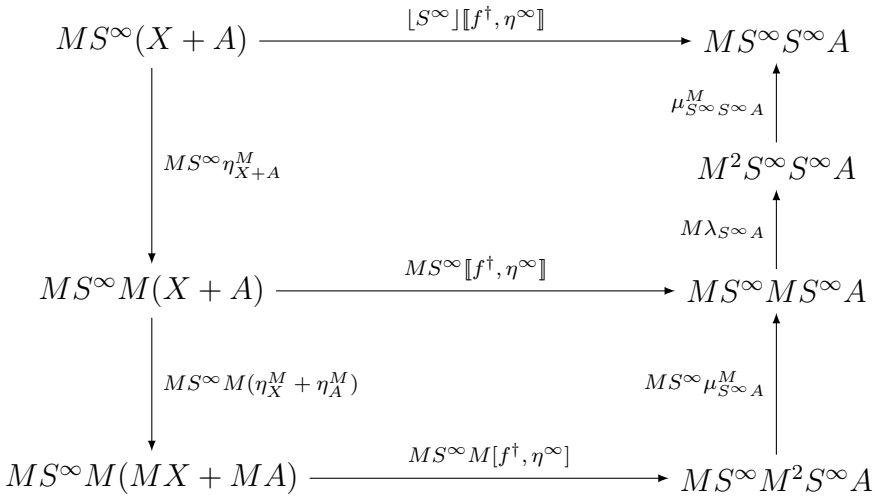
$$\begin{array}{ccc}
X & \xrightarrow{f^\dagger} & \lfloor S^\infty \rfloor A \\
{\scriptstyle f}\downarrow & & \uparrow{\scriptstyle \mu_A^{\lfloor S^\infty \rfloor}} \\
\lfloor S^\infty \rfloor (X \oplus A) & \xrightarrow{\lfloor S^\infty \rfloor [\![f^\dagger, \eta_A^{\lfloor S^\infty \rfloor}]\!]} & \lfloor S^\infty \rfloor^2 A
\end{array}$$

We obtain that $f^\dagger$ is the unique Kleisli′ morphism such that the following diagram, in which the right-most edge is equal to μ^{MS^∞}, commutes in $\mathbb{C}$:



(5.16)

(The triangle on top follows from the appropriate definitions and coproducts in Kleisli′. The fact that the bottom triangle commutes can be read from the following commutative diagram.)

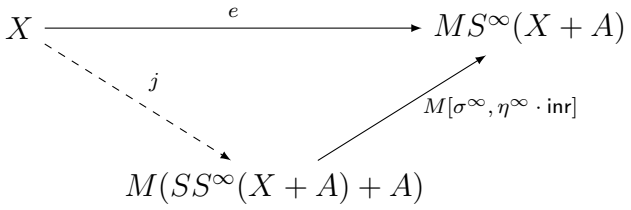


(The upper part follows from the definition of the action $\lfloor S^\infty \rfloor$ on morphism, the lower part follows from the definition of the coproduct mediator in Kleisli′.)

For uniqueness, let $r : MX \to MS^\infty A$ be a $\mathbb{C}$-morphism. Assume that the diagram (5.16) with r substituted for $f^\dagger$ commutes. We notice that for any morphism $g : B \to C$, the morphism Mg is a Kleisli′ morphism, and $\mu_B : MMB \to MB$ is also a Kleisli′ morphism. Thus, since r factors as shown in the diagram (5.16), it is a composition of Kleisli′-morphisms. This means that r is itself a Kleisli′-morphism, and, since $f^\dagger$ is the only morphism in Kleisli′ that makes the diagram (5.16) commute, we obtain $r = f^\dagger$. □

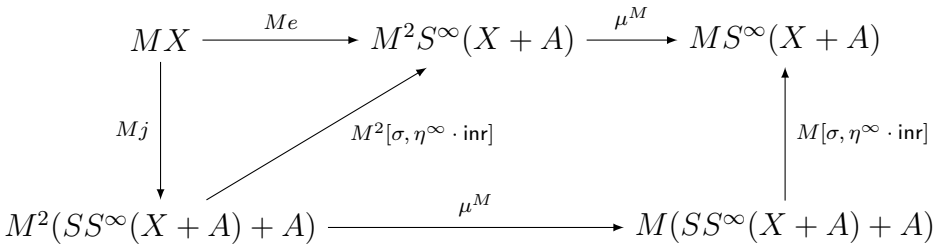
Lemma 5.21. *Let M be a monad, and let S be an iteratable M-module. Consider the monad MS^∞ as defined in Theorem 5.4. Let e be a morphism that factors as*

follows:



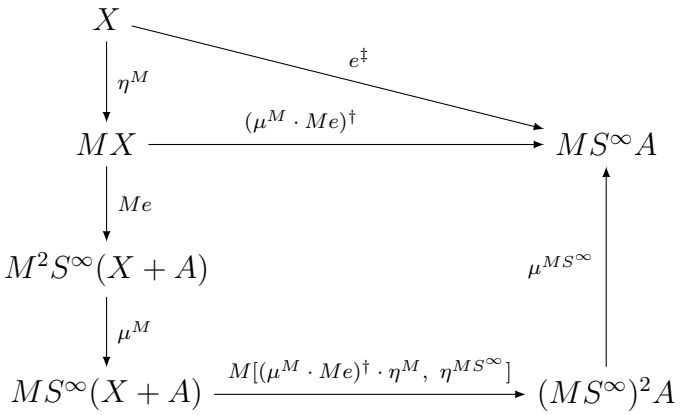
Then, e has a unique solution, which we denote $e^\ddagger$.

Proof. The morphism $\mu^M \cdot Me$ is guarded in $\lfloor S^\infty \rfloor$. For this, consider the following diagram:



(The triangle on the left is the M-image of the guardedness condition of e. The square on the right is the naturality of μ^M.)

Thus, the morphism $\mu^M \cdot Me$ has a unique solution $(\mu^M \cdot Me)^\dagger$. We define $e^\ddagger = (\mu^M \cdot Me)^\dagger \cdot \eta^M$. The morphism $e^\ddagger$ has the desired solution property, which can be seen from the following diagram (the left-most edge is equal to e):



(The triangle on top is the definition of $e^\ddagger$. The part below it follows from Lemma 5.20.)

For uniqueness, let $r : X \to MS^\infty A$ be a solution. Note that $r = \mu^M \cdot Mr \cdot \eta^M_X$. Substituting $\mu^M \cdot Mr$ for $(\mu^M \cdot Me)^\dagger$ in the diagram above yields that the bottom pentagon precomposed with η^M_X commutes. One can show that η^M is epic for Kleisli′-morphisms (that is, $f \cdot \eta = g \cdot \eta$ implies $f = g$, for Kleisli′-morphisms f and g). This means that, since $\mu^M \cdot Mr$ is a Kleisli′-morphism (as a composition of two

KLEISLI′-morphisms), the pentagon commutes, hence $\mu^M \cdot Mr$ is a solution of $\mu^M \cdot Me$ in the sense of Lemma 5.20. This means that $r = \mu^M \cdot Mr \cdot \eta^M_X = (\mu^M \cdot Me)^\dagger \cdot \eta^M_X = e^\ddagger$. □

Note that the property above is stronger than complete iterativity, since a wider class of equation morphisms have unique solutions. A guarded equation morphism of the form

$$X \dashrightarrow MSS^\infty(X+A)+A \xrightarrow{[M\sigma^\infty, \eta^\infty \cdot \mathsf{inr}]} MS^\infty(X+A)$$

can be seen in the form given by Lemma 5.21 as

$$X \dashrightarrow MSS^\infty(X+A)+A \xrightarrow{[M\mathsf{inl}, \eta^M \cdot \mathsf{inr}]} M(SS^\infty(X+A)+A) \xrightarrow{[\sigma^\infty, \eta^\infty \cdot \mathsf{inr}]} MS^\infty(X+A).$$

Note that this more specific result can be obtained easier, using Theorem 3.1 (the composition theorem) together with Theorem 5.19. We need the generality of Lemma 5.21 in order to show horizontal complete iterativity in the next subsection.

5.2.5 Resumptions over cims

Theorem 5.22. *Let M be a cim and S be an iteratable M-module. The monad MS^∞ (see Theorem 5.4) is completely iterative with respect to the module $\overline{M}S^\infty + MSS^\infty$.*

Proof. In this proof, for brevity, we denote S^∞ as T and its module SS^∞ as $\overline{T}$. Let $e : X \to MT(X+A)$ be a guarded equation morphism. We construct a solution of e, which we call $e^\sharp$, and then we show that it is indeed a solution and that it is unique.

The morphism e is guarded, which means that it factors as follows for some morphism j:

$$X \xrightarrow{j} \overline{M}T(X+A) + M\overline{T}(X+A) + A \xrightarrow{[\sigma^M,\, M\sigma^T,\, \eta^{MT} \cdot \mathsf{inr}]} MT(X+A)$$

By $\mathsf{sym}_{A,B} : A+B \to B+A$ we denote the canonical braiding of coproducts. We define the following natural transformation. It is actually a natural isomorphism, since it is a composition of two isomorphisms:

$$t_{X,A} = \Big(T(X+A) \xrightarrow{\xi_{X+A}} \overline{T}(X+A) + X + A \xrightarrow{\mathsf{sym}+\mathsf{id}} X + \overline{T}(X+A) + A\Big)$$

Consider the following morphism:

$$\begin{aligned} f = \Big(X &\xrightarrow{j} \overline{M}T(X+A) + M\overline{T}(X+A) + A \\ &\xrightarrow{\overline{M}t+\mathsf{id}+\mathsf{id}} \overline{M}(X + \overline{T}(X+A) + A) + M\overline{T}(X+A) + A \\ &\xrightarrow{[\sigma^M,\ M(\mathsf{inl}\cdot\mathsf{inr}),\ \eta^M \cdot \mathsf{inr}]} M(X + \overline{T}(X+A) + A)\Big) \end{aligned}$$

The morphism f is a guarded equation morphism in the monad M in the sense of Corollary 5.12 (we treat $\overline{T}(X+A)+A$ as the object of parameters). Therefore, there exists a unique solution $f^\dagger : X \to M(\overline{T}(X+A)+A)$. We define the following natural transformation:

$$k_{X,A} = [\sigma^T, \eta^T \cdot \mathsf{inr}] : \overline{T}(X+A)+A \to T(X+A)$$

The morphism $Mk \cdot f^\dagger : X \to MT(X+A)$ is a guarded equation morphism in the sense of Lemma 5.21, so it has a unique solution

$$e^\sharp = (Mk \cdot f^\dagger)^\ddagger : X \to MTA$$

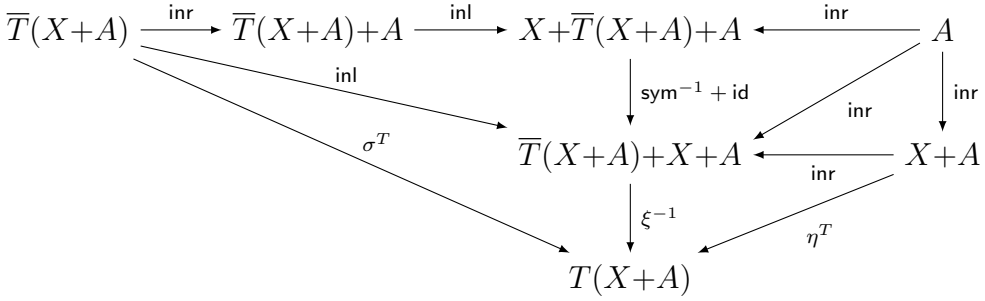
We show that $e^\sharp$ is a unique solution of e in the monad MT. First, we need the following equality:

$$Mt^{-1}_{X,A} \cdot f = e \tag{5.17}$$

To prove it, we calculate:

$$\begin{aligned}
& Mt^{-1}_{X,A} \cdot f \\
= \quad & \text{(def. of } f \text{)} \\
& Mt^{-1} \cdot [\sigma^M,\ M(\mathsf{inl} \cdot \mathsf{inr}),\ \eta^M \cdot \mathsf{inr}] \cdot (\overline{M}t + \mathsf{id} + \mathsf{id}) \cdot j \\
= \quad & \text{(coproducts)} \\
& [Mt^{-1} \cdot \sigma^M \cdot \overline{M}t,\ Mt^{-1} \cdot M(\mathsf{inl} \cdot \mathsf{inr}),\ Mt^{-1} \cdot \eta^M \cdot \mathsf{inr}] \cdot j \\
= \quad & \text{(naturality of } \sigma^M \text{)} \\
& [Mt^{-1} \cdot Mt \cdot \sigma^M,\ Mt^{-1} \cdot M(\mathsf{inl} \cdot \mathsf{inr}),\ Mt^{-1} \cdot \eta^M \cdot \mathsf{inr}] \cdot j \\
= \quad & \text{((see below))} \\
& [\sigma^M,\ M\sigma^T,\ \eta^{MT} \cdot \mathsf{inr}] \cdot j
\end{aligned}$$

The first component of the coproduct follows from the fact that t is an isomorphism. The second and third components follow from the fact that the following diagram commutes (by properties of coproducts and definitions of σ^T and η^T), in which the vertical 'spine' is t^{-1}:



We also need the following equality for all objects Y and B:

$$[\eta^T_{Y+B} \cdot \mathsf{inl},\ k_{Y,B}] = t^{-1}_{Y,B} \tag{5.18}$$

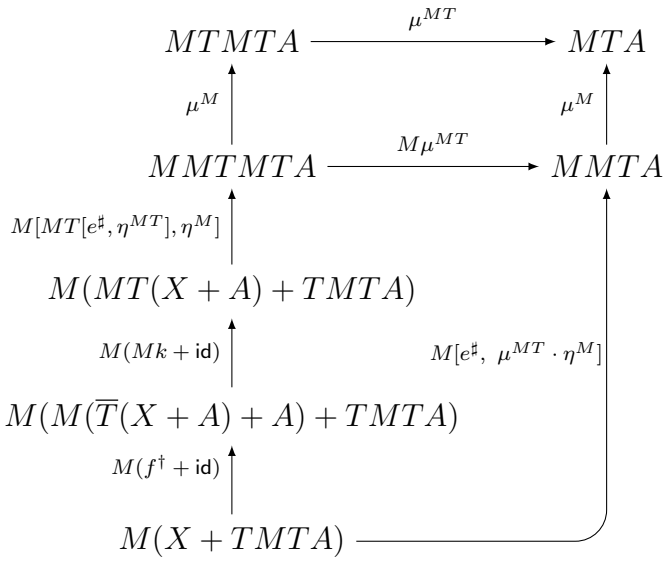
We calculate:

$$\begin{aligned}
& [\eta^T_{Y+B} \cdot \mathsf{inl},\ k_{Y,B}] \\
= \quad & (\text{ def. of } k \ ) \\
& [\eta^T_{Y+B} \cdot \mathsf{inl},\ \sigma^T_{Y+B},\ \eta^T_{Y+B} \cdot \mathsf{inr}] \\
= \quad & (\text{ coproducts }) \\
& [\sigma^T_{Y+B},\ \eta^T_{Y+B} \cdot \mathsf{inl},\ \eta^T_{Y+B} \cdot \mathsf{inr}] \cdot (\mathsf{sym}^{-1} + \mathsf{id}) \\
= \quad & (\text{ definitions of } \sigma^T \text{ and } \eta^T \ ) \\
& [\xi^{-1}_{Y+B} \cdot \mathsf{inl},\ \xi^{-1}_{Y+B} \cdot \mathsf{inr} \cdot \mathsf{inl},\ \xi^{-1}_{Y+B} \cdot \mathsf{inr} \cdot \mathsf{inr}] \cdot (\mathsf{sym}^{-1} + \mathsf{id}) \\
= \quad & (\text{ coproducts }) \\
& [\xi^{-1}_{Y+B} \cdot \mathsf{inl},\ \xi^{-1}_{Y+B} \cdot \mathsf{inr}] \cdot (\mathsf{sym}^{-1} + \mathsf{id}) \\
= \quad & (\text{ coproducts }) \\
& \xi^{-1}_{Y+B} \cdot [\mathsf{inl},\ \mathsf{inr}] \cdot (\mathsf{sym}^{-1} + \mathsf{id}) \\
= \quad & (\text{ coproducts }) \\
& \xi^{-1}_{Y+B} \cdot (\mathsf{sym}^{-1} + \mathsf{id}) \\
= \quad & (\text{ def. of } t^{-1} \ ) \\
& t^{-1}_{Y+B}
\end{aligned}$$

As the next step of the proof, we consider a series of commuting diagrams:

$$\begin{array}{ccccccc}
X & \xrightarrow{f^\dagger} & M(\overline{T}(X+A)+A) & \xrightarrow{Mk} & MT(X+A) & \xrightarrow{MT[e^\sharp,\eta^{MT}]} & MTMTA \\
 & & & & & & \uparrow{\scriptstyle \mu^M} \\
 & & & & & & MMTMTA \\
 & & & & & & \uparrow{\scriptstyle M[MT[e^\sharp,\eta^{MT}],\eta^M]} \\
\downarrow{\scriptstyle f} & & & & & & M(MT(X+A)+TMTA) \\
 & & & & & & \uparrow{\scriptstyle M(Mk+\mathsf{id})} \\
 & & & & & & M(M(\overline{T}(X+A)+A)+TMTA) \\
 & & & & & & \uparrow{\scriptstyle M(f^\dagger+\mathsf{id})} \\
M(X+\overline{T}(X+A)+A) & & \xrightarrow{M(\mathsf{id}+k)} & & M(X+T(X+A)) & \xrightarrow{M(\mathsf{id}+T[e^\sharp,\eta^{MT}])} & M(X+TMTA)
\end{array} \tag{5.19}$$

(This diagram states that the morphism $MT[e^\sharp, \eta^{MT}] \cdot Mk \cdot f^\dagger$ is a solution of $M(\mathsf{id} + T[e^\sharp, \eta^{MT}]) \cdot M(\mathsf{id} + k) \cdot f$, which follows from Lemma 5.10.)

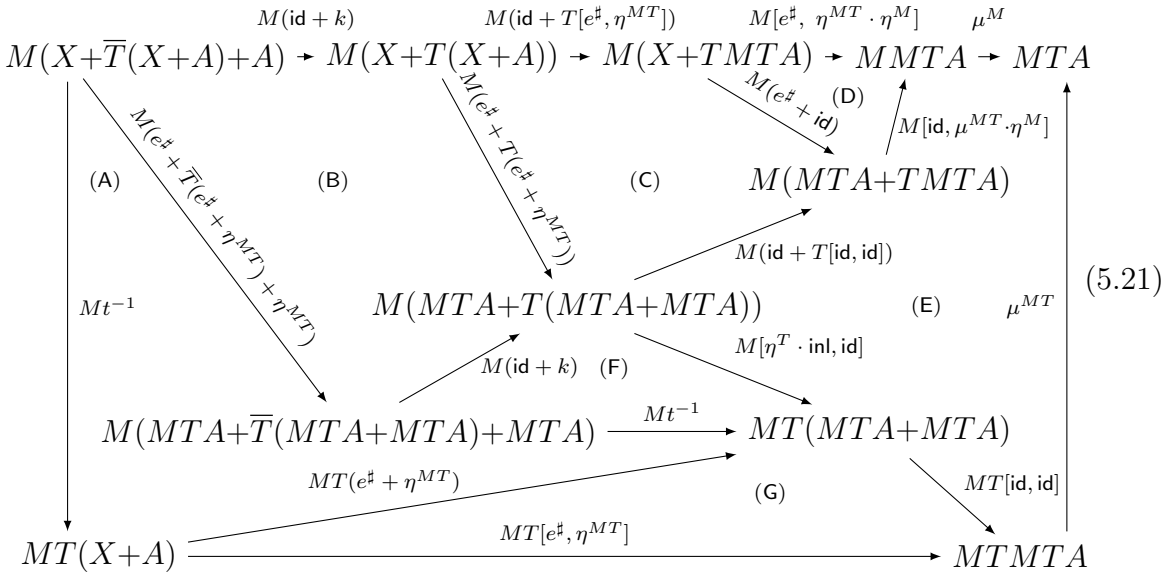


(5.20)

(The square on top follows from the fact that μ^{MT} is given by a distributive law. In detail:

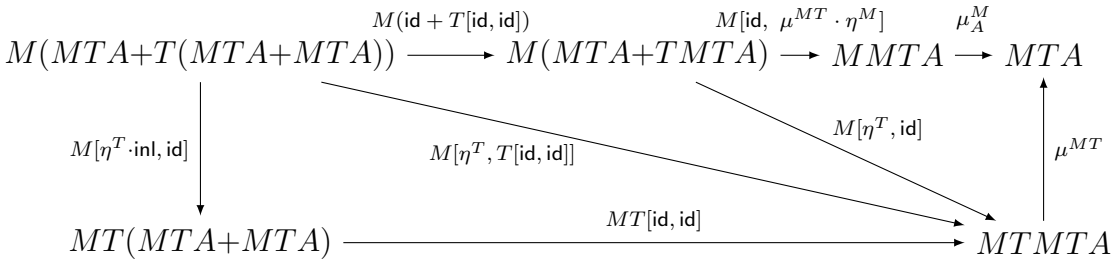
$$
\begin{aligned}
& \mu^{MT}_A \cdot \mu^M_{TMTA} \\
= \quad & (\text{ def. of } \mu^{MT}) \\
& \mu^M_{TA} \cdot MM\mu^T_A \cdot M\lambda_{TA} \cdot \mu^M_{TMTA} \\
= \quad & (\text{ naturality of } \mu^{MT}) \\
& \mu^M_{TA} \cdot \mu^M_{MTA} \cdot MMM\mu^T_A \cdot MM\lambda_{TA} \\
= \quad & (\text{ monad laws }) \\
& \mu^M_{TA} \cdot M\mu^M_{TA} \cdot MMM\mu^T_A \cdot MM\lambda_{TA} \\
= \quad & (\text{ def. of } \mu^{MT}) \\
& \mu^M_{TA} \cdot M\mu^{MT}_A
\end{aligned}
$$

For the bottom part of the diagram, we consider each part of the coproduct $X + TMTA$ separately. The left-hand side is the solution property of $e^\sharp$ in the sense of Lemma 5.21. The right-hand side is trivial.)



(5.21)

((A) naturality of t^{-1}, (B) naturality of k, (C) and (D) properties of coproducts, (E) the diagram below, (F) the equality (5.18), (G) properties of coproducts.)



(The triangle on the left follows from properties of coproducts an the naturality of η^T, the triangle in the middle follows from properties of coproduct, while the square

in the top-right corner is shown by the following calculation.)

$$\begin{aligned}
&\mu^M_{MTA} \cdot M[\mathsf{id},\ \mu^{MT}_A \cdot \eta^M_{TMTA}] \\
=\quad & \text{(monad laws)} \\
&\mu^M_{MTA} \cdot M[\mu^{MT}_A \cdot \eta^{MT}_{MTA},\ \mu^{MT}_A \cdot \eta^M_{TMTA}] \\
=\quad & \text{(coproducts)} \\
&\mu^M_{MTA} \cdot M\mu^{MT}_A \cdot M[\eta^{MT}_{MTA},\ \eta^M_{TMTA}] \\
=\quad & \text{(calculation below the diagram (5.20))} \\
&\mu^{MT}_A \cdot \mu^M_{TMTA} \cdot M[\eta^{MT}_{MTA},\ \eta^M_{TMTA}] \\
=\quad & \text{(def. of } \eta^{MT} \text{)} \\
&\mu^{MT}_A \cdot \mu^M_{TMTA} \cdot M[\eta^M_{TMTA} \cdot \eta^T_{MTA},\ \eta^M_{TMTA}] \\
=\quad & \text{(coproducts)} \\
&\mu^{MT}_A \cdot \mu^M_{TMTA} \cdot M\eta^M_{TMTA} \cdot M[\eta^T_{MTA}, \mathsf{id}] \\
=\quad & \text{(monad laws)} \\
&\mu^{MT}_A \cdot M[\eta^T_{MTA}, \mathsf{id}]
\end{aligned}$$

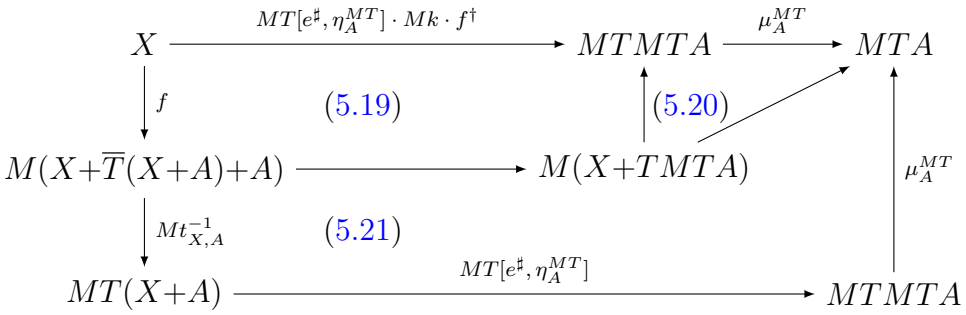
Now, to show that $e^\sharp$ is indeed a solution of e in the monad MS^∞, we first notice that

$$e^\sharp = (Mk \cdot f^\dagger)^\ddagger = \mu^{MT}_A \cdot MT[(Mk \cdot f^\dagger)^\ddagger,\ \eta^{MT}_A] \cdot Mk \cdot f^\dagger = \mu^{MT}_A \cdot MT[e^\sharp,\ \eta^{MT}_A] \cdot Mk \cdot f^\dagger$$

(The second equality is the solution property of $(\text{-})^\ddagger$.) Thus, the desired solution property can be stated as

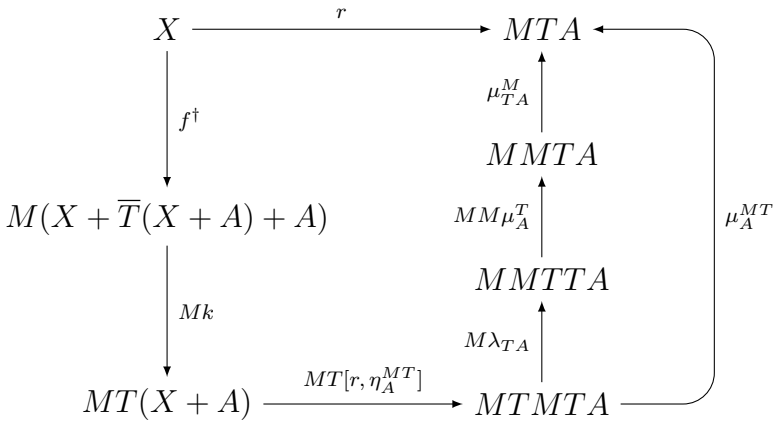
$$\mu^{MT}_A \cdot MT[e^\sharp,\ \eta^{MT}_A] \cdot Mk \cdot f^\dagger = \mu^{MT}_A \cdot MT[e^\sharp, \eta^{MT}_A] \cdot e$$

It can be obtained by factoring e as $Mt^{-1}_{X,A} \cdot f$ (as given by the equality (5.17)) and putting together the diagrams (5.19), (5.20), and (5.21):



(5.22)

For uniqueness, assume that a morphism $r : X \to MTA$ is a solution of e. We show that the following diagram commutes, which gives us that $r = (Mk \cdot f^\dagger)^\ddagger = e^\sharp$ from the uniqueness of solutions given by Lemma 5.21.



The longer path of the perimeter can be rewritten using Lemma 5.10 (the functoriality of solutions) as follows:

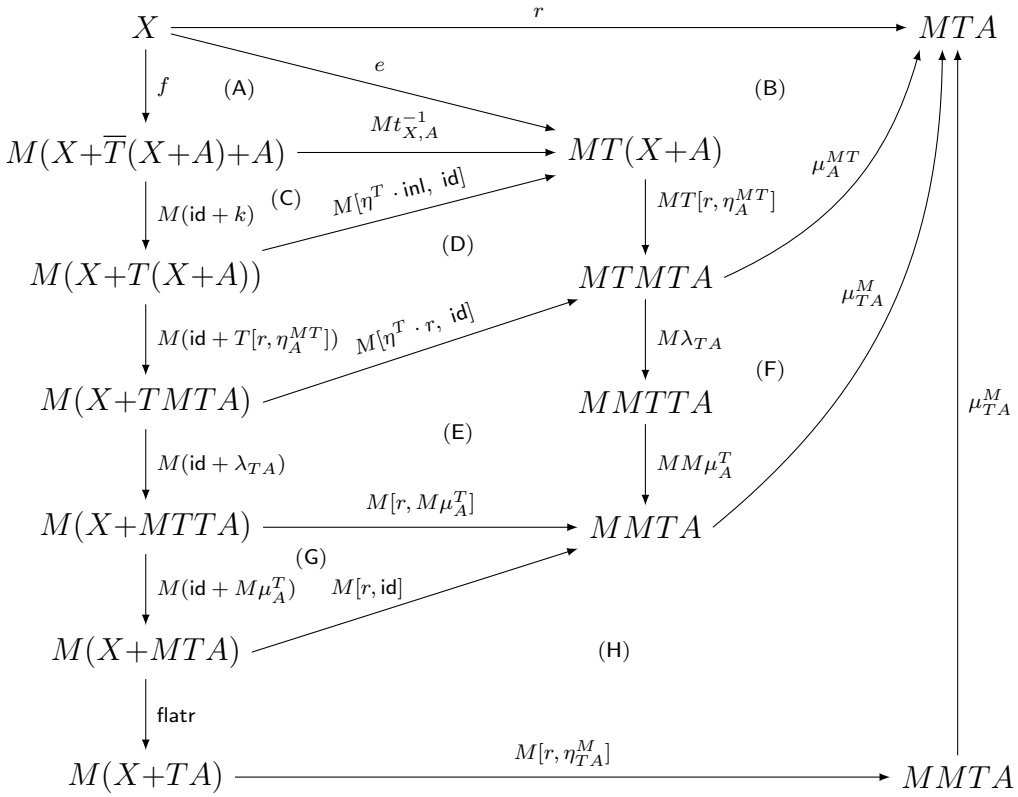
$$\mu^M_{TA} \cdot (M(\mathsf{id} + M\mu^T_A) \cdot M(\mathsf{id} + \lambda_{TA}) \cdot M(\mathsf{id} + T[r, \eta^{MT}_A]) \cdot M(\mathsf{id} + k) \cdot f)^\dagger \qquad (5.23)$$

To show that r and (5.23) are equal, we use Lemma 5.13 ($(\mathsf{flatr} \cdot e)^\dagger = \mu \cdot e^\dagger$). In this case, it states that the morphism (5.23) is the unique solution of the following morphism:

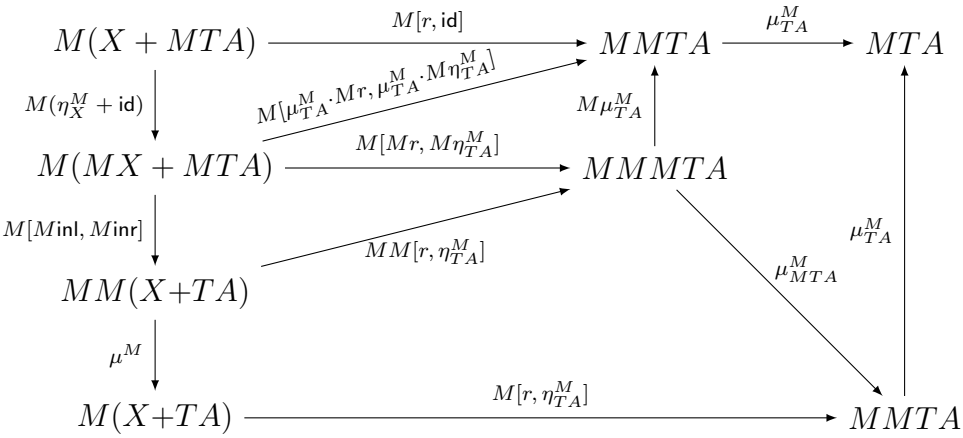
$$\mathsf{flatr}_{X,TA} \cdot M(\mathsf{id} + M\mu^T_A) \cdot M(\mathsf{id} + \lambda_{TA}) \cdot M(\mathsf{id} + T[r, \eta^{MT}_A]) \cdot M(\mathsf{id} + k) \cdot f \qquad (5.24)$$

It is left to show that r is also a solution of (5.24). It can be read from the following

diagram:



((A) the equality (5.17), (B) solution property of r, (C) the equality (5.18), (D) coproducts, (E) componentwise: properties of distributive law, (F) definition of μ^{MT}, (G) coproducts, (H) the diagram below.)



(Every part commutes due to monad laws, properties of coproducts, or naturality of η^M and μ^M.) □

5.3 Resumption algebras

Definition 5.23. For an M-module S on the category $\mathbb{C}$, a *resumption algebra* is a tuple $\langle A,\ f : MA \to A,\ g : SA \to A,\ (\text{-})^\dagger \rangle$, where:

1. The morphism f is an Eilenberg-Moore algebra.
2. The triple $\langle A, g, (\text{-})^\dagger \rangle$ is an Elgot algebra.
3. Coherence: for a flat equation morphism $e : X \to SX + MA$ it holds that $(f \circledcirc e)^\dagger = |e|^\dagger \cdot (f \circledcirc e)$, where $|e|$ is as given in Definition 5.2.

A morphism between two resumption algebras $\langle A, f^A, g^A, (\text{-})^\dagger \rangle$ and $\langle B, f^B, g^B, (\text{-})^\ddagger \rangle$ is a $\mathbb{C}$-morphism $h : A \to B$ such that it is a homomorphism $f^A \to f^B$, and that it is solution-preserving, that is, for any flat equation morphism $e : X \to SX + A$, it holds that $h \cdot e^\dagger = (h \circledcirc e)^\ddagger$.

We denote the category of resumption algebras for the M-module S and resumption algebra morphisms as $\textsc{ResAlg}(M, S)$.

Theorem 5.24. *If an M-module S as a functor is iterative, the obvious forgetful functor $U^{\mathrm{ResAlg}} : \textsc{ResAlg} \to \mathbb{C}$ has a left adjoint given as follows:*

$$
\begin{aligned}
F^{\mathrm{ResAlg}} A &= \langle MS^\infty A, f, g, (\text{-})^\dagger \rangle, \text{ where} \\
f &= MMS^\infty A \xrightarrow{\mu^M_{S^\infty A}} MS^\infty A \\
g &= SMS^\infty A \xrightarrow{\overline{\mu}^S_{S^\infty A}} SS^\infty A \xrightarrow{\tau_A} S^\infty A \xrightarrow{\mu^M_{S^\infty A}} MS^\infty A \\
F^{\mathrm{ResAlg}} h &= MS^\infty h
\end{aligned}
$$

The monad induced by this adjunction is equal to MS^∞.

Proof. This proof is analogous to the proof of Theorem 4.20. Consider the adjunction $F^{\mathrm{Elg}} \dashv U^{\mathrm{Elg}} : \mathbb{C} \rightharpoonup \textsc{Elgot}(S)$ and the Eilenberg-Moore adjunction of the monad $\lceil M \rceil$ from Lemma 5.3 $F^{\mathrm{EM}} \dashv U^{\mathrm{EM}} : \textsc{Elgot}(S) \rightharpoonup \textsc{MAlg}(\lceil M \rceil)$. Objects of $\textsc{MAlg}(\lceil M \rceil)$ are tuples of the following shape:

$$
\langle \langle A, g : SA \to A, (\text{-})^\dagger \rangle, f : \lceil M \rceil \langle A, g, (\text{-})^\dagger \rangle \to \langle A, g, (\text{-})^\dagger \rangle \rangle,
$$

such that the following conditions are satisfied:

1. The triple $\langle A, g, (\text{-})^\dagger \rangle$ is an Elgot algebra.

2. The morphism f is an Eilenberg-Moore algebra action. Since the monadic structure of the lifting $\lceil M \rceil$ is given by the monadic structure of M, this condition is equivalent to saying that f is an Eilenberg-Moore algebra for M.

3. The morphism f is a morphism between two Elgot algebras, hence it preserves solutions: for a flat equation morphism $e : X \to SX + MA$, it holds that $(f \circledcirc e)^\dagger = f \cdot e^\ddagger$, where $e^\ddagger$ denotes the solution of e in $\lceil M \rceil \langle A, g, (\text{-})^\dagger \rangle$.

The expression $f \cdot e^\ddagger$ from the condition (3) can be expanded as follows (compare Definition 5.2):

$$X \xrightarrow{e} SX + MA \xrightarrow{\mathsf{inl}+\mathsf{id}} SX + A + MA \xrightarrow{|e|^\dagger + \mathsf{id}} A + MA \xrightarrow{[\eta^M_A, \mathsf{id}]} MA \xrightarrow{f} A$$

We calculate:

$$\begin{aligned}
& f \cdot [\eta^M_A, \mathsf{id}] \cdot (|e|^\dagger + \mathsf{id}) \cdot (\mathsf{inl} + \mathsf{id}) \cdot e \\
&= [f \cdot \eta^M_A,\ f] \cdot (|e|^\dagger + \mathsf{id}) \cdot (\mathsf{inl} + \mathsf{id}) \cdot e && \text{(coproducts)} \\
&= [\mathsf{id}, f] \cdot (|e|^\dagger + \mathsf{id}) \cdot (\mathsf{inl} + \mathsf{id}) \cdot e && (f \text{ is Eilenberg–Moore}) \\
&= [\mathsf{id}, \mathsf{id}] \cdot (|e|^\dagger + \mathsf{id}) \cdot (\mathsf{inl} + \mathsf{id}) \cdot (\mathsf{id} + f) \cdot e && \text{(coproducts)} \\
&= [\mathsf{id}, \mathsf{id}] \cdot ([g, \mathsf{id}] + \mathsf{id}) \cdot (S|e|^\dagger + \mathsf{id} + \mathsf{id}) \cdot (|e| + \mathsf{id}) \cdot (\mathsf{inl} + \mathsf{id}) \cdot (\mathsf{id} + f) \cdot e && \text{(solution)} \\
&= [\mathsf{id}, \mathsf{id}] \cdot ([g, \mathsf{id}] + \mathsf{id}) \cdot (S|e|^\dagger + \mathsf{id} + \mathsf{id}) \cdot (\mathsf{flatr} + \mathsf{id} + \mathsf{id}) && \text{(def. of |-|)} \\
&\qquad \cdot (Se + \mathsf{id} + \mathsf{id}) \cdot (\mathsf{inl} + \mathsf{id}) \cdot (\mathsf{id} + f) \cdot e \\
&= [\mathsf{id}, \mathsf{id}] \cdot (g + \mathsf{id}) \cdot (S|e|^\dagger + \mathsf{id}) \cdot (\mathsf{flatr} + \mathsf{id}) \cdot (Se + \mathsf{id}) \cdot (\mathsf{id} + f) \cdot e && \text{(coproducts)} \\
&= [\mathsf{id}, \mathsf{id}] \cdot (g + \mathsf{id}) \cdot (S|e|^\dagger + \mathsf{id}) \cdot |e| \cdot (\mathsf{id} + f) \cdot e && \text{(def. of |-|)} \\
&= |e|^\dagger \cdot (\mathsf{id} + f) \cdot e && \text{(solution)} \\
&= |e|^\dagger \cdot (f \circledcirc e) && \text{(def. of } \circledcirc\text{)}
\end{aligned}$$

This makes the condition (3) equivalent to the coherence condition of resumption algebras. This means that the category $\textsc{MAlg}(\lceil M \rceil)$ is equal to $\textsc{ResAlg}(M, S)$. One can easily see that the composition $U^{\mathrm{Elg}} U^{\mathrm{EM}}$ is equal to U^{ResAlg}, hence $F^{\mathrm{EM}} F^{\mathrm{Elg}}$ is the left adjoint to U^{ResAlg}. Simple unfolding of the definitions of F^{Elg} and F^{EM} gives us that the direct definition of their composition is as specified in the theorem. □

Theorem 5.25. *For an M-module S, if S^∞ exists, the functor U^{ResAlg} is strictly monadic. This entails that the category* $\textsc{ResAlg}(M, S)$ *is isomorphic to* $\textsc{MAlg}(MS^\infty)$.

Proof. Similarly to the proof of Theorem 4.21, we use Beck's theorem. Theorem 5.24 already states that U^{ResAlg} is a right adjoint, so it is left to show that it creates coequalisers for those parallel h_0, h_1 in $\textsc{ResAlg}(M, S)$ for which $U^{\mathrm{ResAlg}}h_0$ and $U^{\mathrm{ResAlg}}h_1$ have a split coequaliser in $\mathbb{C}$.

Let $h_0, h_1 : \langle A, f^A, g^A, (\text{-})^\dagger\rangle \to \langle B, f^B, g^B, (\text{-})^\ddagger\rangle$ be such a pair. Let c be a split coequaliser of $U^{\mathrm{ResAlg}}h_0$ and $U^{\mathrm{ResAlg}}h_1$. In other words, there exist morphisms s and t such that the following diagram commutes and in which the two horizontal compositions are the identities:

$$\begin{array}{ccccc} B & \xrightarrow{t} & A & \xrightarrow{h_0} & B \\ \downarrow{\scriptstyle c} & & \downarrow{\scriptstyle h_1} & & \downarrow{\scriptstyle c} \\ C & \xrightarrow{s} & B & \xrightarrow{c} & C \end{array} \tag{5.25}$$

We need to show that there exist a unique resumption algebra $\langle C,\ f^C : MC \to C,\ g^C : SC \to C,\ (\text{-})^\S\rangle$ such that $c : \langle B, f^B, g^B, (\text{-})^\ddagger\rangle \to \langle C, f^C, g^C, (\text{-})^\S\rangle$ is a homomorphism and a coequaliser of h_0 and h_1. From the monadicity of the forgetful functors $U^{\mathrm{EM}} : \textsc{MAlg}(M) \to \mathbb{C}$ and $U^{\mathrm{Elg}} : \textsc{Elgot}(S) \to \mathbb{C}$, we obtain that there exist a unique Eilenberg-Moore algebra $\langle C, f^C\rangle$ and a unique Elgot algebra $\langle C, g^C, (\text{-})^\S\rangle$ with the required properties. Their actions are defined respectively as:

$$f^C = \left(MC \xrightarrow{Ms} MB \xrightarrow{f^B} B \xrightarrow{c} C\right)$$
$$g^C = \left(SC \xrightarrow{Ss} SB \xrightarrow{g^B} B \xrightarrow{c} C\right)$$

For a flat equation morphism $e : X \to SX + C$, its solution is defined as $e^\S = c \cdot (s \circledcirc e)^\ddagger$ (see Adámek *et al.* [15], proof of Theorem 4.8).

Thus, it is enough to show that the two existing algebras satisfy the coherence condition. We need to show that for a flat equation morphism $e : X \to SX + MC$, it holds that $(f^C \circledcirc e)^\S = |e|^\S \cdot (f^C \circledcirc e)$. The morphism c preserves solutions, which means that for all flat equation morphisms $k : X \to SX + B$, it holds that:

$$c \cdot k^\ddagger = (c \circledcirc k)^\S = c \cdot (s \circledcirc c \circledcirc k)^\ddagger \tag{5.26}$$

Using the fact that $c \cdot s = \mathsf{id}$, it requires simple diagram chasing (see Section A.7 in the appendix) to prove that the following holds for a flat equation morphism $e : X \to SX + MC$.

$$(s \circledcirc |e|) \cdot (\mathsf{id}_{SX} + c) = (S(\mathsf{id}_{SX} + c) + \mathsf{id}_B) \cdot (s \circledcirc c \circledcirc |Ms \circledcirc e|) \tag{5.27}$$

Thus, by functoriality, we obtain:

$$(s \circledcirc |e|)^{\ddagger} \cdot (\mathsf{id}_{SX} + c) = (s \circledcirc c \circledcirc |Ms \circledcirc e|)^{\ddagger} \tag{5.28}$$

Together with (5.26), the above gives us the following:

$$c \cdot (s \circledcirc |e|)^{\ddagger} \cdot (\mathsf{id}_{SX} + c) = c \cdot (s \circledcirc c \circledcirc |Ms \circledcirc e|)^{\ddagger} = c \cdot |Ms \circledcirc e|^{\ddagger} \tag{5.29}$$

We calculate that the coherence condition holds:

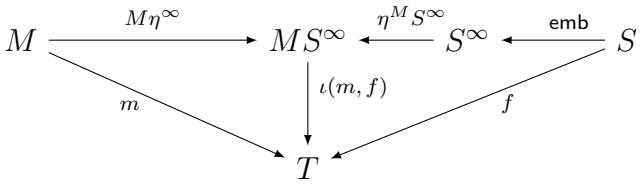
$$\begin{aligned}
(f^C \circledcirc e)^{\S} &= c \cdot (s \circledcirc f^C \circledcirc e)^{\ddagger} && \text{(def. of (-)}^{\S}\text{)} \\
&= c \cdot (s \circledcirc c \circledcirc f^B \circledcirc Ms \circledcirc e)^{\ddagger} && \text{(def. of } f^C\text{)} \\
&= c \cdot (f^B \circledcirc Ms \circledcirc e)^{\ddagger} && \text{((5.26))} \\
&= c \cdot |Ms \circledcirc e|^{\ddagger} \cdot (f^B \circledcirc Ms \circledcirc e) && \text{(coherence)} \\
&= c \cdot (s \circledcirc |e|)^{\ddagger} \cdot (\mathsf{id}_{SX} + c) \cdot (f^B \circledcirc Ms \circledcirc e) && \text{((5.29))} \\
&= c \cdot (s \circledcirc |e|)^{\ddagger} \cdot (c \circledcirc f^B \circledcirc Ms \circledcirc e) && \text{(def. of } \circledcirc\text{)} \\
&= c \cdot (s \circledcirc |e|)^{\ddagger} \cdot (f^C \circledcirc e) && \text{(def. of } f^C\text{)} \\
&= |e|^{\S} \cdot (f^C \circledcirc e) && \text{(def. of (-)}^{\S}\text{)}
\end{aligned}$$

□

5.4 Universal property

In this section, we characterise the monad MS^{∞} by a universal property, similar to the one given in Theorem 4.28 for 'ordinary' monads. This universal property can be stated as follows:

Theorem 5.26. *Let M be a monad, S be an M-module such that S as an enfunctor is iteratable, T be a cim, and $\langle m, f\rangle : \langle M, S\rangle \to U^{\mathrm{Mod}}T$ be a module morphism such that f is ideal. Then, there exists a unique monad morphism $\iota(m, f) : MS^{\infty} \to T$ such that the following diagram commutes:*

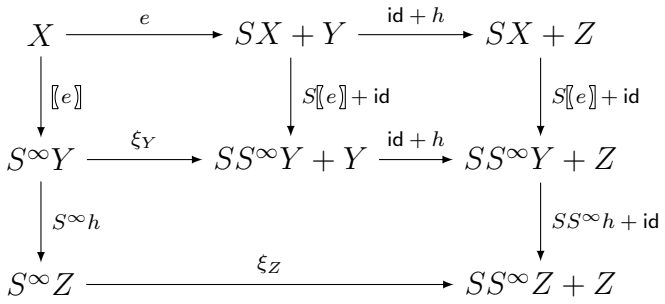


(5.30)

In this section, we give a proof of this theorem, which is similar in structure to the proof of Theorem 4.28, but is considerably more complex, because of the use of Elgot algebras. First, we need a couple of auxiliary lemmata:

Lemma 5.27. *Let S be iteratable. For morphisms $e : X \to SX + Y$ and $h : Y \to Z$, it is the case that $S^\infty h \cdot (\!|e|\!) = (\!|h \circledcirc e|\!) : X \to S^\infty Z$.*

Proof. The following diagram commutes:



(Top-left corner: $(\!|e|\!)$ is a coalgebra homomorphism; top-right corner: $+$ is a bifunctor; bottom: naturality of ξ.)

This means that $S^\infty h \cdot (\!|e|\!)$ is a coalgebra homomorphism from $\langle X,\ (\mathsf{id} + h) \cdot e\rangle = \langle X,\ h \circledcirc e\rangle$ to $\langle S^\infty Z, \xi_Z\rangle$. From the finality of the latter, we obtain $S^\infty h \cdot (\!|e|\!) = (\!|h \circledcirc e|\!)$. □

Lemma 5.28. *Let T be a cim, $e : X \to T(X + A)$ be an equation morphism that factors through a morphism $g : X \to \overline{T}(X + A) + TA$ as in Lemma 5.11 (a), and $a : TA \to A$ be an Eilenberg–Moore algebra. Let*

$$\widehat{g} = \left(X \xrightarrow{g} \overline{T}(X + A) + TA \xrightarrow{\mathsf{id}+a} \overline{T}(X + A) + A \xrightarrow{[\sigma^T,\ \eta^T \cdot \mathsf{inr}]} T(X + A)\right)$$

Then, it is the case that $a \cdot e^\dagger = a \cdot \widehat{g}^\dagger$.

Proof. First, consider the following morphism:

$$p = \left(X \xrightarrow{g} \overline{T}(X + A) + TA \xrightarrow{[\sigma^T \cdot \overline{T}(\mathsf{id}+\eta^T),\ \eta^T \cdot \mathsf{inr}]} T(X + TA)\right)$$

It is the case that $\mathsf{flatr} \cdot p = e$, hence, by Lemma 5.13, it is the case that $e^\dagger = \mu^T \cdot p^\dagger$. Thus, we obtain:

$$\begin{aligned}
a \cdot e^\dagger &= a \cdot \mu^T \cdot p^\dagger && \text{(Lemma 5.13)}\\
&= a \cdot Ta \cdot p^\dagger && (a \text{ is Eilenberg–Moore})\\
&= a \cdot (T(\mathsf{id} + a) \cdot p)^\dagger && \text{(Lemma 5.10)}\\
&= a \cdot \widehat{g}^\dagger && (\text{(see below)})
\end{aligned}$$

Now, it is enough to show that $T(\mathsf{id} + a) \cdot p = \widehat{g}$. We calculate:

$$\begin{aligned}
T(\mathsf{id} + a) \cdot p &= T(\mathsf{id} + a) \cdot [\sigma^T \cdot \overline{T}(\mathsf{id} + \eta^T),\ \eta^T \cdot \mathsf{inr}] \cdot g && \text{(def. of } p\text{)} \\
&= [T(\mathsf{id} + a) \cdot \sigma^T \cdot \overline{T}(\mathsf{id} + \eta^T),\ T(\mathsf{id} + a) \cdot \eta^T \cdot \mathsf{inr}] \cdot g && \text{(coproducts)} \\
&= [\sigma^T \cdot \overline{T}(\mathsf{id} + a) \cdot \overline{T}(\mathsf{id} + \eta^T),\ \eta^T \cdot (\mathsf{id} + a) \cdot \mathsf{inr}] \cdot g && \text{(naturality)} \\
&= [\sigma^T \cdot \overline{T}(\mathsf{id} + a) \cdot \overline{T}(\mathsf{id} + \eta^T),\ \eta^T \cdot \mathsf{inr} \cdot a] \cdot g && \text{(coproducts)} \\
&= [\sigma^T,\ \eta^T \cdot \mathsf{inr} \cdot a] \cdot g && (a \text{ is Eilenberg–Moore}) \\
&= [\sigma^T,\ \eta^T \cdot \mathsf{inr}] \cdot (\mathsf{id} + a) \cdot g && \text{(coproducts)} \\
&= \widehat{g} && \text{(def. of } \widehat{g}\text{)}
\end{aligned}$$

□

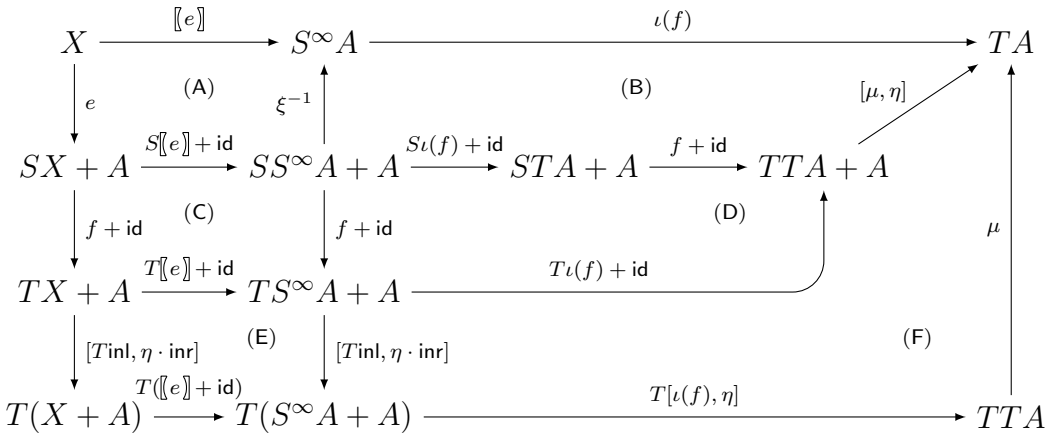
Lemma 5.29. *Let T be a cim, S be an iteratable endofunctor, $e : X \to SX + A$ be a morphism, and $f : S \to T$ be an ideal natural transformation. Consider the morphism*

$$k = \left(X \xrightarrow{e} SX + A \xrightarrow{f + \mathsf{id}} TX + A \xrightarrow{[T\mathsf{inl},\ \eta^T \cdot \mathsf{inr}]} T(X + A)\right)$$

Then, it is the case that

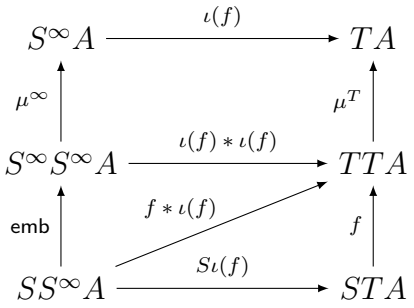
$$\left(X \xrightarrow{k^\dagger} TA\right) = \left(X \xrightarrow{[\![e]\!]} S^\infty A \xrightarrow{\iota(f)} TA\right)$$

Proof. We show that $\iota(f) \cdot [\![e]\!]$ is a solution of k, that is, that the following diagram commutes:



((A) final coalgebra, (B) (see below), (C) and (D) naturality of f, (E) and (F) coproducts.)

It is left to verify that the part (B) commutes. We express ξ^{-1} as $[\mu^\infty \cdot \mathsf{emb},\ \eta^\infty] : SS^\infty A + A \to S^\infty A$, and proceed component-wisely. The left component:



(Top-down: $\iota(f)$ is a monad morphism, the universal property of S^∞, natural transformations.)

The diagram for the right component is simply $\eta^T = \iota(f) \cdot \eta^\infty$, which follows from the fact that $\iota(f)$ is a monad morphism. □

Proof of Theorem 5.26. The structure of this proof closely follows the structure for the equivalent result for inductive resumptions (Theorem 4.28). We use the isomorphism Δ (Lemma 2.26) to translate the problem into the language of functors between Eilenberg-Moore categories. First, we define the two following forgetful functors:

$$U^M : \textsc{ResAlg}(M,S) \to \textsc{MAlg}(M) \qquad U^S : \textsc{ResAlg}(M,S) \to \textsc{Elgot}(S)$$
$$U^M\langle A, f, g, (\text{-})^\dagger\rangle = \langle A, f\rangle \qquad U^S\langle A, f, g, (\text{-})^\dagger\rangle = \langle A, g, (\text{-})^\dagger\rangle$$
$$U^M k = k \qquad U^S k = k$$

Both $\textsc{ResAlg}(M,S)$ and $\textsc{Elgot}(S)$ are monadic, so the functors U^M and U^S can be seen as (carrier-preserving) functors between Eilenberg–Moore categories, that is, morphisms in EM. Their Δ^{-1}-images are as follows, where $\beta_A : SS^\infty A \to S^\infty A$ is the natural family of the actions of the free completely iterative S-algebras:

$$\begin{aligned}\Delta^{-1}U^M &= \left(M \xrightarrow{M\eta^{MS^\infty}} MMS^\infty \xrightarrow{p} MS^\infty\right), \text{ where } U^M F^{\text{ResAlg}} A = \langle MS^\infty A, p_A\rangle \\ &= \left(M \xrightarrow{M\eta^{MS^\infty}} MMS^\infty \xrightarrow{\mu^M S^\infty} MS^\infty\right) = \left(M \xrightarrow{M\eta^\infty} MS^\infty\right)\end{aligned}$$

$$\begin{aligned}\Delta^{-1}U^S &= \left(S \xrightarrow{S\eta^{MS^\infty}} SMS^\infty \xrightarrow{q} MS^\infty\right), \text{ where } U^S F^{\text{CIA}} A = \langle S^\infty A, q_A\rangle \\ &= \left(S \xrightarrow{S\eta^{MS^\infty}} SMS^\infty \xrightarrow{\overline{\mu}^S S^\infty} SS^\infty \xrightarrow{\beta} S^\infty \xrightarrow{\eta^M S^\infty} MS^\infty\right) \\ &= \left(S \xrightarrow{S\eta^\infty} SS^\infty \xrightarrow{\beta} S^\infty \xrightarrow{\eta^M S^\infty} MS^\infty\right) = \left(S \xrightarrow{\mathsf{emb}} S^\infty \xrightarrow{\eta^M S^\infty} MS^\infty\right)\end{aligned}$$

Since $\textsc{Malg}(MS^\infty) \cong \textsc{ResAlg}(M,S)$ and $\textsc{Malg}(S^\infty) \cong \textsc{Elgot}(S)$, it is enough to prove that there exits a unique carrier-preserving functor Φ that makes the following diagram commute:

$$\begin{array}{ccccc} \textsc{Malg}(M) & \xleftarrow{U^M} & \textsc{ResAlg}(M,S) & \xrightarrow{U^S} & \textsc{Elgot}(S) \\ & {\scriptstyle \Delta m} \searrow & \uparrow {\scriptstyle \Phi} & \nearrow {\scriptstyle \Delta f} & \\ & & \textsc{Malg}(T) & & \end{array} \tag{5.31}$$

Then, we take $\iota(m,f) = \Delta^{-1}\Phi$. We proceed in a number of steps: (i) we define Φ and show that it is a functor; (ii) we show that the diagram (5.31) commutes; (iii) we show that Φ is a unique such functor that makes the diagram (5.31) commute.

(i) First, we define Φ and show that it is indeed a functor. Let $\langle A,\ a : TA \to A\rangle$ be an Eilenberg–Moore algebra. Since the composition of a monad morphism with an Eilenberg–Moore algebra is again an Eilenberg–Moore algebra (Lemma 2.26), the pair $\langle A,\ S^\infty A \xrightarrow{\iota(f)_A} TA \xrightarrow{a} A\rangle$ is an Eilenberg–Moore S^∞-algebra. The corresponding Elgot S-algebra is given as:

$$\langle A,\ SA \xrightarrow{\mathsf{emb}_A} S^\infty A \xrightarrow{\iota(f)_A} TA \xrightarrow{a} A,\ (\text{-})^\ddagger\rangle = \langle A,\ SA \xrightarrow{f_A} TA \xrightarrow{a} A,\ (\text{-})^\ddagger\rangle$$

where the solution for an equation morphism $e : X \to SX + A$ is given as follows:

$$e^\ddagger = \left(X \xrightarrow{[\![e]\!]} S^\infty A \xrightarrow{\iota(f)_A} TA \xrightarrow{a} A\right)$$

We define:

$$\begin{aligned} &\Phi\langle A,\ a : TA \to A\rangle = \langle A,\ MA \xrightarrow{m_A} TA \xrightarrow{a} A,\ SA \xrightarrow{f_A} TA \xrightarrow{a} A,\ (\text{-})^\ddagger\rangle \\ &\Phi h = h \end{aligned}$$

First, we need to check that for a cim T and an Eilenberg–Moore algebra $\langle A,\ a : TA \to A\rangle$, the tuple $\Phi\langle A, a\rangle$ is a resumption algebra. By Lemma 2.26, $a \cdot m_A$ is an Eilenberg–Moore algebra. So, it is left to verify the coherence condition for resumption algebras, that is, $((a \cdot m_A) \circledcirc e)^\ddagger = |e|^\ddagger \cdot ((a \cdot m_A) \circledcirc e)$. We calculate. The left-hand side (where $(\text{-})^\dagger$ denotes solutions of guarded equation morphisms in the monad T):

$$\begin{aligned} ((a \cdot m_A) \circledcirc e)^\ddagger &= a \cdot \iota(f)_A \cdot [\![(a \cdot m_A) \circledcirc e]\!] && \text{(def. of } (\text{-})^\ddagger) \\ &= a \cdot ([T\mathsf{inl},\ \eta^T \cdot \mathsf{inr}] \cdot (f + (a \cdot m_A)) \cdot e)^\dagger && \text{(Lemma 5.29)} \\ &= a \cdot ([T\mathsf{inl}, T\mathsf{inr}] \cdot (f + m_A) \cdot e)^\dagger && \text{(Lemma 5.28)} \end{aligned}$$

The right-hand side:

$$
\begin{aligned}
&|e|^{\ddagger}\cdot((a\cdot m_A)\circledcirc e)\\
&\quad= a\cdot[\![\,|e|\,]\!]\cdot(\mathsf{id}+(a\cdot m_A))\cdot e && \text{(def. of (-)}^{\ddagger}\text{)}\\
&\quad= a\cdot([T\mathsf{inl},\ \eta^T\cdot\mathsf{inr}]\cdot(f+\mathsf{id})\cdot|e|)^{\dagger}\cdot(\mathsf{id}+(a\cdot m_A))\cdot e && \text{(Lemma 5.29)}\\
&\quad= a\cdot\mu^T\cdot T[([T\mathsf{inl},\ \eta^T\cdot\mathsf{inr}]\cdot(f+\mathsf{id})\cdot|e|)^{\dagger},\ \eta^T] && \text{(solution)}\\
&\qquad\cdot[T\mathsf{inl},\ \eta^T\cdot\mathsf{inr}]\cdot(f+\mathsf{id})\cdot|e|\cdot(\mathsf{id}+(a\cdot m_A))\cdot e\\
&\quad= a\cdot\mu^T\cdot T[([T\mathsf{inl},\ \eta^T\cdot\mathsf{inr}]\cdot(f+\mathsf{id})\cdot|e|)^{\dagger},\ \eta^T] && \text{(def. of |-|)}\\
&\qquad\cdot[T\mathsf{inl},\ \eta^T\cdot\mathsf{inr}]\cdot(f+\mathsf{id})\cdot(\mathsf{flatr}+\mathsf{id})\cdot(Se+\mathsf{id})\cdot(\mathsf{id}+(a\cdot m_A))\cdot e\\
&\quad= a\cdot\mu^T\cdot[T([T\mathsf{inl},\ \eta^T\cdot\mathsf{inr}]\cdot(f+\mathsf{id})\cdot|e|)^{\dagger},\ T\eta^T\cdot\eta^T] && \text{(coproducts)}\\
&\qquad\cdot(f+\mathsf{id})\cdot(\mathsf{flatr}+\mathsf{id})\cdot(Se+\mathsf{id})\cdot(\mathsf{id}+(a\cdot m_A))\cdot e\\
&\quad= [a\cdot\mu^T\cdot T([T\mathsf{inl},\ \eta^T\cdot\mathsf{inr}]\cdot(f+\mathsf{id})\cdot|e|)^{\dagger},\ a\cdot\mu^T\cdot T\eta^T\cdot\eta^T] && \text{(coproducts)}\\
&\qquad\cdot(f+\mathsf{id})\cdot(\mathsf{flatr}+\mathsf{id})\cdot(Se+\mathsf{id})\cdot(\mathsf{id}+(a\cdot m_A))\cdot e\\
&\quad= [a\cdot\mu^T\cdot T([T\mathsf{inl},\ \eta^T\cdot\mathsf{inr}]\cdot(f+\mathsf{id})\cdot|e|)^{\dagger},\ a\cdot\eta^T] && \text{(monad laws)}\\
&\qquad\cdot(f+\mathsf{id})\cdot(\mathsf{flatr}+\mathsf{id})\cdot(Se+\mathsf{id})\cdot(\mathsf{id}+(a\cdot m_A))\cdot e\\
&\quad= [a\cdot\mu^T\cdot T([T\mathsf{inl},\ \eta^T\cdot\mathsf{inr}]\cdot(f+\mathsf{id})\cdot|e|)^{\dagger},\ \mathsf{id}] && (a\text{ is Eilenberg–Moore})\\
&\qquad\cdot(f+\mathsf{id})\cdot(\mathsf{flatr}+\mathsf{id})\cdot(Se+\mathsf{id})\cdot(\mathsf{id}+(a\cdot m_A))\cdot e\\
&\quad= [a\cdot\mu^T\cdot T([T\mathsf{inl},\ \eta^T\cdot\mathsf{inr}]\cdot(f+\mathsf{id})\cdot|e|)^{\dagger},\ a] && \text{(coproducts)}\\
&\qquad\cdot(f+\mathsf{id})\cdot(\mathsf{flatr}+\mathsf{id})\cdot(Se+\mathsf{id})\cdot(\mathsf{id}+m_A)\cdot e\\
&\quad= a\cdot[\mu^T\cdot T([T\mathsf{inl},\ \eta^T\cdot\mathsf{inr}]\cdot(f+\mathsf{id})\cdot|e|)^{\dagger},\ \mathsf{id}] && \text{(coproducts)}\\
&\qquad\cdot(f+\mathsf{id})\cdot(\mathsf{flatr}+\mathsf{id})\cdot(Se+\mathsf{id})\cdot(\mathsf{id}+m_A)\cdot e\\
&\quad= a\cdot[\mu^T,\mathsf{id}]\cdot(T([T\mathsf{inl},\ \eta^T\cdot\mathsf{inr}]\cdot(f+\mathsf{id})\cdot|e|)^{\dagger}+\mathsf{id}) && \text{(coproducts)}\\
&\qquad\cdot(f+\mathsf{id})\cdot(\mathsf{flatr}+\mathsf{id})\cdot(Se+\mathsf{id})\cdot(\mathsf{id}+m_A)\cdot e\\
&\quad= a\cdot[\mu^T,\mathsf{id}]\cdot(f+\mathsf{id})\cdot(S([T\mathsf{inl},\ \eta^T\cdot\mathsf{inr}]\cdot(f+\mathsf{id})\cdot|e|)^{\dagger}+\mathsf{id}) && \text{(naturality of } f\text{)}\\
&\qquad\cdot(\mathsf{flatr}+\mathsf{id})\cdot(Se+\mathsf{id})\cdot(\mathsf{id}+m_A)\cdot e
\end{aligned}
$$

To see that both sides are equal, it is enough to show that the latter without the leading a is a solution of $[T\mathsf{inl}, T\mathsf{inr}]\cdot(f+m_A)\cdot e$. It can be proven by mundane calculation, see Section A.8 in the appendix.

It is left to check that Φ is well-defined also on morphisms, that is, $\Phi(h:\langle A,a\rangle\to\langle B,b\rangle)$ is the appropriate homomorphism of Eilenberg–Moore algebras and that it

preserves solutions. To check the former, we calculate:

$$\begin{aligned} h \cdot a \cdot m_A &= b \cdot Th \cdot m_A && (h \text{ is an algebra homomorph.}) \\ &= b \cdot m_B \cdot Mh && (\text{naturality of } m) \end{aligned}$$

We check that Φ preserves solutions. In detail, it means that for any flat equation morphism $e : X \to SX + A$, it is the case that $k \cdot e^\ddagger = (h \circledcirc e)^\ddagger$. We calculate:

$$\begin{aligned} h \cdot e^\ddagger &= h \cdot a \cdot \iota(f)_A \cdot [\![e]\!] && (\text{def. of } (\text{-})^\ddagger) \\ &= b \cdot Th \cdot \iota(f)_A \cdot [\![e]\!] && (h \text{ is an algebra homomorph.}) \\ &= b \cdot \iota(f)_B \cdot H^\infty h \cdot [\![e]\!] && (\text{naturality of } \iota(f)) \\ &= b \cdot \iota(f)_B \cdot [\![h \circledcirc e]\!] && (\text{Lemma 5.27}) \\ &= (h \circledcirc e)^\ddagger && (\text{def. of } (\text{-})^\ddagger) \end{aligned}$$

(ii) We show that the diagram (5.31) commutes. The left-hand side:

$$\begin{aligned} U^M \Phi \langle A,\ TA \xrightarrow{a} A \rangle &= U^M \langle A,\ MA \xrightarrow{a \cdot m_A} A,\ SA \xrightarrow{a \cdot f_A} A,\ (\text{-})^\ddagger \rangle && (\text{def. of } \Phi) \\ &= \langle A,\ MA \xrightarrow{m_A} TA \xrightarrow{a} A \rangle && (\text{def. of } U^M) \\ &= (\Delta m) \langle A,\ TA \xrightarrow{a} A \rangle && (\text{def. of } \Delta) \end{aligned}$$

The right-hand side:

$$\begin{aligned} U^S \Phi \langle A,\ TA \xrightarrow{a} A \rangle &= U^S \langle A,\ MA \xrightarrow{a \cdot m_A} A,\ SA \xrightarrow{a \cdot f_A} A,\ (\text{-})^\ddagger \rangle && (\text{def. of } \Phi) \\ &= \langle A,\ SA \xrightarrow{f_A} TA \xrightarrow{a} A,\ (\text{-})^\ddagger \rangle && (\text{def. of } U^S) \\ &= \langle A,\ S^\infty A \xrightarrow{\iota(f)_A} TA \xrightarrow{a} A \rangle && (\text{as discussed in (i)}) \\ &= (\Delta f) \langle A,\ TA \xrightarrow{a} A \rangle && (\text{def. of } \Delta) \end{aligned}$$

(iii) We show that Φ is the only carrier-preserving functor that makes the diagram (5.31) commute. For this, consider a monad morphism r that makes the diagram (5.30) commute when substituted for $\iota(m, f)$. By the same argument as in the proof of Theorem 4.28 (that is, μ^∞ is given by a distributive law), we obtain that $r = \mu^T \cdot (m * \iota(f))$. That is, Δr understood as a functor to $\textsc{Malg}(MS^\infty)$ is given as follows:

$$(\Delta r)(A,\ TA \xrightarrow{a} A) = \langle A,\ MS^\infty A \xrightarrow{m * \iota(f)_A} TTA \xrightarrow{\mu^T_A} TA \xrightarrow{a} A \rangle \tag{5.32}$$

The above understood as a resumption algebra can be given as follows:

$$\begin{aligned} \langle A,\ & MA \xrightarrow{M\eta^\infty_A} MS^\infty A \xrightarrow{m * \iota(f)_A} TTA \xrightarrow{\mu^T_A} TA \xrightarrow{a} A, \\ & SA \xrightarrow{\mathsf{emb}_A} S^\infty A \xrightarrow{\eta^M} MS^\infty A \xrightarrow{m * \iota(f)_A} TTA \xrightarrow{\mu^T_A} TA \xrightarrow{a} A,\ (\text{-})^\sharp \rangle \end{aligned}$$

where, for a flat equation morphism $e : X \to SX + A$, its solution $e^\sharp$ is defined as follows:

$$\begin{aligned} e^\sharp &= \Big(X \xrightarrow{[\![e]\!]} S^\infty A \xrightarrow{\eta^M} MS^\infty A \xrightarrow{m*\iota(f)_A} TTA \xrightarrow{\mu^T_A} TA \xrightarrow{a} A\Big) \\ &= \Big(X \xrightarrow{[\![e]\!]} S^\infty A \xrightarrow{\iota(f_A)} TA \xrightarrow{a} A\Big) = e^\ddagger \end{aligned}$$

We also obtain (again, by the same argument as in in the proof of Theorem 4.28) that $a \cdot \mu^T_A \cdot (m * \iota(f)_A) = a \cdot m_A$ and a $\cdot \mu^T_A \cdot (m * \iota(f)_A) \cdot \eta^M_{S^\infty A} \cdot \mathsf{emb}_A = a \cdot f_A$, which means that $(\Delta r)\langle A, a\rangle = \Phi\langle A, a\rangle$, hence $r = \iota(m, f)$. □

5.5 Two-sided modules

Adámek *et al.* first considered cims to be ideal monads [6], and later generalised them to idealised monads [12]. The use of right modules emphasises the asymmetry of the monadic composition when modelling computations. For example, let the monad in question formalise I/O, and the module be a subset of I/O programs that at some point of execution prints out `hello world`. This property is clearly preserved by the composition with any other computation on the right, since the latter is performed after the message is printed out. However, it is not necessarily preserved by the composition with an arbitrary computation on the left: if the latter fails with an exception or loops indefinitely, the former never gets a chance to print out its message.

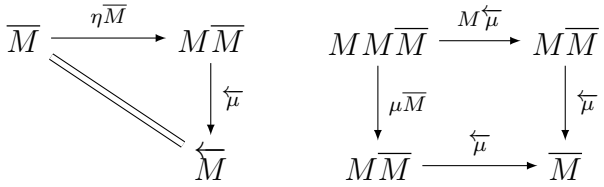
Nevertheless, in the case of cims, modules formalise a very concrete property: the notion of guardedness, which ensures productivity; see the discussion in Section 2.3.2. Informally, such a module contains effects that gradually contribute to the solution, for example, in the case of I/O, they perform observable actions. The values that are not in the module might not contribute, but they also do not obstruct the computation (this fact is formalised by Lemma 5.11). This leads to the corollary that every monad completely iterative with respect to a right module can be extended to a monad completely iterative with respect to a *two-sided* module without altering the monadic structure.

A more practical reason for the use of two-sided modules in this section is that one of our goals is to explore symmetries between the theory of 'ordinary' monads and the theory of cims. For example, consider Hyland, Plotkin, and Power's [65] result that $M(\Sigma M)^*$ is a coproduct in M*nd*.

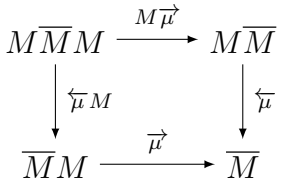
Definition 5.30. A *two-sided module* over a monad M is a triple $\langle \overline{M},\ \overleftarrow{\mu} : M\overline{M} \to \overline{M}, \overrightarrow{\mu} : \overline{M}M \to \overline{M}\rangle$ such that the following hold:

1. $\langle \overline{M}, \overrightarrow{\mu} \rangle$ is a module over M.

2. $\langle \overline{M}, \overleftarrow{\mu} \rangle$ is a *left module* over M, that is, the following diagrams commute:



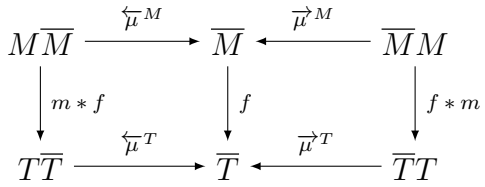
3. The morphisms $\overleftarrow{\mu}$ and $\overrightarrow{\mu}$ are coherent in the sense that the following diagram commutes:



Definition 5.31. A morphism between a two-sided M-module $\langle \overline{M}, \overleftarrow{\mu}^M, \overrightarrow{\mu}^M \rangle$ and a two-sided T-module $\langle \overline{T}, \overleftarrow{\mu}^T, \overrightarrow{\mu}^T \rangle$ is given by a pair of natural transformations $\langle m : M \to T,\ f : \overline{M} \to \overline{T} \rangle$ such that:

1. m is a monad morphism,

2. the following diagram commutes:



As in the case of ordinary modules, we define the *representation* of a monad:

Lemma 5.32. *Given a monad M, the triple $\langle M, \mu^M, \mu^M \rangle$ is a two-sided module over M.*

We also define an equivalent of idealised monads:

Definition 5.33. A *two-sided idealised monad* is a triple $\langle M, \langle \overline{M}, \overleftarrow{\mu}, \overrightarrow{\mu} \rangle, \sigma \rangle$, where $\langle \overline{M}, \overleftarrow{\mu}, \overrightarrow{\mu} \rangle$ is a two-sided M-module, and $\sigma : \overline{M} \to M$ is a natural transformation such that $\langle \mathsf{id}, \sigma \rangle$ is a morphism between $\langle \overline{M}, \overleftarrow{\mu}, \overrightarrow{\mu} \rangle$ and $\langle M, \mu^M, \mu^M \rangle$.

Definition 5.34. A two-sided idealised monad $\langle M, \langle \overline{M}, \overleftarrow{\mu}, \overrightarrow{\mu} \rangle, \sigma \rangle$ is a *two-sided cim* if every equation morphism $e : X \to M(X + A)$ that factors as

$$X \dashrightarrow \overline{M}(X + A) + A \xrightarrow{[\sigma,\ \eta^M \cdot \mathsf{inr}]} M(X + A)$$

has a unique solution.

Morphisms between two-sided cims are given, as in the case of cims, by coherent monad morphisms. In detail, given two-sided cims $\langle M, \langle \overline{M}, \overleftarrow{\mu}^M, \overrightarrow{\mu}^M \rangle, \sigma^M \rangle$ and $\langle T, \langle \overline{T}, \overleftarrow{\mu}^T, \overrightarrow{\mu}^T \rangle, \sigma^T \rangle$, a coherent morphism between them is given by a monad morphism $m : M \to T$ such that there exists a natural transformation $\overline{m} : \overline{M} \to \overline{T}$ such that $\sigma^T \cdot \overline{m} = m \cdot \sigma^M$.

We call the category of two-sided cims and coherent monad morphisms TCim.

Given an iteratable endofunctor Σ, the free cim $\langle \Sigma^\infty, \langle \Sigma\Sigma^\infty, \overline{\mu}^\infty \rangle, \sigma^\infty \rangle$ can be seen as a two-sided cim. We need to construct the 'left' module action $\overleftarrow{\mu}^\infty : \Sigma^\infty \Sigma\Sigma^\infty \to \Sigma\Sigma^\infty$ and show the necessary coherence conditions. For this, we use the following lemma:

Lemma 5.35. *Given a complete Elgot Σ-algebra $\langle A,\ a : \Sigma A \to A,\ (\text{-})^\dagger \rangle$, the tuple $\langle \Sigma A,\ \Sigma a : \Sigma\Sigma A \to \Sigma A,\ (\text{-})^\ddagger \rangle$ (where for a flat equation morphism $e : X \to \Sigma X + \Sigma A$, we define $e^\ddagger = [\Sigma(a \circledcirc e)^\dagger, \mathsf{id}] \cdot e$) is a complete Elgot Σ-algebra. Moreover, the morphism a is solution-preserving.*

Proof. This lemma is a special case of Adámek *et al.*'s result [15, Lemma 5.6], which states that for a morphism $g : Y \to A$, the tuple $\langle \Sigma A + Y,\ \Sigma(\Sigma A + Y) \xrightarrow{\Sigma[a,g]} \Sigma A \xrightarrow{\mathsf{inl}} \Sigma A + Y,\ (\text{-})^\ddagger \rangle$ (where for $e : X \to \Sigma X + \Sigma A + Y$, we have $e^\ddagger = ([\Sigma([a, g] \circledcirc e)^\dagger, \mathsf{id}] + \mathsf{id}) \cdot e$) is a complete Elgot Σ-algebra. Our case arises for $Y = 0$ (the initial object) and $g : 0 \to A$ being the unique morphism from the initial object. □

A corollary of the lemma above is the existence of a distributive law of Σ^∞ over Σ:

Corollary 5.36. *Let Σ be an iteratable endofunctor. Then, there exists a distributive law of a monad over an endofunctor $\theta : \Sigma^\infty \Sigma \to \Sigma\Sigma^\infty$.*

Proof. We define an Eilenberg–Moore lifting $\lceil \Sigma \rceil : \textsc{Elgot}(\Sigma) \to \textsc{Elgot}(\Sigma)$ as follows:

$$\lceil \Sigma \rceil \langle A, a, (\text{-})^\dagger \rangle = \langle \Sigma A, \Sigma a, (\text{-})^\ddagger \rangle \quad \text{(see Lemma 5.35)}$$
$$\lceil \Sigma \rceil h = \Sigma h$$

We need to check that $\lceil\Sigma\rceil$ is indeed a functor. It is left to see that if $h : A \to B$ is solution-preserving for two complete Elgot Σ-algebras $\langle A, a, (\text{-})^\dagger\rangle$ and $\langle B, b, (\text{-})^\flat\rangle$, then Σh is solution-preserving for $\lceil\Sigma\rceil\langle A, a, (\text{-})^\dagger\rangle = \langle \Sigma A, \Sigma a, (\text{-})^\ddagger\rangle$ and $\lceil\Sigma\rceil\langle B, b, (\text{-})^\flat\rangle = \langle \Sigma B, \Sigma b, (\text{-})^\sharp\rangle$. We calculate:

$$\begin{aligned}
(\Sigma h \circledcirc e)^\sharp &= [\Sigma(b \circledcirc \Sigma h \circledcirc e)^\flat, \mathsf{id}] \cdot (\Sigma h \circledcirc e) && \text{(def. of } (\text{-})^\sharp) \\
&= [\Sigma((b \cdot \Sigma h) \circledcirc e)^\flat, \mathsf{id}] \cdot (\Sigma h \circledcirc e) && \text{(def. of } \circledcirc) \\
&= [\Sigma((h \cdot a) \circledcirc e)^\flat, \mathsf{id}] \cdot (\Sigma h \circledcirc e) && \text{(Theorem 2.81)} \\
&= [\Sigma(h \circledcirc a \circledcirc e)^\flat, \mathsf{id}] \cdot (\Sigma h \circledcirc e) && \text{(def. of } \circledcirc) \\
&= [\Sigma(h \cdot (a \circledcirc e)^\dagger), \mathsf{id}] \cdot (\Sigma h \circledcirc e) && (h \text{ is sol.-preserving}) \\
&= [\Sigma(h \cdot (a \circledcirc e)^\dagger), \Sigma h] \cdot e && \text{(coproducts)} \\
&= \Sigma h \cdot [\Sigma(a \circledcirc e)^\dagger, \mathsf{id}] \cdot e && \text{(coproducts)} \\
&= \Sigma h \cdot e^\ddagger && \text{(def. of } (\text{-})^\ddagger)
\end{aligned}$$

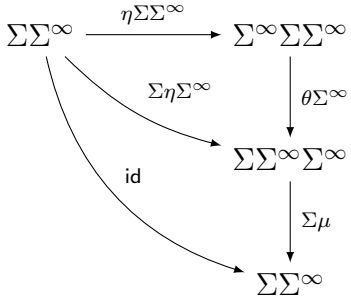
□

Theorem 5.37. *For an iteratable endofunctor Σ, the triple*

$$\mathbf{S} = \langle \Sigma^\infty,\ \langle \Sigma\Sigma^\infty,\ \Sigma\mu^\infty \cdot \theta\Sigma^\infty,\ \overline{\mu}^\infty\rangle,\ \sigma^\infty\rangle$$

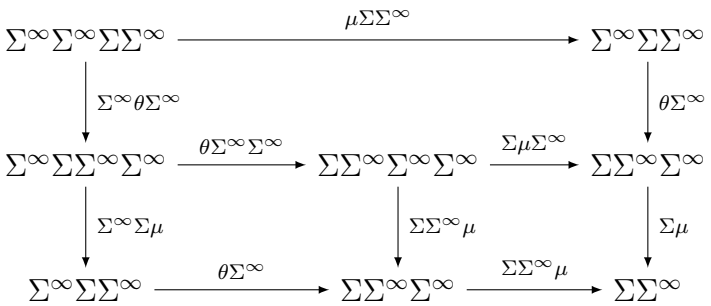
is a two-sided cim.

Proof. Since we know that $U^{\mathrm{TCim}}\mathbf{S}$ is a cim, it is left to check the conditions for the left module 5.30, and that $\langle \mathsf{id}, \sigma^\infty\rangle$ is an aproppriate morphism of two-sided modules. Left module:



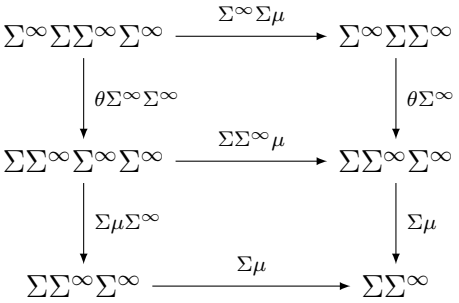
(The triangle of top follows from the fact that θ is a distributive law, the bottom

triangle follows from monad laws.)



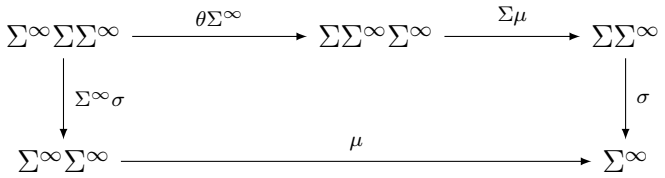
(The top part of the diagram follows from the fact that θ is a distributive law, the bottom-left square follows from the naturality of θ, the bottom-right square follows from monad laws.)

The coherence between the left and the right modules:



(The top square follows from the naturality of θ, the bottom square follows from monad laws.)

The proof that $\langle \mathsf{id}, \sigma \rangle$ is the appropriate morphism between two-sided modules is a bit more involved. Of course, the right module part is trivial. For the left module part, we need to show that the following diagram commutes:



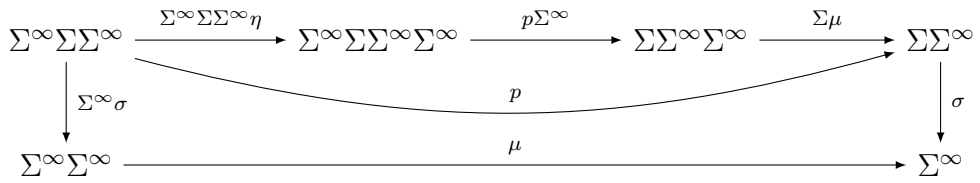
For this, we use the definition of θ from the lifting $\lceil \Sigma \rceil : \textsc{MAlg}(\Sigma^\infty) \to \textsc{MAlg}(\Sigma^\infty)$. In detail:

$$\theta = \left(\Sigma^\infty \Sigma \xrightarrow{\Sigma^\infty \Sigma \eta} \Sigma^\infty \Sigma \Sigma^\infty \xrightarrow{p} \Sigma \Sigma^\infty \right)$$

where p_A is a natural family of morphisms such that

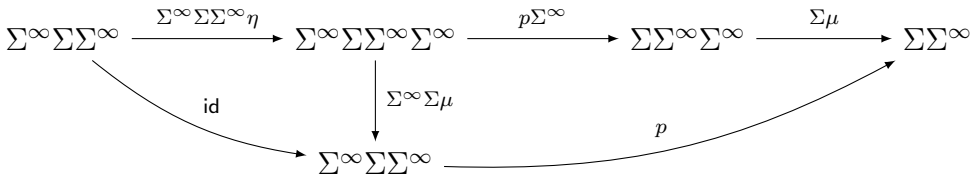
$$\lceil S \rceil \langle \Sigma^\infty A,\, \mu_A \rangle = \langle \Sigma \Sigma^\infty A,\ \Sigma^\infty \Sigma \Sigma^\infty A \xrightarrow{p_A} \Sigma \Sigma^\infty A \rangle$$

Thus, it is enough to show that the following diagram commutes:



(5.33)

First, we show that the top part of the diagram commutes. Consider the family of homomorphism between Eilenberg–Moore Σ^∞-algebras $\mu_A : \langle \Sigma^\infty\Sigma^\infty A,\ \mu_{\Sigma^\infty A}\rangle \to \langle \Sigma^\infty A,\ \mu_A\rangle$ (the fact that it is a homomorphism is exactly the associativity diagram for μ). This means that $\lceil\Sigma\rceil\mu_A = \Sigma\mu_A : \langle \Sigma\Sigma^\infty\Sigma^\infty,\ p_{\Sigma^\infty A}\rangle \to \langle \Sigma\Sigma^\infty,\ p_A\rangle$ is an algebra homomorphism. Thus, we obtain the following diagram, where the square on the right is the condition for $\lceil\Sigma\rceil\mu_A$ being an algebra homomorphism, and the triangle on the left follows from the monad laws:



To show that the bottom part of the diagram (5.33) commutes, it is enough to show that $\sigma_A : \Sigma\Sigma^\infty A \to \Sigma^\infty A$ is a homomorphism between $\lceil\Sigma\rceil\langle \Sigma^\infty A,\ \mu_A\rangle = \langle \Sigma\Sigma^\infty A,\ p_A : \Sigma^\infty\Sigma\Sigma^\infty A \to \Sigma\Sigma^\infty A\rangle$ and $\langle \Sigma^\infty A,\ \mu_A\rangle$. They can be seen respectively as the following Elgot algebras:

$$\lceil\Sigma\rceil\langle \Sigma^\infty A,\ \sigma_A : \Sigma\Sigma^\infty A \to \Sigma^\infty A,\ (\text{-})^\dagger\rangle = \langle \Sigma\Sigma^\infty A,\ \Sigma\sigma_A : \Sigma\Sigma\Sigma^\infty A \to \Sigma\Sigma^\infty A,\ (\text{-})^\ddagger\rangle$$

and

$$\langle \Sigma^\infty A,\ \sigma_A : \Sigma\Sigma^\infty A \to \Sigma^\infty A,\ (\text{-})^\dagger\rangle$$

Moreover, σ_A is a homomorphism from the latter to the former (since it is solution-preserving, see Lemma 5.35), hence it is a homomorphism of the respective Eilenberg–Moore Σ^∞-algebras. □

5.5.1 Free two-sided extension

Now, we show that very cim can be extended to a two-sided cim. Moreover, this extension is canonical, as it arises as a left adjoint to the forgetful functor TCim → Cim.

Theorem 5.38. *Let $U^{\mathrm{TCim}} : \mathrm{TCIM} \to \mathrm{CIM}$ be the obvious forgetful functor. It has a left adjoint $F^{\mathrm{TCim}} : \mathrm{CIM} \to \mathrm{TCIM}$ given as follows:*

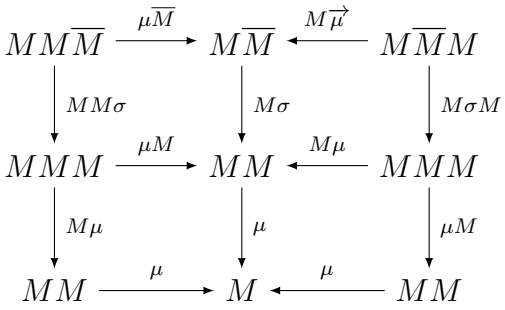
$$F^{\mathrm{TCim}}\langle M,\, \langle \overline{M}, \overline{\mu}\rangle,\, \sigma\rangle = \langle M,\ \langle M\overline{M}, \mu\overline{M}, M\overline{\mu}\rangle,\ \mu \cdot M\sigma\rangle$$
$$F^{\mathrm{TCim}} m = m$$

Proof. We need to show that: (i) F^{TCim} is a functor and (ii) it is a left adjoint to U^{TCim}.

(i) Given a cim $\langle M, \langle \overline{M}, \overline{\mu}\rangle, \sigma\rangle$, we need to check that $\langle M, \langle M\overline{M}, \mu\overline{M}, M\overline{\mu}\rangle, \mu\cdot M\sigma\rangle$ is a two-sided cim. First, we check that $\langle M\overline{M}, \mu\overline{M}, M\overline{\mu}\rangle$ is a two-sided module over M:

1. It is a (right) module by Example 2.57 (3).
2. It is a left module: both necessary diagrams (that is, $\mu\overline{M} \cdot \eta M\overline{M} = \mathsf{id}$ and $\mu\overline{M} \cdot M\mu\overline{M} = \mu\overline{M} \cdot \mu M\overline{M}$) follow trivially from the monad laws.
3. Coherence (that is, $\mu\overline{M} \cdot MM\overline{\mu} = M\overline{\mu} \cdot \mu\overline{M}M$) follows from the naturality of μ.

We also need to check that $\langle \mathsf{id}, \mu \cdot M\sigma\rangle$ is a morphism between two-sided modules, that is, the following diagram commutes:



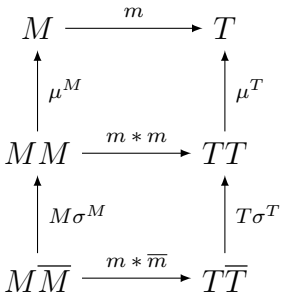
(Top-left corner: naturality of μ, top-right corner: $\langle \mathsf{id}, \sigma\rangle$ is a module morphism, bottom squares: monad laws.) Finally, we need to prove that an equation morphism that factors as

$$X \dashrightarrow M\overline{M}(X + A) + A \xrightarrow{[\mu\cdot M\sigma,\ \eta\cdot \mathsf{inr}]} M(X + A)$$

has a unique solution. This follows from Lemma 5.11 (b).

To see that U^{TCim} is a functor, it is left to see that for a coherent monad morphism $m : \langle M, \langle \overline{M}, \overline{\mu}^M\rangle, \sigma^M\rangle \to \langle T, \langle \overline{T}, \overline{\mu}^T\rangle, \sigma^T\rangle$, the morphism $U^{\mathrm{TCim}} m$ is also coherent.

For this, we see that the following diagram commutes, where $\overline{m}$ is any witness that m is coherent:



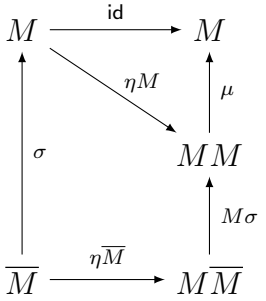
(The square on top: m is a monad morphism, the square on the bottom: coherence of m.)

(ii) It is left to show that F^{TCim} is a left adjoint to U^{TCim}. To do that, we define the unit u and counit c of the adjunction:

$$
\begin{aligned}
&u : \mathsf{Id}_{\mathrm{CIM}} \to U^{\mathrm{TCim}} F^{\mathrm{TCim}} \\
&u_{\langle M, \langle \overline{M}, \overline{\mu} \rangle, \sigma \rangle} : \langle M, \langle \overline{M}, \overline{\mu} \rangle, \sigma \rangle \to \langle M, \langle M\overline{M}, M\overline{\mu} \rangle, \mu \cdot M\sigma \rangle \\
&u = \mathsf{id}
\end{aligned}
$$

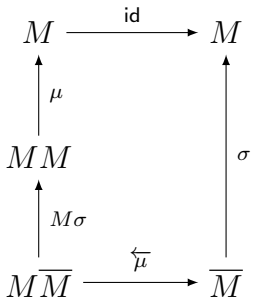
$$
\begin{aligned}
&c : F^{\mathrm{TCim}} U^{\mathrm{TCim}} \to \mathsf{Id}_{\mathrm{TCIM}} \\
&c_{\langle M, \langle \overline{M}, \overleftarrow{\mu}, \overrightarrow{\mu} \rangle, \sigma \rangle} : \langle M, \langle M\overline{M}, M\overrightarrow{\mu}, \overrightarrow{\mu} \rangle, \mu \cdot M\sigma \rangle \to \langle M, \langle \overline{M}, \overleftarrow{\mu}, \overrightarrow{\mu} \rangle, \sigma \rangle \\
&c = \mathsf{id}
\end{aligned}
$$

Note that the identities id above stand for identities in the base category, not the categories CIM or TCIM. The fact that c is coherent can be read from the following commutative diagram:



(Top-right corner: monad laws, bottom-left corner: naturality of η.) The fact that u

is coherent follows from the following diagram:



(This diagram follows from the fact that $\langle \mathsf{id}, \sigma \rangle$ is a morphism between two-sided modules.) It is trivial that the zig–zag equalities hold. □

5.5.2 Coproduct with free cim

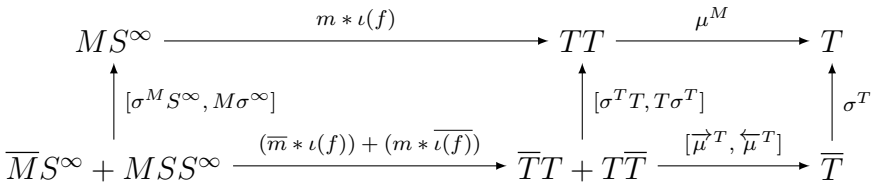
As the final result of this chapter, we show that the free two-sided extension of the coinductive resumption monad $M(\Sigma M)^\infty$ is the coproduct of M and Σ^∞ in TCim. First, we need an auxiliary lemma:

Lemma 5.39. *Let T be a two-sided cim, M be a cim, S be a (right) M-module, and $\langle m, f \rangle$ be a module morphism from $\langle M, S \rangle$ to $\langle T, T \rangle$. Then, the mediating monad morphism $\iota(m, f)$ given in Theorem 5.26 is coherent.*

Proof. We define the witness that $\iota(m, f)$ is coherent as follows:

$$[\overrightarrow{\mu}^T, \overleftarrow{\mu}^T] \cdot (\overline{m} * \iota(f)) + (m * \overline{\iota(f)}) : \overline{M}S^\infty + MSS^\infty \to \overline{T}$$

To see that it is indeed a witness, we notice that the following diagram commutes:



(the square on the left: m and $\iota(f)$ are coherent, the square on the right: $\langle \mathsf{id}, \sigma^T \rangle$ is a morphism of two-sided modules.) □

Theorem 5.40. *By* **R** *we denote the monad*

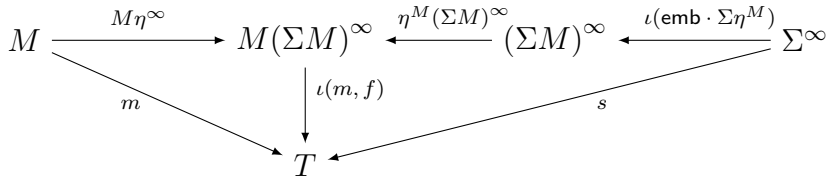
$$\langle M(\Sigma M)^\infty,\ \overline{M}(\Sigma M)^\infty + M\Sigma M(\Sigma M)^\infty,\ [\sigma^M S^\infty, M\sigma^\infty] \rangle$$

Let $\mathbf{M} = \langle M, \langle \overline{M}, \overleftarrow{\mu}^M, \overrightarrow{\mu}^M \rangle, \sigma^M \rangle$ *be a two-sided cim, and* Σ *be an iteratable endofunctor. Then, the two-sided cim* $F^{\mathrm{TCim}}\mathbf{R}$ *is the coproduct of* $\mathbf{M}$ *and* $\mathbf{S}$ *(see Theorem 5.37) in the category* TCim.

Proof. Consider a two-sided cim $\mathbf{T} = \langle T, \langle \overline{T}, \overleftarrow{\mu}^T, \overrightarrow{\mu}^T \rangle, \sigma^T \rangle$ and the cospan $M \xrightarrow{m} T \xleftarrow{s} \Sigma^\infty$ in MND such that both m and s are coherent. Consider the following morphism:

$$f = \left(\Sigma M \xrightarrow{(s \cdot \mathsf{emb}) * m} TT \xrightarrow{\mu^T} T \right)$$

We propose the monad morphism $\iota(m, f)$ given in Theorem 5.26 as the coproduct mediator. In detail, we need to show that $\iota(m, f)$ is the unique monad morphism that makes the following diagram in MND commute, and that all involved monad morphisms are coherent:

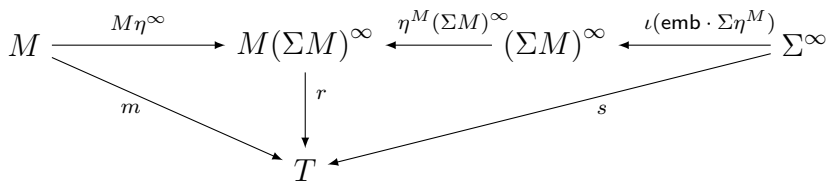


(5.34)

We show that the diagram (5.34) commutes. The triangle on the left follows from the universal property of coinductive resumptions (Theorem 5.26). For the triangle on the right, we calculate:

$$\begin{aligned}
&\iota(m, f) \cdot \eta^M(\Sigma M)^\infty \cdot \iota(\mathsf{emb} \cdot \Sigma\eta^M) \\
&= \mu^M \cdot (m * \iota(f)) \cdot \eta^M(\Sigma M)^\infty \cdot \iota(\mathsf{emb} \cdot \Sigma\eta^M) && \text{(def. of } \iota(m, f)) \\
&= \iota(f) \cdot \iota(\mathsf{emb} \cdot \Sigma\eta^M) && (m \text{ is a monad morph.}) \\
&= \iota(\iota(f) \cdot \mathsf{emb} \cdot \Sigma\eta^M) && \text{(Lemma 2.46, fusion)} \\
&= \iota(f \cdot \Sigma\eta^M) && \text{(Lemma 2.46, cancellation)} \\
&= \iota(\mu \cdot ((s \cdot \mathsf{emb}) * m) \cdot \Sigma\eta^M) && \text{(def. of } f) \\
&= \iota(s \cdot \mathsf{emb}) && (m \text{ is a monad morph.}) \\
&= s \cdot \iota(\mathsf{emb}) && \text{(Lemma 2.46, fusion)} \\
&= s && \text{(Lemma 2.46, reflection)}
\end{aligned}$$

We need to show that $\iota(m, f)$ is a unique morphism that makes the diagram (5.34) commute. For this, consider a monad morphism $r : M(\Sigma M)^\infty \to T$ that makes the following diagram commute:



(5.35)

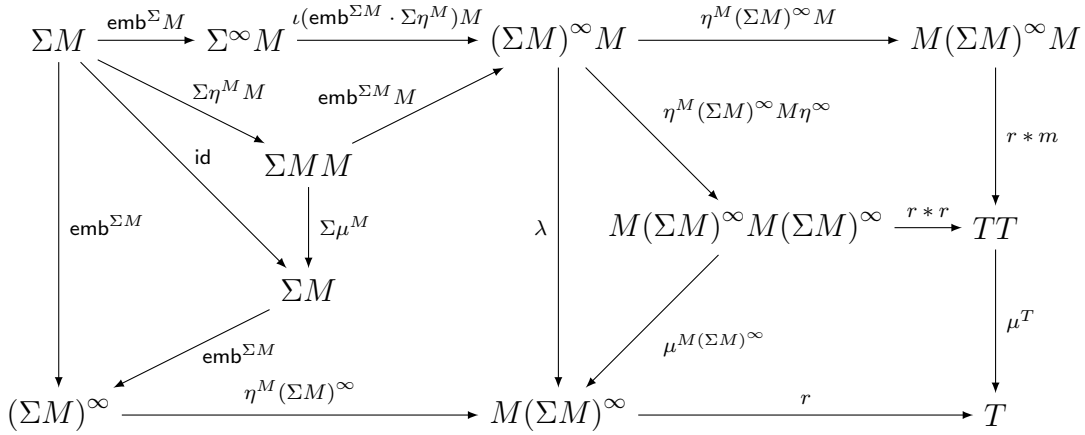
We show that r satisfies the universal property given in Theorem 5.26, that is, $m = r \cdot M\eta^\infty$, which follows trivially from the diagram (5.35), and $f = r \cdot \eta^M(\Sigma M)^\infty \cdot \mathsf{emb}^{\Sigma M}$. Unfolding the definition of f, we obtain that it is enough to prove that

$$\mu^T \cdot ((s \cdot \mathsf{emb}^\Sigma) * m) = r \cdot \eta^M(\Sigma M)^\infty \cdot \mathsf{emb}^{\Sigma M}$$

Substituting for s using the equality given by the right-hand side of the diagram (5.35), it is enough to show that

$$\mu^T \cdot (((r \cdot \eta^M(\Sigma M)^\infty \cdot \iota(\mathsf{emb}^{\Sigma M} \cdot \Sigma\eta^M)) \cdot \mathsf{emb}^\Sigma) * m) = r \cdot \eta^M(\Sigma M)^\infty \cdot \mathsf{emb}^{\Sigma M}$$

This equality is shown with the following commutative diagram:



The coherence of $M\eta : M \to M(\Sigma M)^\infty$ is witnessed by

$$\begin{aligned} \overline{M} &\xrightarrow{\overline{M}\eta^\infty} \overline{M}(\Sigma M)^\infty \xrightarrow{\mathsf{inl}} \overline{M}(\Sigma M)^\infty + M\Sigma M(\Sigma M)^\infty \\ &\xrightarrow{\eta^{M(\Sigma M)^\infty}} M(\Sigma M)^\infty(\overline{M}(\Sigma M)^\infty + M\Sigma M(\Sigma M)^\infty) \end{aligned}$$

The coherence of $\eta^M(\Sigma M)^\infty \cdot \iota(\mathsf{emb} \cdot \Sigma\eta^M)$ is witnessed by

$$\begin{aligned} \Sigma\Sigma^\infty &\xrightarrow{\eta^M\Sigma\eta^M * \iota(\mathsf{emb} \cdot \Sigma\eta^M)} M\Sigma M(\Sigma M)^\infty \xrightarrow{\mathsf{inr}} \overline{M}(\Sigma M)^\infty + M\Sigma M(\Sigma M)^\infty \\ &\xrightarrow{\eta^{M(\Sigma M)^\infty}} M(\Sigma M)^\infty(\overline{M}(\Sigma M)^\infty + M\Sigma M(\Sigma M)^\infty) \end{aligned}$$

To show the coherence of $\iota(m, f) : F^{\mathrm{TCim}}\mathbf{R} \to \mathbf{T}$, it is enough to show the coherence of $\varphi(\iota(m, f)) : \mathbf{R} \to U^{\mathrm{TCim}}\mathbf{T}$ (where φ is the natural isomorphism associated to the adjunction $F^{\mathrm{TCim}} \dashv U^{\mathrm{TCim}}$), which is shown in Lemma 5.39. □

Chapter 6

Discussion and related work

6.1 Summary

This dissertation is a collection of results concerning cims, their applications, properties, and constructions. We use the language and tools of elementary category theory, especially that the studied objects are very categorical themselves: adjunctions, free monads (generated by various entities), distributive laws, or liftings. Thus, in this respect, the presented constructions feel natural and hardly surprising to a category theory practitioner. Figure 6.1 shows a summary of the introduced constructions.

Importantly, the gain from the fact that we use only elementary category theory is that the results are easily expressible in programming languages with coinductive types. Haskell programmers already use instances of the coinductive resumption monad, see Section 6.2, our aim was to provide some theory behind it.

This dissertation can be also understood as an attempt to better understand completely iterative monads, which are important elements of the general theory of recursion and iteration [36, 37, 6, 95, 91], which has been an active area of research from the very beginning of computer science.

6.2 Related work

Foundations of coinduction. From the very first days of computer science, coinductive definitions were needed to describe Turing-complete models of computation (see Jacobs [67, Sec. 7] for a coalgebraic account). In semantics, the coinductive understanding of programs became necessary to deal with concurrency, see, for example, Milner [87].

There has been a great amount of research on the mathematics behind coinductive definitions. For example, for cardinality reasons, the powerset functor does not have a

The composition theorem

general	UMF	($F \dashv U$, M – cim), Theorem 3.1
instances	the metric state monad $(L(\text{-} \times S))^S$	(L – step-counting lifting of a metric space, cf. [38]), Corollary 3.11
	the states monad $(\overrightarrow{S}(\text{-} \times S))^S$	($\overrightarrow{S}A = (\text{-} \times S)^\infty A$), Definition 3.13

The inductive resumption monad

general	MS^*	(M – monad, S – module), Section 4.2
instances	the coproduct of M and Σ^* in Mnd (cf. [65]) $M(\Sigma M)^*$	(Σ – endofunctor, M – monad), Example 4.23
	the inductive logging state monad RW^*	(R – reader, W – writer), Example 4.18

The coinductive resumption monad

general	MS^∞	(M – monad, S – module), Section 5.1
instances	the coproduct of M and Σ^∞ in TCim $M(\Sigma M)^\infty$	(Σ – endofunctor, M – monad), Corollary 5.40
	probabilistic process $\mathsf{D}(\Sigma\mathsf{D})^\infty$	(Σ – signature (endofunctor), D – probability distribution monad), Example 5.5
	the coinductive logging state monad RW^∞	(R – reader, W – writer), Example 5.6(a)
	the states monad (again) RWW^∞	(R – reader, W – writer), Example 5.6(b)

Figure 6.1: Summary of the introduced structures

final coalgebra (nor initial algebra). To allow models of unbounded non-determinism, Plotkin [103] proposed the powerdomain construction in domain theory. Aczel and Mendler [7] proved that every endofunctor on the category of sets has a final coalgebra in the category of proper classes. Aczel [5] also proposed an axiomatisation of the non-well-founded set theory, which is ZFC with the foundation axiom replaced by its negation. This allows a set to be a member of itself, thus eliminating the cardinality issues. See also Turi and Rutten [115] for more detailed introduction on the foundations of final coalgebra semantics.

Data types as final coalgebras. Coalgebraic models of infinite data types were discussed by Wand [121] and Hagino [51]. In programming—especially in functional languages—it led to the development of a number of corecursion schemes, which allow

to define corecursive functions and reason about them using universal properties [40, 58, 60, 82, 117, 118]. Most of the constructions presented in this dissertation are built around final coalgebras, so they can be easily expressed in lazy languages, such as Haskell.

Semantics. Relevant work on semantics include a number of operational approaches. We list some of them. Peyton Jones [98] introduced reduction semantics for the IO monad in Haskell. Leroy and Grall [77] and Nakata and Uustalu [92, 93, 94] studied coinductive proof trees together with structures like streams of states and resumptions. Coinductive techniques were used by Schmidt [108] to give abstract trace-based semantics and Ancona [20] in the case of object-oriented languages.

In type theory, Hancock and Setzer [52, 53] and Michelbrink and Setzer [83] used (weakly) final coalgebras to build 'interaction trees', to build models similar to those given by the free cim.

Iterative and coalgebraic monads. Elgot pioneered a purely algebraic approach to recursive definitions, with no reference to any background structure like order or metric. He introduced the concept of iterative theories and monads [36] (which need some finiteness constraints) and then, together with Bloom and Tindell, complete iterative theories [37] (with no constraints on cardinality). Nelson [95] generalised and simplified Elgot's results on iterative theories via so-called iterative algebras (on the category of sets).

Later, in a series of papers, Adámek *et al.* generalised Elgot and others' results to the category-theoretic, coalgebraic setting [6, 9, 12, 15, 16, 84]. Milius [84] discusses the properties of the monad Σ^∞ (that it is a cim and the free cim) in the generality presented in this dissertation.

The monad Σ^∞ was also discussed by Moss [91], who, independently of Adámek *et al.* [6], proved a result equivalent to complete iterativity. Ghani *et al.* [31, 43, 80] also worked on solving equations coalgebraically, and introduced so-called *coalgebraic monads* to unify different kinds of term monads (finite, infinite, rational terms, term graphs).

Resumptions. In semantics, domain-theoretic resumptions were studied by Milner [87], Plotkin [103], and Papaspyrou [97]. The version given by final coalgebras $\nu X.(O \times X)^I$ on Set is due to Abramsky [2] (who used resumptions in the definition of interaction categories [3]). Later, the generalisation to $\nu X.M(O \times X)^I$ (for any

SET-monad M) was studied by Hasuo and Jacobs [55]. Importantly, such resumptions form a category, in which objects are given by objects on the base category, while an arrow $A \to B$ is given by a resumption $\nu X.M(B \times X)^A$. Such categories are traced monoidal (where the tensor is given by coproduct), which allowed Abramsky to model typed concurrency using *interaction categories* [3] and to give a categorical account [4] for Girard's *geometry of interaction* [46], which is an important construction in proof theory. The generalised, monadic version of resumptions do not have a natural notion of trace, even if they form a category. Even though, it is an interesting question whether they can be used to build generalised interaction categories.

The monadic version $\mu X.M(\Sigma X + (\text{-}))$ (for an endofunctor Σ) is due to Cenciarelli and Moggi [28]. Hyland, Plotkin, and Power [65] proved that it is the coproduct of M and Σ^* in the category of monads and monad morphisms. The coinductive version in the form of $\nu X.M(X + (\text{-}))$ was studied by Goncharov and Schröder [50] to give semantics of side-effecting processes. Note that our coinductive resumption monad MS^∞ is not in general given by carriers of final coalgebras.

In programming, different forms of resumptions have been independently rediscovered dozens of times, and used for flexible structuring of programs, though usually without much formal treatment. In the Lisp community, resumptions were dubbed 'engines' (Haynes and Friedman [57], Dybvig and Hieb [34]) or 'trampolined' programs (Ganz, Friedman, and Wand [41]). All of those constructions were all given in terms of continuations, and not (co)algebraic data types. However, in this dissertation, we discuss non-trivial adjunctions that give rise to the (co)inductive resumption monads, so we can obtain 'CPS-ed' versions via the codensity monads generated by the right adjoints (see Hinze [59]). In Haskell, the resumption monad $M(\Sigma M)^\infty$ can be found under the name 'free monad transformer' (since there exists a canonical monad morphism $M \to M(\Sigma M)^\infty$, the functor $M \mapsto M(\Sigma M)^\infty$ is a monad transformer in the sense of Moggi [88]). Different programming patterns that involve resumptions were discussed by Claessen [29], Kiselyov [72], and Harrison [54]. Interleaving of data and effects in the algebraic context was also studied by Filinski and Støvring [39] (whose M-and-F-algebras are subsumed by algebras for modules, see Example 4.24), and Atkey *et al.* [22].

Generic trace semantics. A relevant construction is the generic trace semantics introduced by Jacobs *et al.* [56, 69] (see also Goncharov [49]). It is given by final coalgebras of an endofunctor Σ on a Kleisli category of a monad M. Just as in the case of resumptions, the monad M is responsible for local effects. For example, a

trace semantics of a probabilistic finite automaton can be specified by the functor $\Sigma X = 2 \times X^I$ on Set, where I is the input alphabet and 2 is the set of boolean values (accepting vs non-accepting states), and the probability distribution monad that specifies the 'branching' of the system. We show that the coinductive resumption monad is an example of the generic trace semantics, since it can be seen as a free cim in the Kleisli category of M, see Theorem 5.19).

6.3 Future work

6.3.1 More abstract approach

Street [110] generalised the concept of monads to the 2-categorical setting. Given a 2-category, a monad consists of an object $\mathbb{C}$, a 1-cell $M : \mathbb{C} \to \mathbb{C}$, and a pair of 2-cells $\eta : \mathsf{Id} \to M$ and $\mu : MM \to M$ that satisfy the obvious conditions. Most of the theory presented in the first part of Chapter 2 has been generalised to the new setting. Hyland, Plotkin, and Power [65] suggest investigating the construction $M(\Sigma M)^*$ (see Section 2.2.8) in such a broader context. Concerning the constructions presented in this dissertation, one can easily define a module over the monad M as a 1-cell $S : \mathbb{C} \to \mathbb{C}$ together with a 2-cell $\overline{\mu} : SM \to M$ that satisfy the conditions given in Definition 2.53 in Section 2.3.2.

Although the definition of cims uses objects (of variables X and parameters A) of the base category, one can easily prove the following:

Lemma 6.1. *Let G and H be endofunctors, and $e_X : GX \to M(GX + HX)$ be a natural transformation with guarded components. Then, $e^\dagger_X : GX \to MHX$ is natural in X. The converse also holds: If for every such a natural transformation e there exists a natural transformation $e^\dagger$ such that the appropriate solution diagram commutes, then M is a cim.*

This clearly generalises to a 2-categorical definition of complete iterative monads. The question is how much of the theory presented in this dissertation can be given in the more abstract setting of 2-categories and to what extent the corresponding constructions in Chapters 4 and 5 have a common 2-categorical underpinning.

There are other incarnations of similar constructions. For example, Adámek *et al.* [12, 13] consider also a finitary case: they define an *iterative monad* (without 'completely') as a finitary monad on a locally finitely presentable category, such that all guarded equation morphisms with finitely presentable object of variables have unique solutions. Similarly, there exists a finitary version of complete Elgot algebras

(namely, *Elgot algebras*) together with an analogue of Theorem 2.83. This suggests that the presented constructions should scale to the finitary case, but the details are yet to be worked out.

Also, there are other similarities between different construction that scream for a higher-category-theoretic treatment. For example, as we indicate in Chapter 4, the theory of modules (adjunctions, distributive laws, liftings) is similar in nature to the theory of monads. Such a general approach was taken, for instance, in the case of distributive law in general; see Rypáček's PhD dissertation [106]. It is reasonable to ask how the distributive laws presented in this dissertation fit in Rypáček's framework.

On the practical side, we could benefit from such a more general approach by simplifying proofs in this dissertation, which are mostly tedious calculations, especially those deferred to Appendix A. Obviously, one could argue that there is a certain amount of complexity built in the strucutre of cims. Epsecially the definition of guardedness seems to be an obstacle, and the complexity doubles when one wants to reason about composition of monads (although Uustalu [116] worked with a version complete iterativity with a more abstract notion of guardedness).

6.3.2 Duality between monads and cims

Adámek *et al.*'s and our results suggest a form of 'least- vs greatest fixed points' duality between ordinary monads and cims: free objects are given by $F^*A = \mu X.FX + A$ and $F^\infty A = \nu X.FX + A$ respectively, and coproducts with free objects by $M(\Sigma M)^*$ and $M(\Sigma M)^\infty$. There are other constructions on monads that involve initial algebras, for example the coproduct of two ideal monads, given by Ghani and Uustalu [44]. Recall from Definition 2.58 that a monad M idealised with a module $\overline{M}$ is called *ideal* if $M = \overline{M} + \mathsf{Id}$.

Ghani and Uustalu's construction can be summarised as follows: Let $\mathbb{C}$ be a category with binary coproducts, while $M = \overline{M} + \mathsf{Id}$ and $T = \overline{T} + \mathsf{Id}$ be two ideal monads on $\mathbb{C}$. First, we define a functor in the product category $\mathbb{C} \times \mathbb{C}$ via carriers of initial algebras:

$$\langle M'A, T'A\rangle = \mu\langle X, Y\rangle.\langle M(Y + A),\, T(X + A)\rangle$$

The functor part of the coproduct of M and T in $\textsc{Mnd}(\mathbb{C})$ is given as:

$$(M \boxplus T)A = M'A + T'A + A$$

Intuitively, we can describe such a coproduct as a set of trees with interleaved $\overline{M}$ and $\overline{T}$ as nodes and As in leaves.

Consider two signatures Σ and Γ, and their respective free cims on Set. Their coproduct in Cim is given (by freeness) as $(\Sigma + \Gamma)^\infty$, which is clearly different from $\Sigma^\infty \boxplus \Gamma^\infty$. The corollary is that, in general, coproducts in Mnd of completely iterative monads are not necessarily completely iterative. Or, more accurately, they might be iterative with respect to, for example, trivial modules, but in such a case coproduct injections are not solution-preserving.

The conjecture is that the coproduct in Cim of two ideal cims can be given similarly to Ghani and Uustalu's construction, but with ν instead of μ. Moreover, as shown by Adámek *et al.* [11], Ghani and Uustalu's formula can be applied to a wider scope of monads (so-called *consistent* monads) on Set. It is an interesting question to see if this approach also scales to the case of cims.

6.3.3 Type theory

Another interesting challenge is in formalising the results presented in this dissertation in an interactive proof system, such as Coq or Agda. Modern proof systems implement coinductive data types, but they are usually heavily restricted in order to enjoy decidable type checking.

Such an implementation would be useful for reasoning about coinductive programs and coinductive data structures. So far, they can be implemented in a regular programming languages with coinductive data types, and use, for example, to test programs with I/O, using tools like QuickCheck [30].

Appendix A

Proofs and calculations

A.1 The rolling rule

Lemma A.1. *Let F, G be endofunctors. Then $\nu FG \cong F\nu GF$.*

Proof. Let $\langle \nu FG, \beta : \nu FG \to FG\nu FG \rangle$ be the final FG-coalgebra, and $\langle \nu GF, \gamma : \nu GF \to GF\nu GF \rangle$ be the final GF-coalgebra. We define the following morphisms (subscripts for the lens brackets indicate the functor for the final coalgebra):

$$\begin{aligned} r &: \nu FG \to F\nu GF \\ r &= F[\![G\beta]\!]_{GF} \cdot \beta \end{aligned}$$

$$\begin{aligned} r^{-1} &: F\nu GF \to \nu FG \\ r^{-1} &= [\![F\gamma]\!]_{FG} \end{aligned}$$

To prove that r is an isomorphism, with r^{-1} being its inverse, we show that $\beta \cdot r^{-1} \cdot r = FGr^{-1} \cdot FGr \cdot \beta$, which means that $r^{-1} \cdot r$ is an FG-coalgebra homomorphism $r^{-1} \cdot r : \langle \nu FG, \beta \rangle \to \langle \nu FG, \beta \rangle$. By uniqueness and reflection, $r^{-1} \cdot r = [\![\beta]\!] = id_{\nu FG}$.

$$\begin{aligned} & \beta \cdot r^{-1} \cdot r = \beta \cdot [\![F\gamma]\!]_{FG} \cdot F[\![G\beta]\!]_{GF} \cdot \beta \\ = \quad & \text{(computation)} \\ & \beta \cdot \beta^{-1} \cdot FG[\![F\gamma]\!]_{FG} \cdot F\gamma \cdot F\gamma^{-1} \cdot FGF[\![G\beta]\!]_{GF} \cdot FG\beta \cdot \beta \\ = \quad & \text{(isomorphisms)} \\ & FG[\![F\gamma]\!]_{FG} \cdot FGF[\![G\beta]\!]_{GF} \cdot FG\beta \cdot \beta \\ = \quad & \text{(definitions of } r \text{ and } r^{-1} \text{)} \\ & FGr^{-1} \cdot FGFr \cdot \beta \end{aligned}$$

To prove the converse, we first calculate:

$$\begin{aligned} & r \cdot r^{-1} = F[\![G\beta]\!]_{GF} \cdot \beta \cdot [\![F\gamma]\!]_{FG} \\ = & \quad (\text{ computation }) \\ & F[\![G\beta]\!]_{GF} \cdot \beta \cdot \beta^{-1} \cdot FG[\![F\gamma]\!]_{FG} \cdot F\gamma \\ = & \quad (\text{ isomorphisms }) \\ & F[\![G\beta]\!]_{GF} \cdot FG[\![F\gamma]\!]_{FG} \cdot F\gamma \end{aligned}$$

The last expression is an F-image of $r^{-1} \cdot r$, but with F and G swapped. Thus, we can prove similarly to the previous case that it is equal to the identity on νGF. □

A.2 Direct definition of the monadic structure of inductive resumptions

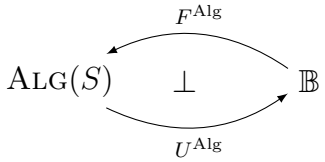
Step 1: A distributive law of a functor over a monad. We start with the following distributive law of S (as a functor) over M.

$$\delta = \left(SM \xrightarrow{\overline{\mu}^S} S \xrightarrow{\eta^M S} MS \right)$$

Step 2: A lifting to the category of algebras. The distributive law induces the following lifting:

$$\begin{aligned} & M' : \textsc{Alg}(S) \to \textsc{Alg}(S) \\ & M'\langle A,\, a : SA \to A \rangle = \langle MA,\, SMA \xrightarrow{\delta_A} MSA \xrightarrow{Ma} MA \rangle \\ & M'f = Mf \end{aligned}$$

Step 3: A lifting to the category of Eilenberg-Moore algebras. Consider the free-forgetful adjunction:



where

$$\begin{aligned} & F^{\mathrm{Alg}} A = \langle S^* A,\, \tau : SS^* A \to S^* A \rangle \qquad && U^{\mathrm{Alg}} \langle A, a : SA \to A \rangle = A \\ & F^{\mathrm{Alg}} f = S^* f && U^{\mathrm{Alg}} f = f \end{aligned}$$

The counit of this adjunction is given by:

$$\varepsilon_{\langle A,\, a:SA\to A\rangle} : \langle S^*A, \tau\rangle \to \langle A, a\rangle$$
$$\varepsilon_{\langle A,\, a:SA\to A\rangle} = (\!|a, \mathsf{id}|\!)$$

where $(\!|a, \mathsf{id}|\!)$ is the unique algebra homomorphism induced by the morphism:

$$[a, \mathsf{id}] : SA + A \to A$$

Consider the comparison functor $\Phi : \textsc{Alg}(S) \to \textsc{MAlg}(S^*)$:

$$\Phi\langle A,\, a : SA \to A\rangle = \langle A,\, \varepsilon_{\langle A,a\rangle} = (\!|a, \mathsf{id}|\!) : S^*A \to A\rangle$$
$$\Phi f = f$$

The adjunction $F \dashv U$ is monadic, so Φ is an isomorphism. Its inverse is given as:

$$\Phi^{-1}\langle A, a : S^*A \to A\rangle = \langle A, S \xrightarrow{\mathsf{emb}_A} S^*A \xrightarrow{a} A\rangle$$
$$\Phi^{-1} f = f$$

We can define the lifting of M to $\textsc{MAlg}(S^*)$ as follows:

$$\begin{aligned}
\lceil M\rceil\langle A, a : S^*A \to A\rangle &= \Phi M' \Phi^{-1}\langle A, a\rangle \\
&= \Phi M'\langle A, a \cdot \mathsf{emb}_A\rangle \\
&= \Phi\langle MA,\, SMA \xrightarrow{\delta_A} MSA \xrightarrow{M\mathsf{emb}_A} MS^*A \xrightarrow{Ma} MA\rangle \\
&= \langle MA,\, (\!|Ma \cdot M\mathsf{emb}_A \cdot \delta_A,\, \mathsf{id}|\!) : S^*MA \to MA\rangle
\end{aligned}$$

Obviously, the sole isomorphism $\textsc{Alg}(S) \cong \textsc{MAlg}(S^*)$ is not enough to make $\lceil M\rceil$ a lifting; to make sure that the appropriate conditions hold, we need the isomorphism in $\textsc{Adj}(M)$ (the category of adjunctions that induce M). Fortunately, this is the case for every monadic adjunction.

Step 4: A distributive law of a monad over a monad. The distributive law induced by the lifting $\lceil M\rceil$ is given as follows. Consider the free Eilenberg-Moore algebra $\langle S^*A,\, \mu_A^* : S^*S^*A \to S^*A\rangle$, and its $\lceil M\rceil$-image:

$$\lceil M\rceil\langle S^*A, \mu_A^*\rangle = \langle MS^*A,\, g : S^*MS^*A \to MS^*A\rangle,$$

where

$$g = (\!|M\mu_A^* \cdot M\mathsf{emb}_{S^*A} \cdot \delta_{S^*A},\, \mathsf{id}|\!) = (\!|\mu^M_{S^*A} \cdot \tau_A \cdot \overline{\mu}^S_{S^*A},\, \mathsf{id}|\!)$$

The distributive law is given by the following composition:

$$\lambda_A = \left(S^*MA \xrightarrow{S^*M\eta_A^*} S^*MS^*A \xrightarrow{g} MS^*A\right)$$

We can fuse the composition into one fold using the following lemma:

Lemma A.2. *Given two endofunctors F and H together with two morphisms $f : FB \to B$ and $h : A \to B$, it holds that $(\!(f, h)\!) = (\!(f, \mathsf{id})\!) \cdot F^* h : F^* A \to B$.*

This gives an alternative definition: $\lambda_A = (\!(\mu^M_{S^*A} \cdot \tau_A \cdot \overline{\mu}^S_{S^*A},\ M\eta^*_A)\!)$.

A.3 Proof of Lemma 5.3

The value of the assignment is a complete Elgot algebra. Let $e : X \to SX + MA$ be a flat equation morphism. To show that $e^\ddagger$ is a solution, we first calculate:

$$
\begin{aligned}
& S([a, \mathsf{id}] \cdot (S|e|^\dagger + \mathsf{id}) \cdot ((\mathsf{flatr} \cdot Se) + \mathsf{id})) \cdot \mathsf{flatr} \\
= \quad & \text{(coproducts)} \\
& S([a, \mathsf{id}] \cdot (S|e|^\dagger + \mathsf{id}) \cdot (\mathsf{flatr} + \mathsf{id}) \cdot (Se + \mathsf{id})) \cdot \mathsf{flatr} \\
= \quad & \text{(functor)} \\
& S[a, \mathsf{id}] \cdot S(S|e|^\dagger + \mathsf{id}) \cdot S(\mathsf{flatr} + \mathsf{id}) \cdot S(Se + \mathsf{id}) \cdot \mathsf{flatr} \\
= \quad & \text{(naturality of } \mathsf{flatr} \text{)} \\
& S[a, \mathsf{id}] \cdot \mathsf{flatr} \cdot S(S|e|^\dagger + \mathsf{id}) \cdot S(\mathsf{flatr} + \mathsf{id}) \cdot S(Se + \mathsf{id}) \\
= \quad & \text{(naturality of } \mathsf{flatr} \text{)} \\
& S[\mathsf{id}, \mathsf{id}] \cdot \mathsf{flatr} \cdot S(a + \mathsf{id}) \cdot S(S|e|^\dagger + \mathsf{id}) \cdot S(\mathsf{flatr} + \mathsf{id}) \cdot S(Se + \mathsf{id}) \\
= \quad & \text{(flattening)} \\
& \mathsf{flat}' \cdot S[\eta^M, \mathsf{id}] \cdot S(a + \mathsf{id}) \cdot S(S|e|^\dagger + \mathsf{id}) \cdot S(\mathsf{flatr} + \mathsf{id}) \cdot S(Se + \mathsf{id})
\end{aligned}
$$

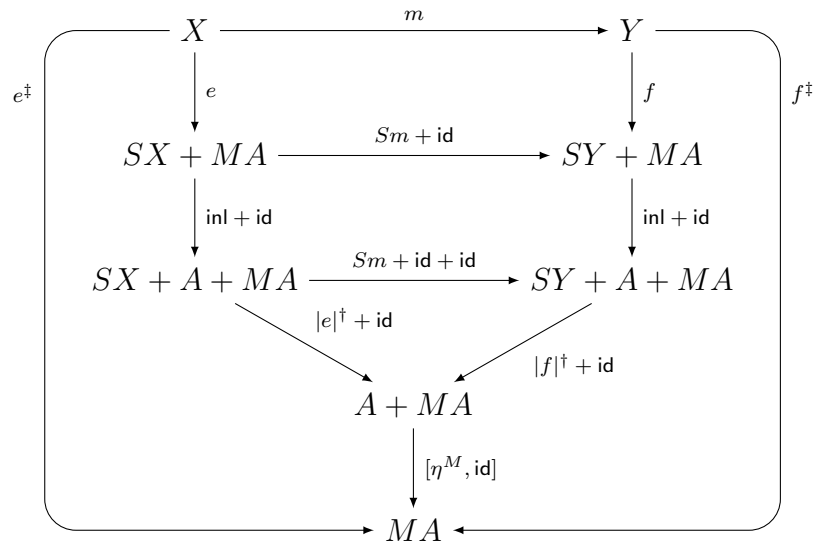
Now, we use it in the following calculation:

$$
\begin{aligned}
& e^\ddagger \\
= \quad & \text{(definition of } (\text{-})^\ddagger \text{)} \\
& [\eta^M, \mathsf{id}] \cdot ((|e|^\dagger \cdot \mathsf{inl}) + \mathsf{id}) \cdot e \\
= \quad & \text{(solution property of } (\text{-})^\dagger \text{)} \\
& [\eta^M, \mathsf{id}] \cdot (([a, \mathsf{id}] \cdot (S|e|^\dagger + \mathsf{id}) \cdot |e| \cdot \mathsf{inl}) + \mathsf{id}) \cdot e \\
= \quad & \text{(definition of } |\text{-}| \text{)} \\
& [\eta^M, \mathsf{id}] \cdot (([a, \mathsf{id}] \cdot (S|e|^\dagger + \mathsf{id}) \cdot ((\mathsf{flatr} \cdot Se) + \mathsf{id}) \cdot \mathsf{inl}) + \mathsf{id}) \cdot e
\end{aligned}
$$

$$
\begin{aligned}
&= \quad \text{(coproducts)} \\
&\quad [\eta^M, \mathsf{id}] \cdot ((a \cdot S|e|^\dagger \cdot \mathsf{flatr} \cdot Se) + \mathsf{id}) \cdot e \\
&= \quad \text{(solution property of (-)}^\dagger \text{)} \\
&\quad [\eta^M, \mathsf{id}] \cdot ((a \cdot S([a, \mathsf{id}] \cdot (S|e|^\dagger + \mathsf{id}) \cdot |e|) \cdot \mathsf{flatr} \cdot Se) + \mathsf{id}) \cdot e \\
&= \quad \text{(definition of |-|)} \\
&\quad [\eta^M, \mathsf{id}] \cdot ((a \cdot S([a, \mathsf{id}] \cdot (S|e|^\dagger + \mathsf{id}) \cdot ((\mathsf{flatr} \cdot Se) + \mathsf{id})) \cdot \mathsf{flatr} \cdot Se) + \mathsf{id}) \cdot e \\
&= \quad \text{(the above)} \\
&\quad [\eta^M, \mathsf{id}] \cdot ((a \cdot \mathsf{flat}' \cdot S[\eta^M, \mathsf{id}] \cdot S(a + \mathsf{id}) \cdot S(S|e|^\dagger + \mathsf{id}) \cdot S(\mathsf{flatr} + \mathsf{id}) \\
&\qquad\qquad \cdot S(Se + \mathsf{id}) \cdot Se) + \mathsf{id}) \cdot e \\
&= \quad \text{(coproducts)} \\
&\quad [\eta^M \cdot a \cdot \mathsf{flat}', \mathsf{id}] \cdot (S([\eta^M, \mathsf{id}] \cdot ((a \cdot S|e|^\dagger \cdot \mathsf{flatr} \\
&\qquad\qquad \cdot Se) + \mathsf{id}) \cdot e) + \mathsf{id}) \cdot e \\
&= \quad \text{(coproducts)} \\
&\quad [\eta^M \cdot a \cdot \mathsf{flat}', \mathsf{id}] \cdot (S([\eta^M, \mathsf{id}] \cdot (([a, \mathsf{id}] \cdot (S|e|^\dagger + \mathsf{id}) \cdot (\mathsf{flatr} + \mathsf{id}) \\
&\qquad\qquad \cdot (Se + \mathsf{id}) \cdot \mathsf{inl}) + \mathsf{id}) \cdot e) + \mathsf{id}) \cdot e \\
&= \quad \text{(definition of |-|)} \\
&\quad [\eta^M \cdot a \cdot \mathsf{flat}', \mathsf{id}] \cdot (S([\eta^M, \mathsf{id}] \cdot (([a, \mathsf{id}] \cdot (S|e|^\dagger + \mathsf{id}) \cdot |e| \cdot \mathsf{inl}) + \mathsf{id}) \cdot e) + \mathsf{id}) \cdot e \\
&= \quad \text{(solution property of (-)}^\dagger \text{)} \\
&\quad [\eta^M \cdot a \cdot \mathsf{flat}', \mathsf{id}] \cdot (S([\eta^M, \mathsf{id}] \cdot ((|e|^\dagger \cdot \mathsf{inl}) + \mathsf{id}) \cdot e) + \mathsf{id}) \cdot e \\
&= \quad \text{(definitions of } a' \text{ and (-)}^\ddagger \text{)} \\
&\quad [a', \mathsf{id}] \cdot (Se^\ddagger + \mathsf{id}) \cdot e
\end{aligned}
$$

For functoriality, assume that $e : X \to SX + MA$ and $f : Y \to SY + MA$ are equation morphisms, and let $m : X \to Y$ be an $(S(\text{-}) + MA)$-coalgebra homomorphism.

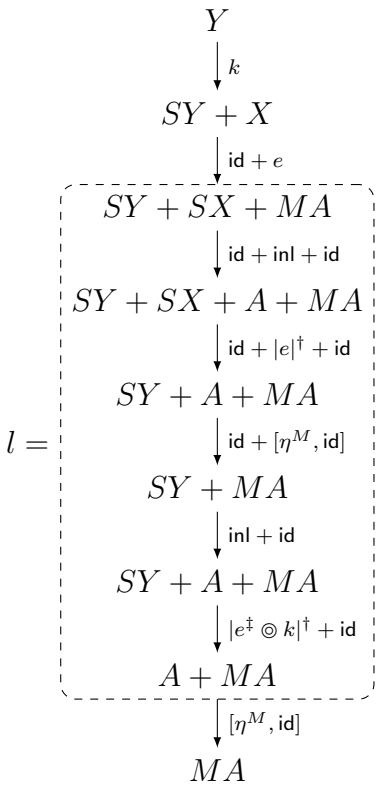
The following diagram commutes:



The top square commutes because m is a homomorphism. The middle square commutes due to the naturality of inl. The fact that the triangle commutes follows from Sm being a coalgebra homomorphism $|e| \to |f|$ (which is easy to verify) and then functoriality of $(\text{-})^\dagger$.

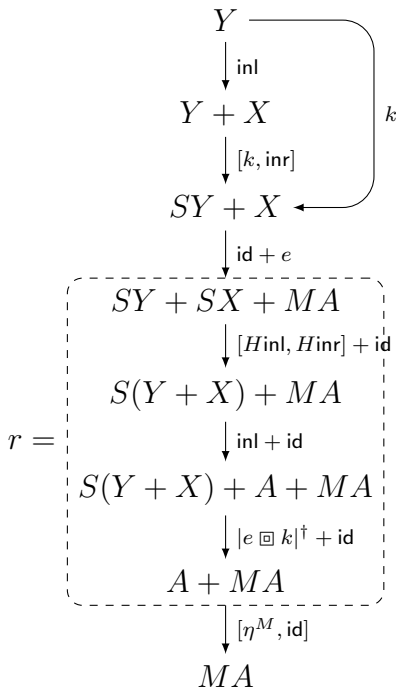
To prove compositionality, let $e : X \to SX + MA$ and $k : Y \to SY + X$ be equation morphisms. We unfold the definitions:

$(e^\ddagger \circledcirc k)^\ddagger =$



$l =$

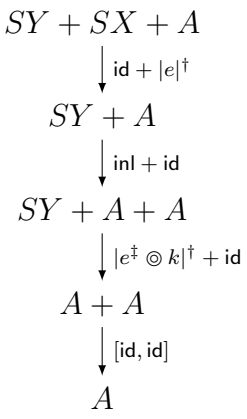
$(e \boxdot k)^\ddagger \cdot \mathsf{inl} =$



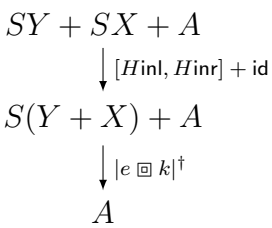
$r =$

To show that they are equal, it is sufficient to prove that the highlighted morphisms l and r are equal. Since they do not interfere with the last component of the coproduct, namely MA, we can ignore it. After simple transformations of coproducts, it means that it is sufficient to prove that the following morphisms are equal:

$l' =$



$r' =$



We calculate:

$$
\begin{aligned}
& l' \\
= \quad & \text{(definition)} \\
& [\mathsf{id}, \mathsf{id}] \cdot (|e^\ddagger \circledcirc k|^\dagger + \mathsf{id}) \cdot (\mathsf{inl} + \mathsf{id}) \cdot (\mathsf{id} + |e|^\dagger) \\
= \quad & \text{(solution)} \\
& [\mathsf{id}, \mathsf{id}] \cdot ([a, \mathsf{id}] + \mathsf{id}) \cdot (S|e^\ddagger \circledcirc k|^\dagger + \mathsf{id} + \mathsf{id}) \cdot (|e^\ddagger \circledcirc k| + \mathsf{id}) \cdot (\mathsf{inl} + \mathsf{id}) \cdot (\mathsf{id} + |e|^\dagger) \\
= \quad & \text{(definition of } |\text{(-)}| \text{)} \\
& [\mathsf{id}, \mathsf{id}] \cdot ([a, \mathsf{id}] + \mathsf{id}) \cdot (S|e^\ddagger \circledcirc k|^\dagger + \mathsf{id} + \mathsf{id}) \cdot ((\mathsf{flatr} \cdot S(e^\ddagger \circledcirc k)) + \mathsf{id} + \mathsf{id}) \cdot (\mathsf{inl} + \mathsf{id}) \cdot (\mathsf{id} + |e|^\dagger) \\
= \quad & \text{(coproducts)} \\
& [a, \mathsf{id}, \mathsf{id}] \cdot (S|e^\ddagger \circledcirc k|^\dagger + \mathsf{id} + \mathsf{id}) \cdot ((\mathsf{flatr} \cdot S(e^\ddagger \circledcirc k)) + \mathsf{id} + \mathsf{id}) \cdot (\mathsf{id} + \mathsf{inr}) \cdot (\mathsf{id} + |e|^\dagger) \\
= \quad & \text{(coproducts)} \\
& [a, \mathsf{id}] \cdot (S|e^\ddagger \circledcirc k|^\dagger + \mathsf{id}) \cdot ((\mathsf{flatr} \cdot S(e^\ddagger \circledcirc k)) + \mathsf{id}) \cdot (\mathsf{id} + |e|^\dagger) \\
= \quad & \text{(definition of } |\text{(-)}| \text{)} \\
& [a, \mathsf{id}] \cdot (S|e^\ddagger \circledcirc k|^\dagger + \mathsf{id}) \cdot (|e^\ddagger \circledcirc k| + \mathsf{id}) \cdot (\mathsf{id} + |e|^\dagger) \\
= \quad & \text{(solution)} \\
& |e^\ddagger \circledcirc k|^\dagger \cdot (\mathsf{id} + |e|^\dagger)
\end{aligned}
$$

Hence:

$$
l' = \quad
\begin{array}{c}
SY + SX + A \\
\downarrow{\scriptstyle \mathsf{id} + |e|^\dagger} \\
SY + A \\
\downarrow{\scriptstyle |e^\ddagger \circledcirc k|^\dagger} \\
A
\end{array}
$$

In exactly the same manner we calculate that the following holds:

$$|e^{\ddagger} \circledcirc k| \quad =$$

$$\begin{array}{c} SY + A \\ \downarrow Sk + \mathsf{id} \\ S(SY + X) + A \\ \downarrow S(\mathsf{id} + e) + \mathsf{id} \\ S(SY + SX + MA) + A \\ \downarrow S(\mathsf{id} + \mathsf{inl} + \mathsf{id}) + \mathsf{id} \\ S(SY + SX + A + MA) + A \\ \downarrow S(\mathsf{id} + |e|^{\dagger} + \mathsf{id}) + \mathsf{id} \\ S(SY + A + MA) + A \\ \downarrow S(\mathsf{id} + [\eta^{M}, \mathsf{id}]) + \mathsf{id} \\ S(SY + MA) + A \\ \downarrow \mathsf{flatr} + \mathsf{id} \\ S(SY + A) + A \end{array} \quad = \quad \begin{array}{c} SY + A \\ \downarrow Sk + \mathsf{id} \\ S(SY + X) + A \\ \downarrow S(\mathsf{id} + e) + \mathsf{id} \\ S(SY + SX + MA) + A \\ \downarrow \mathsf{flatr} + \mathsf{id} \\ S(SY + SX + A) + A \\ \downarrow S(\mathsf{id} + |e|^{\dagger}) + \mathsf{id} \\ S(SY + A) + A \end{array}$$

Thus, one can easily show by naturality of $+$ that $SY + SX + A \xrightarrow{\mathsf{id}+|e|^{\dagger}} SY + A \xrightarrow{|e^{\ddagger} \circledcirc k|} S(SY + A) + A$ is equal to:

$$p = \left[\begin{array}{c} SY + SX + A \\ \downarrow Sk + \mathsf{id} + \mathsf{id} \\ S(SY + X) + SX + A \\ \downarrow S(\mathsf{id} + e) + \mathsf{id} + \mathsf{id} \\ S(SY + SX + MA) + SX + A \\ \downarrow \mathsf{flatr} + \mathsf{id} + \mathsf{id} \\ S(SY + SX + A) + SX + A \end{array}\right]$$

$$q = \left[\begin{array}{c} S(SY + SX + A) + SX + A \\ \downarrow \mathsf{id} + |e|^{\dagger} \\ S(SY + SX + A) + A \end{array}\right]$$

$$\begin{array}{c} \downarrow S(\mathsf{id} + |e|^{\dagger}) + \mathsf{id} \\ S(SY + A) + A \end{array}$$

Hence, by functoriality, $l' = |e^{\ddagger} \circledcirc k|^{\dagger} \cdot (\mathsf{id} + |e|^{\dagger}) = (q \cdot p)^{\dagger} = (|e|^{\dagger} \circledcirc p)^{\dagger}$. By compositionality, $l' = (|e|^{\dagger} \circledcirc p)^{\dagger} = (|e| \boxdot p)^{\dagger} \cdot \mathsf{inl}$. Since p does not touch the second and third components of the coproduct, we have that $SY + SX + A + SX + A \xrightarrow{|e| \boxdot p} S(SY + SX + A + SX + A) + A \xrightarrow{S(\mathsf{id}+[\mathsf{id},\mathsf{id}])+\mathsf{id}} S(SY + SX + A) + A$ is equal to:

$$
\begin{array}{c}
SY + SX + A + SX + A \\
\downarrow \mathsf{id} + [\mathsf{id}, \mathsf{id}] \\
\boxed{\begin{array}{c}
SY + SX + A \\
\downarrow p \\
S(SY + SX + A) + SX + A \\
\downarrow \mathsf{id} + Se + \mathsf{id} \\
S(SY + SX + A) + S(SX + MA) + A \\
\downarrow \mathsf{id} + \mathsf{flatr} + \mathsf{id} \\
S(SY + SX + A) + S(SX + A) + A \\
\downarrow [H\mathsf{inl}, H\mathsf{inr}] + \mathsf{id} \\
S(SY + SX + A + SX + A) + A \\
\downarrow S(\mathsf{id} + [\mathsf{id}, \mathsf{id}]) + \mathsf{id} \\
S(SY + SX + A) + A
\end{array}} \\
\end{array}
\quad s =
$$

By functoriality, we get $l' = (|e| \circledcirc p)^\dagger \cdot \mathsf{inl} = s^\dagger \cdot (\mathsf{id} + [\mathsf{id}, \mathsf{id}]) \cdot \mathsf{inl} = s^\dagger$.

On the other side, we have:

$|e \circledcirc k| \cdot ([H\mathsf{inl}, H\mathsf{inr}] + \mathsf{id}) =$

$$
\begin{array}{c}
SY + SX + A \\
\downarrow [H\mathsf{inl}, H\mathsf{inr}] + \mathsf{id} \\
S(Y + X) + A \\
\downarrow S[k, \mathsf{inr}] + \mathsf{id} \\
S(SY + X) + A \\
\downarrow S(\mathsf{id} + e) + \mathsf{id} \\
S(SY + SX + MA) + A \\
\downarrow S([H\mathsf{inl}, H\mathsf{inr}] + \mathsf{id}) + \mathsf{id} \\
S(S(Y + X) + MA) + A \\
\downarrow \mathsf{flatr} + \mathsf{id} \\
S(S(Y + X) + A) + A
\end{array}
\quad = \quad
t =
\boxed{\begin{array}{c}
SY + SX + A \\
\downarrow [H\mathsf{inl}, H\mathsf{inr}] + \mathsf{id} \\
S(Y + X) + A \\
\downarrow S[k, \mathsf{inr}] + \mathsf{id} \\
S(SY + X) + A \\
\downarrow S(\mathsf{id} + e) + \mathsf{id} \\
S(SY + SX + MA) + A \\
\downarrow \mathsf{flatr} + \mathsf{id} \\
S(SY + SX + A) + A
\end{array}}
\begin{array}{c}
\\ \\ \\ \\ \\ \\ \\ \\ \\
\downarrow S([H\mathsf{inl}, H\mathsf{inr}] + \mathsf{id}) + \mathsf{id} \\
S(S(Y + X) + A) + A
\end{array}
$$

By functoriality, this means that $r' = |e \circledcirc k|^\dagger \cdot ([H\mathsf{inl}, H\mathsf{inr}] + \mathsf{id}) = t^\dagger$. By a coproduct and monad calculation, one can show that $s = t$, which yields $l' = s^\dagger = t^\dagger = r'$.

The assignment is an endofunctor. Let $h : \langle A, a, (\text{-})^\dagger \rangle \to \langle B, b, (\text{-})^\flat \rangle$ be a solution-preserving morphism. We need to show that Mh is also solution preserving

$\langle MA, a', (\text{-})^\ddagger\rangle \to \langle MB, b', (\text{-})^\sharp\rangle$. Let $e : X \to SX + MA$ be a flat equation morphism. We need to show that $(Mh \circledcirc e)^\sharp = Mh \cdot e^\ddagger$. To distinguish the $|(\text{-})|$ constructions for the algebras $\langle A, a, (\text{-})^\dagger\rangle$ and $\langle B, b, (\text{-})^\flat\rangle$, we use subscripts $|(\text{-})|_A$ and $|(\text{-})|_B$ respectively. We calculate:

(A)

$$
\begin{aligned}
& |Mh \circledcirc e|_B \\
= \quad & \text{(definition of } |\text{-}| \text{)} \\
& (\mathsf{flatr} + \mathsf{id}_B) \cdot (S(Mh \circledcirc e) + \mathsf{id}_B) \\
= \quad & \text{(definition of } \circledcirc \text{)} \\
& (\mathsf{flatr} + \mathsf{id}_B) \cdot (S((\mathsf{id} + Mh) \cdot e) + \mathsf{id}_B) \\
= \quad & \text{(functor)} \\
& (\mathsf{flatr} + \mathsf{id}_B) \cdot (S(\mathsf{id} + Mh) + \mathsf{id}_B) \cdot (Se + \mathsf{id}_B) \\
= \quad & \text{(naturality of } \mathsf{flatr} \text{)} \\
& (S(\mathsf{id} + h) + \mathsf{id}_B) \cdot (\mathsf{flatr} + \mathsf{id}_B) \cdot (Se + \mathsf{id}_B)
\end{aligned}
$$

(B)

$$
\begin{aligned}
& h \circledcirc |e|_A \\
= \quad & \text{(definition of } |\text{-}| \text{)} \\
& h \circledcirc ((\mathsf{flatr} + \mathsf{id}_A) \cdot (Se + \mathsf{id}_A)) \\
= \quad & \text{(definition of } \circledcirc \text{)} \\
& (\mathsf{id} + h) \cdot (\mathsf{flatr} + \mathsf{id}_A) \cdot (Se + \mathsf{id}_A)
\end{aligned}
$$

(C)

$$
\begin{aligned}
& |Mh \circledcirc e|_B \cdot (\mathsf{id} + h) \\
= \quad & \text{((A))} \\
& (S(\mathsf{id} + h) + \mathsf{id}_B) \cdot (\mathsf{flatr} + \mathsf{id}_B) \cdot (Se + \mathsf{id}_B) \cdot (\mathsf{id} + h) \\
= \quad & \text{(coproducts)} \\
& (S(\mathsf{id} + h) + \mathsf{id}_B) \cdot (\mathsf{id} + h) \cdot (\mathsf{flatr} + \mathsf{id}_A) \cdot (Se + \mathsf{id}_A) \\
= \quad & \text{((B))} \\
& (S(\mathsf{id} + h) + \mathsf{id}_B) \cdot (h \circledcirc |e|_A)
\end{aligned}
$$

By functoriality, (C) entails that $|Mh \circledcirc e|^\flat_B \cdot (\mathsf{id} + h) = (h \circledcirc |e|_A)^\flat$.

(D)

$$
\begin{aligned}
& |Mh \circledcirc e|^\flat_B \cdot \mathsf{inl}_{SX,B} \\
= \quad & \text{(solution property)} \\
& [b, \mathsf{id}_B] \cdot (S|Mh \circledcirc e|^\flat_B + \mathsf{id}_B) \cdot |Mh \circledcirc e|_B \cdot \mathsf{inl}_{SX,B} \\
= \quad & \text{((A))} \\
& [b, \mathsf{id}_B] \cdot (S|Mh \circledcirc e|^\flat_B + \mathsf{id}_B) \cdot (S(\mathsf{id} + h) + \mathsf{id}_B) \cdot (\mathsf{flatr} + \mathsf{id}_B) \cdot (Se + \mathsf{id}_B) \cdot \mathsf{inl}_{SX,B} \\
= \quad & \text{(coproducts)} \\
& b \cdot S|Mh \circledcirc e|^\flat_B \cdot S(\mathsf{id} + h) \cdot \mathsf{flatr} \cdot Se \\
= \quad & \text{(functoriality by (C))} \\
& b \cdot S(h \circledcirc |e|_A)^\flat \cdot \mathsf{flatr} \cdot Se \\
= \quad & \text{(coproducts)} \\
& [b, \mathsf{id}_B] \cdot (S(h \circledcirc |e|_A)^\flat + \mathsf{id}_B) \cdot (\mathsf{id} + h) \cdot (\mathsf{flatr} + \mathsf{id}_A) \cdot (Se + \mathsf{id}_A) \cdot \mathsf{inl}_{SX,A} \\
= \quad & \text{((B))} \\
& [b, \mathsf{id}_B] \cdot (S(h \circledcirc |e|_A)^\flat + \mathsf{id}_B) \cdot (h \circledcirc |e|_A) \cdot \mathsf{inl}_{SX,A} \\
= \quad & \text{(solution property)} \\
& (h \circledcirc |e|_A)^\flat \cdot \mathsf{inl}_{SX,A}
\end{aligned}
$$

Finally:

$$
\begin{aligned}
& (Mh \circledcirc e)^\sharp \\
= \quad & \text{(definition of (-)}^\sharp \text{)} \\
& [\eta^M, \mathsf{id}] \cdot (|Mh \circledcirc e|^\flat_B + \mathsf{id}) \cdot (\mathsf{inl}_{SX,B} + \mathsf{id}) \cdot (Mh \circledcirc e) \\
= \quad & \text{((D))} \\
& [\eta^M, \mathsf{id}] \cdot ((h \circledcirc |e|_A)^\flat + \mathsf{id}) \cdot (\mathsf{inl}_{SX,A} + \mathsf{id}) \cdot (Mh \circledcirc e) \\
= \quad & \text{(} h \text{ preserves solutions)} \\
& [\eta^M, \mathsf{id}] \cdot ((h \cdot |e|^\dagger_A) + \mathsf{id}) \cdot (\mathsf{inl}_{SX,A} + \mathsf{id}) \cdot (Mh \circledcirc e) \\
= \quad & \text{(definition of } \circledcirc \text{)} \\
& [\eta^M, \mathsf{id}] \cdot ((h \cdot |e|^\dagger_A) + \mathsf{id}) \cdot (\mathsf{inl}_{SX,A} + \mathsf{id}) \cdot (\mathsf{id} + Mh) \cdot e \\
= \quad & \text{(naturality of } \eta^M \text{ and coproducts)}
\end{aligned}
$$

$$
\begin{aligned}
& Mh \cdot [\eta^M, \mathsf{id}] \cdot (|e|_A^\dagger + \mathsf{id}) \cdot (\mathsf{inl}_{SX,A} + \mathsf{id}) \cdot e \\
= & \quad \text{(definition of (-)}^\ddagger \text{)} \\
& Mh \cdot e^\ddagger
\end{aligned}
$$

The assignment is a monad. All we need to do is to prove that both η^M and μ^M preserve solutions. Below, we show the case for μ^M and leave η^M to the reader. Let $h : X \to SX + M^2A$ be an equation morphism in $a'' = (a')'$. We denote its solution as $h^\sharp$. We need to show that $\mu^M \cdot h^\sharp = (\mu^M \circledcirc h)^\ddagger$.

First, consider the morphism $||h|| : S(SX+MA)+A \to S(S(SX+MA)+A)+A$ (which is the operation |-| applied twice to h). Using naturality and basic coproduct manipulations, one can prove that $||h|| \cdot ((\mathsf{flatr} \cdot Sh) + \mathsf{id}) = (S((\mathsf{flatr} \cdot Sh) + \mathsf{id}) + \mathsf{id}) \cdot ((\mathsf{flatr} \cdot \mathsf{flatr} \cdot Sh) + \mathsf{id})$. Using functoriality, it means that $||h||^\dagger \cdot ((\mathsf{flatr} \cdot Sh) + \mathsf{id}) = ((\mathsf{flatr} \cdot \mathsf{flatr} \cdot Sh) + \mathsf{id})^\dagger$. We use this equality in the following calculation:

$$
\begin{aligned}
& ||h||^\dagger \cdot ((\mathsf{flatr} \cdot Sh) + \mathsf{id}) \\
= & \quad \text{(the above)} \\
& ((\mathsf{flatr} \cdot \mathsf{flatr} \cdot Sh) + \mathsf{id})^\dagger \\
= & \quad \text{(monad laws)} \\
& ((\mathsf{flatr} \cdot S(\mathsf{id} + \mu^M) \cdot Sh) + \mathsf{id})^\dagger \\
= & \quad \text{(definition of } \circledcirc \text{)} \\
& ((\mathsf{flatr} \cdot S(\mu^M \circledcirc h)) + \mathsf{id})^\dagger \\
= & \quad \text{(definition of |-|)} \\
& |\mu^M \circledcirc h|^\dagger
\end{aligned}
$$

We can now prove that μ^M preserves solutions. We calculate:

$$
\begin{aligned}
& \mu^M \cdot h^\sharp \\
= \quad & \text{(definition of (-)}^\sharp \text{)} \\
& \mu^M \cdot [\eta^M \cdot |h|^\ddagger \cdot \mathsf{inl}, \mathsf{id}] \cdot h \\
= \quad & \text{(definition of (-)}^\ddagger \text{)} \\
& \mu^M \cdot [\eta^M \cdot [\eta^M \cdot ||h||^\dagger \cdot \mathsf{inl}, \mathsf{id}] \cdot |h| \cdot \mathsf{inl}, \mathsf{id}] \cdot h \\
= \quad & \text{(definition of |-|)} \\
& \mu^M \cdot [\eta^M \cdot [\eta^M \cdot ||h||^\dagger \cdot \mathsf{inl}, \mathsf{id}] \cdot ((\mathsf{flatr} \cdot Sh) + \mathsf{id}) \cdot \mathsf{inl}, \mathsf{id}] \cdot h \\
= \quad & \text{(coproducts)} \\
& \mu^M \cdot [\eta^M \cdot \eta^M \cdot ||h||^\dagger \cdot \mathsf{inl} \cdot \mathsf{flatr} \cdot Sh, \mathsf{id}] \cdot h \\
= \quad & \text{(coproducts and monad laws)} \\
& [\eta^M \cdot ||h||^\dagger \cdot \mathsf{inl} \cdot \mathsf{flatr} \cdot Sh, \mu^M] \cdot h \\
= \quad & \text{(naturality of } \mathsf{inl} \text{)} \\
& [\eta^M \cdot ||h||^\dagger \cdot ((\mathsf{flatr} \cdot Sh) + \mathsf{id}) \cdot \mathsf{inl}, \mu^M] \cdot h \\
= \quad & \text{(the above)} \\
& [\eta^M \cdot |\mu^M \circledcirc h|^\dagger \cdot \mathsf{inl}, \mu^M] \cdot h \\
= \quad & \text{(coproducts and definition of } \circledcirc \text{)} \\
& [\eta^M \cdot |\mu^M \circledcirc h|^\dagger \cdot \mathsf{inl}, \mathsf{id}] \cdot (\mu^M \circledcirc h) \\
= \quad & \text{(definition of (-)}^\ddagger \text{)} \\
& (\mu^M \circledcirc h)^\ddagger
\end{aligned}
$$

A.4 Direct definition of the monadic structure of coinductive resumptions

Step 1: A lifting to the category of Elgot algebras. Let M be a monad and S be a right M-module. We start with the following lifting of M to the category $\textsc{Elgot}(S)$ of Elgot algebras for the endofunctor S:

$$\lceil M \rceil \langle A, a : SA \to A, (\text{-})^\dagger \rangle = \langle MA,\, a' : SMA \to MA,\, (\text{-})^\ddagger \rangle,$$

where

$$a' = \left(SMA \xrightarrow{\overline{\mu}^S_A} SA \xrightarrow{a} A \xrightarrow{\eta^M_A} MA \right)$$

For a flat equation morphism $e : X \to SX + MA$, we define an auxiliary morphism $|e|$ and the solution $e^\ddagger$:

$$|e| = \Big(SX + A \xrightarrow{Se+\mathsf{id}} S(SX + MA) + A \xrightarrow{\mathsf{flatr}_{SX,A}+\mathsf{id}} S(SX + A) + A\Big)$$
$$e^\ddagger = \Big(X \xrightarrow{e} SX + MA \xrightarrow{\mathsf{inl}+\mathsf{id}} SX + A + MA \xrightarrow{|e|^\dagger+\mathsf{id}} A + MA \xrightarrow{[\eta^M_A,\mathsf{id}]} MA\Big)$$

Step 2: A lifting to the category of Eilenberg-Moore algebras of the free cim. Consider the (monadic) adjunction $F^{\mathrm{Elg}} \dashv U^{\mathrm{Elg}}$. The left adjoint is given on objects as:

$$F^{\mathrm{Elg}}A = \langle S^\infty A,\, a : SS^\infty A \to S^\infty A,\, (\text{-})^\dagger\rangle,$$

where the action is given as follows ($\xi_A : S^\infty A \to SS^\infty + A$ is the final coalgebra action):

$$a = \Big(SS^\infty A \xrightarrow{\mathsf{inl}} SS^\infty A + A \xrightarrow{\xi_A^{-1}} S^\infty A\Big),$$

and for a flat equation morphism $e : X \to SX + S^\infty A$, the solution is defined via the unique homomorphism from the equation morphism (understood as a $(S(\text{-}) + S^\infty A)$-coalgebra) to the final coalgebra:

$$e^\dagger = \Big(X \xrightarrow{[\![e]\!]} S^\infty S^\infty A \xrightarrow{\mu^\infty_A} S^\infty A\Big)$$

The counit of the adjunction is given as the solution to the final coalgebra action:

$$\varepsilon^{\mathrm{Elg}}_{\langle A,a,(\text{-})^\dagger\rangle} = \xi^\dagger_A : S^\infty A \to A$$

Thus, the comparison functor $\Phi : \textsc{Elgot}(S) \to \textsc{MAlg}(S^\infty)$ is given on objects as:

$$\Phi\langle A, a, (\text{-})^\dagger\rangle = \langle A, \varepsilon^{\mathrm{Elg}}_{\langle A,a,(\text{-})^\dagger\rangle}\rangle = \langle A, \xi^\dagger_A\rangle$$

We need the inverse of Φ. Our candidate is given on objects as:

$$\Phi^{-1}\langle A, m : S^\infty A \to A\rangle = \langle A,\, SA \xrightarrow{\mathsf{emb}_A} S^\infty A \xrightarrow{m} A,\, (\text{-})^\ddagger\rangle,$$

where the solution to a flat equation morphism $e : X \to SX + A$ is given as:

$$e^\ddagger = \Big(X \xrightarrow{[\![e]\!]} S^\infty A \xrightarrow{m} A\Big)$$

To prove that it indeed is an inverse of Φ, we use the fact that for any isomorphism f, if $g \cdot f = \mathsf{id}$, then $g = f^{-1}$:

$$\begin{aligned}
&\Phi^{-1}\Phi\langle A, k, (\text{-})^\dagger\rangle = \Phi^{-1}\langle A, \xi^\dagger\rangle = \langle A,\ \xi^\dagger \cdot \mathsf{emb}, (\text{-})^\ddagger\rangle \\
=\quad & \text{(solution property for } (\text{-})^\dagger\text{, twice)} \\
&\langle A,\ [k, \mathsf{id}] \cdot (S([k, \mathsf{id}] \cdot (S\xi^\dagger + \mathsf{id}) \cdot \xi) + \mathsf{id}) \cdot \xi \cdot \mathsf{emb},\ (\text{-})^\ddagger\rangle \\
=\quad & \text{(calculation)} \\
&\langle A, k, (\text{-})^\ddagger\rangle
\end{aligned}$$

In this case, since $e^\ddagger$ is defined as $\xi^\dagger \cdot [\![e]\!]$, by unfolding one step of the computation, we get $\xi \cdot [\![e]\!] = \xi \cdot \xi^{-1} \cdot (S[\![e]\!] + \mathsf{id}) \cdot e = (S[\![e]\!] + \mathsf{id}) \cdot e$. So, by functoriality, $e^\ddagger = \xi^\dagger \cdot [\![e]\!] = e^\dagger$.

Putting the pieces together, we can define the corresponding lifting of M to $\textsc{Malg}(S^\infty)$, denoted as $\widehat{M}$, as:

$$\begin{aligned}
\widehat{M}\langle A,\ m : S^\infty A \to A\rangle &= \Phi\lceil M\rceil\Phi^{-1}\langle A, m\rangle \\
&= \Phi\lceil M\rceil\langle A,\ m \cdot \mathsf{emb}_A, (\text{-})^\ddagger\rangle \\
&= \Phi\langle MA,\ \eta^M_A \cdot m \cdot \mathsf{emb}_A \cdot \overline{\mu}^S_A,\ (\text{-})^\dagger\rangle \\
&= \langle MA,\ \xi^\dagger_{MA}\rangle,
\end{aligned}$$

where a solution to a flat equation morphism $e : X \to SX + MA$ is given as:

$$\begin{aligned}
|e| &= \Big(SX + A \xrightarrow{Se+\mathsf{id}} S(SX + MA) + A \xrightarrow{\mathsf{flatr}_{SX,A}+\mathsf{id}} S(SX + A) + A\Big) \\
e^\dagger &= \Big(X \xrightarrow{e} SX + MA \xrightarrow{\mathsf{inl}+\mathsf{id}} SX + A + MA \xrightarrow{|e|^\ddagger+\mathsf{id}} A + MA \xrightarrow{[\eta^M_A,\mathsf{id}]} MA\Big) \\
&= \Big(X \xrightarrow{e} SX + MA \xrightarrow{\mathsf{inl}+\mathsf{id}} SX + A + MA \xrightarrow{[\![|e|]\!]+\mathsf{id}} S^\infty A + MA \\
&\qquad\qquad \xrightarrow{m+\mathsf{id}} A + MA \xrightarrow{[\eta^M_A,\mathsf{id}]} MA\Big)
\end{aligned}$$

The solution of ξ_{MA} is thus given as:

$$\begin{aligned}
|\xi_{MA}| &= \Big(SS^\infty MA + A \xrightarrow{S\xi_{MA}+\mathsf{id}} S(SS^\infty MA + MA) + A \\
&\qquad\qquad \xrightarrow{\mathsf{flatr}+\mathsf{id}} S(SS^\infty MA + A) + A\Big) \\
\xi^\dagger_{MA} &= \Big(S^\infty MA \xrightarrow{\xi_{MA}} SS^\infty MA + MA \xrightarrow{\mathsf{inl}+\mathsf{id}} SS^\infty MA + A + MA \\
&\qquad\qquad \xrightarrow{[\![|\xi_{MA}|]\!]+\mathsf{id}} S^\infty A + MA \xrightarrow{m+\mathsf{id}} A + MA \xrightarrow{[\eta^M_A,\mathsf{id}]} MA\Big)
\end{aligned}$$

Step 3: A distributive law between monads. Consider the free Eilenberg-Moore algebra $\langle S^\infty A,\, \mu_A^\infty : S^\infty S^\infty A \to S^\infty A\rangle$ and its $\widehat{M}$-image $\langle MS^\infty A,\, \xi^\dagger_{MS^\infty A} : S^\infty MS^\infty A \to MS^\infty A\rangle$, where:

$$|\xi_{MS^\infty A}| = \Big(SS^\infty MS^\infty A + S^\infty A \xrightarrow{S\xi_{MS^\infty A}+\mathsf{id}} S(SS^\infty MS^\infty A + MS^\infty A) + S^\infty A \xrightarrow{\mathsf{flatr}+\mathsf{id}} S(SS^\infty MA + S^\infty A) + S^\infty A\Big)$$

$$\xi^\dagger_{MS^\infty A} = \Big(S^\infty MS^\infty A \xrightarrow{\xi_{MS^\infty A}} SS^\infty MS^\infty A + MS^\infty A \xrightarrow{\mathsf{inl}+\mathsf{id}} SS^\infty MS^\infty A + S^\infty A + MS^\infty A \xrightarrow{[\![|\xi_{MS^\infty A}|]\!]+\mathsf{id}} S^\infty S^\infty A + MS^\infty A \xrightarrow{\mu_A^\infty+\mathsf{id}} S^\infty A + MS^\infty A \xrightarrow{[\eta_A^M,\mathsf{id}]} MS^\infty A\Big)$$

The distributive law is given as the following composition:

$$\lambda_A = \Big(S^\infty MA \xrightarrow{S^\infty M\eta_A^\infty} S^\infty MS^\infty A \xrightarrow{\xi^\dagger_{MS^\infty A}} MS^\infty A\Big)$$

This definition can be slightly simplified. We start with a lemma:

Lemma A.3. *The morphism* $[\![|\xi_{MX}|]\!] : SS^\infty MX + X \to S^\infty X$ *is natural in* X.

Proof. For a morphism $f : X \to Y$, we need to show that the following diagram commutes:

$$\begin{array}{ccc} SS^\infty MX + X & \xrightarrow{SS^\infty Mf + f} & SS^\infty MY + Y \\ \downarrow{\scriptstyle [\![|\xi_{MY}|]\!]} & & \downarrow{\scriptstyle [\![|\xi_{MY}|]\!]} \\ S^\infty X & \xrightarrow{S^\infty f} & S^\infty Y \end{array}$$

Let k be a morphism defined as:

$$k = \Big(SS^\infty MX + X \xrightarrow{S\xi_{MX}+\mathsf{id}} S(SS^\infty MX + MX) + X \xrightarrow{\mathsf{flatr}+f} S(SS^\infty MX + X) + Y\Big)$$

One can show that both paths of the diagram above are equal to the unique coalgebra homomorphism induced by k, that is, $[\![k]\!] : SS^\infty MX + X \to S^\infty Y$. It is enough to show that both paths are coalgebra homomorphisms from $\langle SS^\infty MX + X,\, k\rangle$ to the final coalgebra $\langle S^\infty Y, \xi_Y\rangle$, which is easy to verify using the computation law and simple diagram chasing. □

Using this lemma, we can 'swap' η_A^∞ and $[\![|\xi_{MA^\infty}|]\!]$. We obtain:

$$\lambda_A = \Big(S^\infty MA \xrightarrow{\xi_{MA}} SS^\infty MA + MA \xrightarrow{\mathsf{inl}+\mathsf{id}} SS^\infty MA + A + MA \xrightarrow{[\![|\xi_{MA}|]\!]+\mathsf{id}} S^\infty A + MA \xrightarrow{S^\infty\eta_A^\infty + M\eta_A^\infty} S^\infty S^\infty A + MS^\infty A \xrightarrow{\mu_A^\infty+\mathsf{id}} S^\infty A + MS^\infty A \xrightarrow{[\eta_A^M,\mathsf{id}]} MS^\infty A\Big)$$

The monad laws give us:

$$\lambda_A = \Big(S^\infty MA \xrightarrow{\xi_{MA}} SS^\infty MA + MA \xrightarrow{\mathsf{inl}+\mathsf{id}} SS^\infty MA + A + MA \xrightarrow{[\![|\xi_{MA}|]\!]+\mathsf{id}} S^\infty A + MA \xrightarrow{[\eta^M_{S^\infty A}, M\eta^\infty_A]} MS^\infty A\Big)$$

A.5 Proof of Lemma 5.11 (a)

First, we need the following technical lemmas:

Lemma A.4. *Let M be a monad idealised with $\overline{M}$, A and B be objects. The following diagram commutes:*

$$\begin{array}{ccc} \overline{M}(B + MA) & \xrightarrow{\sigma} & M(B + MA) \\ \downarrow{\scriptstyle \mathsf{flatr}} & & \downarrow{\scriptstyle \mathsf{flatr}} \\ \overline{M}(B + A) & \xrightarrow{\sigma} & M(B + A) \end{array}$$

Proof. We calculate:

$$\begin{aligned} & \sigma \cdot \mathsf{flatr} \\ = \quad & \text{(def. of } \mathsf{flatr} \text{)} \\ & \sigma \cdot \overline{\mu} \cdot \overline{M}[M\mathsf{inl}, M\mathsf{inr}] \cdot \overline{M}(\eta + \mathsf{id}) \\ = \quad & \text{(} \sigma \text{ is a module morphism)} \\ & \mu \cdot \sigma \cdot \overline{M}[M\mathsf{inl}, M\mathsf{inr}] \cdot \overline{M}(\eta + \mathsf{id}) \\ = \quad & \text{(naturality of } \sigma \text{)} \\ & \mu \cdot M[M\mathsf{inl}, M\mathsf{inr}] \cdot M(\eta + \mathsf{id}) \cdot \sigma \\ = \quad & \text{(def. of } \mathsf{flatr} \text{)} \\ & \mathsf{flatr} \cdot \sigma \end{aligned}$$

□

Lemma A.5. *Let M be a monad, A and B be objects, and $f : B \to MA$ be a morphism. Then, the following diagram commutes:*

$$\begin{array}{ccccccc} M(B + MA + A) & & & \xrightarrow{M[f, \mathsf{id}, \eta_A]} & & & M^2A \\ \downarrow{\scriptstyle M(\mathsf{id} + [\mathsf{id}, \eta_A])} & & & & & & \downarrow{\scriptstyle \mu_A} \\ M(B + MA) & \xrightarrow{\mathsf{flatr}} & M(B + A) & \xrightarrow{M[f, \eta_A]} & M^2A & \xrightarrow{\mu_A} & MA \end{array}$$

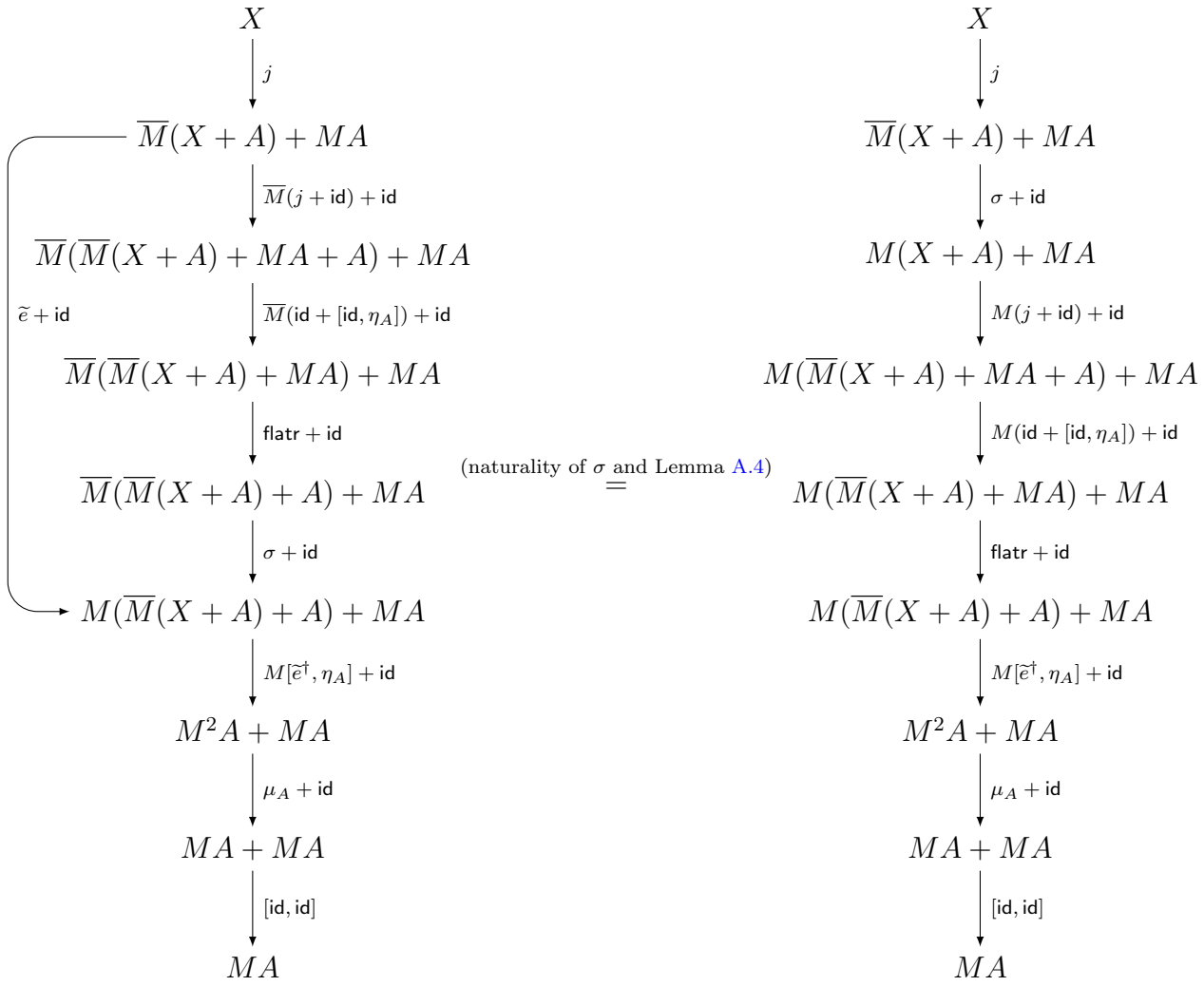
Proof. We calculate:

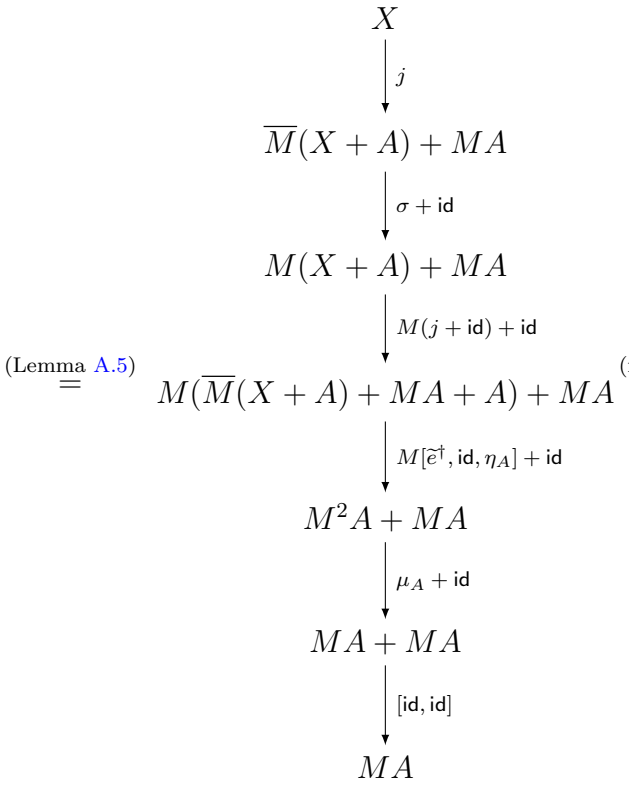
$$
\begin{aligned}
& \mu_A \cdot M[f, \eta_A] \cdot \mathsf{flatr} \cdot M(\mathsf{id} + [\mathsf{id}, \eta_A]) \\
= \quad & \text{(def. of } \mathsf{flatr} \text{)} \\
& \mu_A \cdot M[f, \eta_A] \cdot \mu_{B+A} \cdot M[M\mathsf{inl}, M\mathsf{inr}] \cdot M(\eta_B + \mathsf{id}) \cdot M(\mathsf{id} + [\mathsf{id}, \eta_A]) \\
= \quad & \text{(coproduct)} \\
& \mu_A \cdot M[f, \eta_A] \cdot \mu_{B+A} \cdot M[M\mathsf{inl}, M\mathsf{inr}] \cdot M(\eta_B + [\mathsf{id}, \eta_A]) \\
= \quad & \text{(coproduct)} \\
& \mu_A \cdot M[f, \eta_A] \cdot \mu_{B+A} \cdot M[M\mathsf{inl} \cdot \eta_B,\ M\mathsf{inr},\ M\mathsf{inr} \cdot \eta_A] \\
= \quad & \text{(naturality of } \mu \text{)} \\
& \mu_A \cdot \mu_{MA} \cdot MM[f, \eta_A] \cdot M[M\mathsf{inl} \cdot \eta_B,\ M\mathsf{inr},\ M\mathsf{inr} \cdot \eta_A] \\
= \quad & \text{(coproduct)} \\
& \mu_A \cdot \mu_{MA} \cdot M[M[f, \eta_A] \cdot M\mathsf{inl} \cdot \eta_B,\ M[f, \eta_A] \cdot M\mathsf{inr},\ M[f, \eta_A] \cdot M\mathsf{inr} \cdot \eta_A] \\
= \quad & \text{(coproduct)} \\
& \mu_A \cdot \mu_{MA} \cdot M[Mf \cdot \eta_B,\ M\eta_A,\ M\eta_A \cdot \eta_A] \\
= \quad & \text{(monad laws)} \\
& \mu_A \cdot M\mu_A \cdot M[Mf \cdot \eta_B,\ M\eta_A,\ M\eta_A \cdot \eta_A] \\
= \quad & \text{(coproduct)} \\
& \mu_A \cdot M[\mu_A \cdot Mf \cdot \eta_B,\ \mu_A \cdot M\eta_A,\ \mu_A \cdot M\eta_A \cdot \eta_A] \\
= \quad & \text{(naturality of } \eta \text{)} \\
& \mu_A \cdot M[\mu_A \cdot \eta_{MA} \cdot f,\ \mu_A \cdot M\eta_A,\ \mu_A \cdot M\eta_A \cdot \eta_A] \\
= \quad & \text{(monad laws)} \\
& \mu_A \cdot M[f, \mathsf{id}, \eta_A]
\end{aligned}
$$

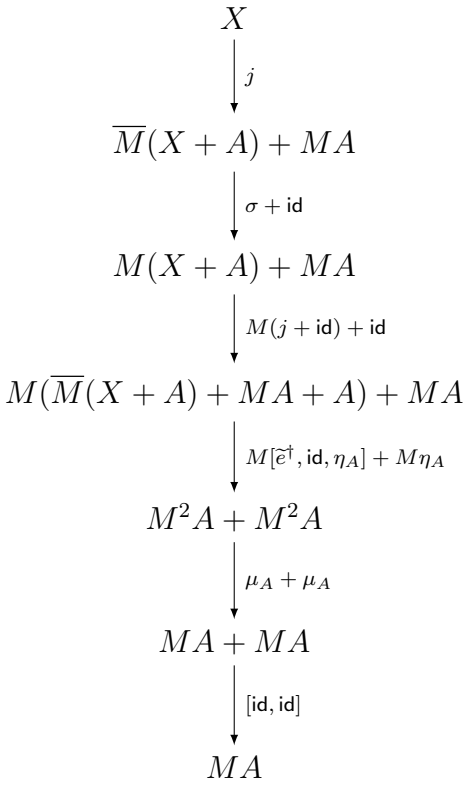
□

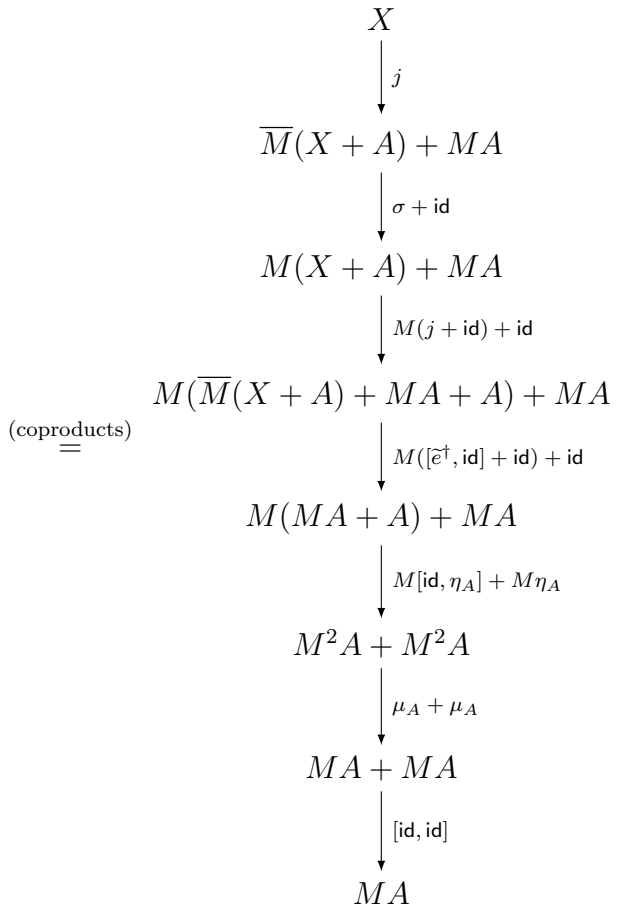
It is left to verify that (i) $e^\ddagger$ is indeed a solution, and that (ii) it is unique.
(i) We verify the solution property:

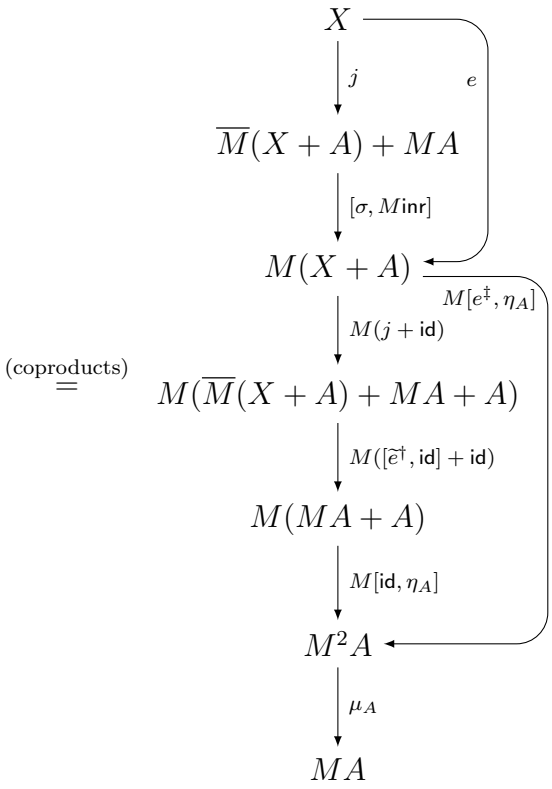
$$e^\ddagger \overset{\text{(def. of } e^\ddagger\text{)}}{=} [\widetilde{e}^\dagger, \mathsf{id}] \cdot j \overset{\text{(solution property of } \widetilde{e}^\dagger\text{)}}{=}$$


X
j
$\overline{M}(X+A)+MA$
$\overline{M}(j+\mathsf{id})+\mathsf{id}$
$\overline{M}(\overline{M}(X+A)+MA+A)+MA$
$\widetilde{e}+\mathsf{id}$
$\overline{M}(\mathsf{id}+[\mathsf{id},\eta_A])+\mathsf{id}$
$\overline{M}(\overline{M}(X+A)+MA)+MA$
$\mathsf{flatr}+\mathsf{id}$
$\overline{M}(\overline{M}(X+A)+A)+MA$
$\sigma+\mathsf{id}$
$M(\overline{M}(X+A)+A)+MA$
$M[\widetilde{e}^\dagger,\eta_A]+\mathsf{id}$
M^2A+MA
$\mu_A+\mathsf{id}$
$MA+MA$
$[\mathsf{id},\mathsf{id}]$
MA
(naturality of σ and Lemma A.4)
=
X
j
$\overline{M}(X+A)+MA$
$\sigma+\mathsf{id}$
$M(X+A)+MA$
$M(j+\mathsf{id})+\mathsf{id}$
$M(\overline{M}(X+A)+MA+A)+MA$
$M(\mathsf{id}+[\mathsf{id},\eta_A])+\mathsf{id}$
$M(\overline{M}(X+A)+MA)+MA$
$\mathsf{flatr}+\mathsf{id}$
$M(\overline{M}(X+A)+A)+MA$
$M[\widetilde{e}^\dagger,\eta_A]+\mathsf{id}$
M^2A+MA
$\mu_A+\mathsf{id}$
$MA+MA$
$[\mathsf{id},\mathsf{id}]$
MA


X
j
$\overline{M}(X+A)+MA$
$\sigma+\mathsf{id}$
$M(X+A)+MA$
$M(j+\mathsf{id})+\mathsf{id}$
(Lemma A.5)
=
$M(\overline{M}(X+A)+MA+A)+MA$
$M[\widetilde{e}^\dagger,\mathsf{id},\eta_A]+\mathsf{id}$
M^2A+MA
$\mu_A+\mathsf{id}$
$MA+MA$
$[\mathsf{id},\mathsf{id}]$
MA


X
j
$\overline{M}(X+A)+MA$
$\sigma+\mathsf{id}$
$M(X+A)+MA$
$M(j+\mathsf{id})+\mathsf{id}$
(monads)
=
$M(\overline{M}(X+A)+MA+A)+MA$
$M[\widetilde{e}^\dagger,\mathsf{id},\eta_A]+M\eta_A$
M^2A+M^2A
$\mu_A+\mu_A$
$MA+MA$
$[\mathsf{id},\mathsf{id}]$
MA

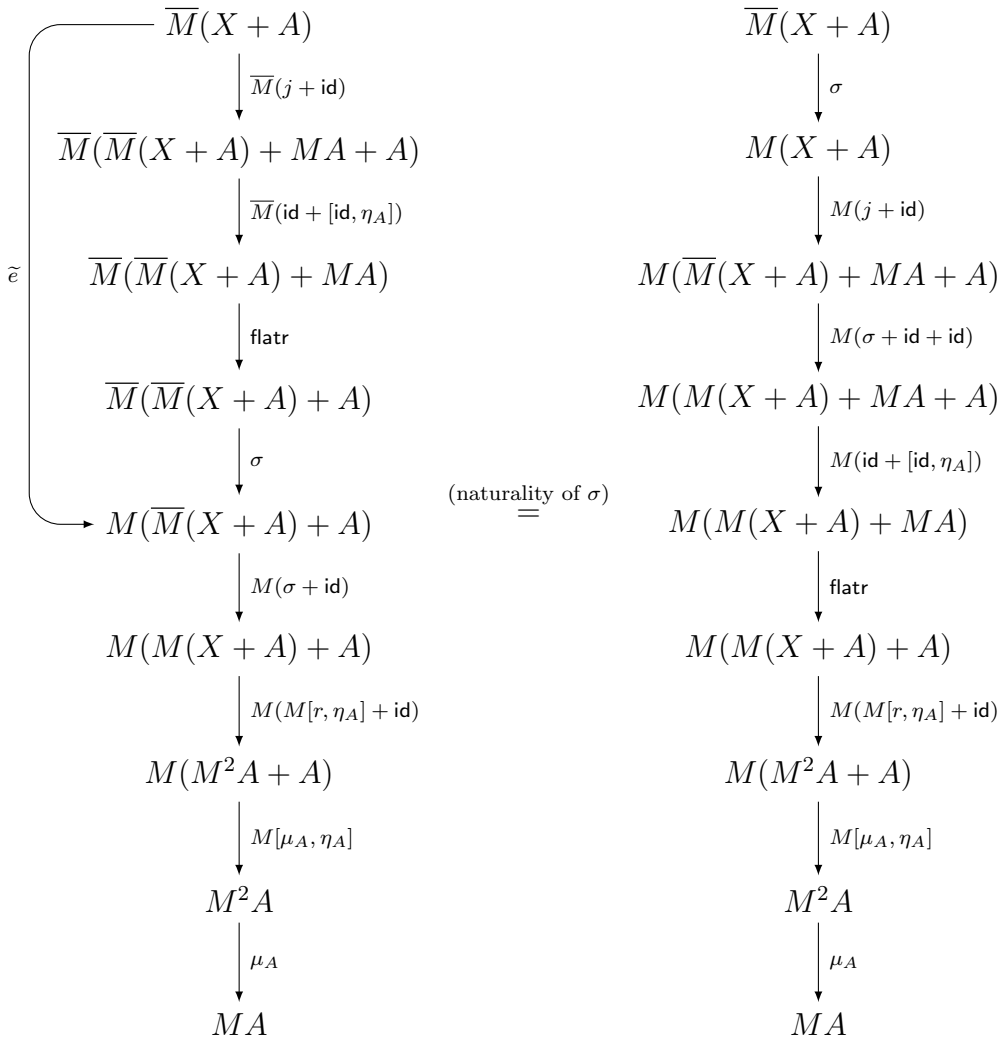

X
j
$\overline{M}(X+A)+MA$
$\sigma+\mathsf{id}$
$M(X+A)+MA$
$M(j+\mathsf{id})+\mathsf{id}$
$M(\overline{M}(X+A)+MA+A)+MA$
(coproducts)
=
$M([\widetilde{e}^\dagger,\mathsf{id}]+\mathsf{id})+\mathsf{id}$
$M(MA+A)+MA$
$M[\mathsf{id},\eta_A]+M\eta_A$
M^2A+M^2A
$\mu_A+\mu_A$
$MA+MA$
$[\mathsf{id},\mathsf{id}]$
MA


X
j
e
$\overline{M}(X+A)+MA$
$[\sigma,M\mathsf{inr}]$
$M(X+A)$
$M[e^\ddagger,\eta_A]$
(coproducts)
=
$M(j+\mathsf{id})$
$M(\overline{M}(X+A)+MA+A)$
$M([\widetilde{e}^\dagger,\mathsf{id}]+\mathsf{id})$
$M(MA+A)$
$M[\mathsf{id},\eta_A]$
M^2A
μ_A
MA

(ii) It is left to check that $e^\ddagger$ is a unique solution. Let $r : X \to MA$ be any solution of e. Unfolding the definition of $e^\ddagger$ and the solution property of r, we need to show that the following two morphisms are equal:

$$e^\ddagger = X \xrightarrow{j} \overline{M}(X+A) + MA \xrightarrow{[\widetilde{e}^\dagger, \mathsf{id}]} MA$$
$$r = X \xrightarrow{j} \overline{M}(X+A) + MA \xrightarrow{[\sigma, M\mathsf{inr}]} M(X+A) \xrightarrow{M[r,\eta_A]} M^2A \xrightarrow{\mu_A} MA$$

Thus, it is enough to show that $\widetilde{e}^\dagger = \mu_A \cdot M[r, \eta_A] \cdot \sigma$ and that $\mathsf{id} = \mu_A \cdot M[r, \eta_A] \cdot M\mathsf{inr}$. The latter is trivial. For the former, we show that $\mu_A \cdot M[r, \eta_A] \cdot \sigma$ is a solution of $\widetilde{e}$, and the conclusion follows from the uniqueness of solutions.



$$
\begin{array}{c}
\overline{M}(X+A) \\
\downarrow{\scriptstyle \sigma} \\
M(X+A) \\
\downarrow{\scriptstyle M(j+\mathsf{id})} \\
M(\overline{M}(X+A)+MA+A) \\
\downarrow{\scriptstyle M(\sigma+\mathsf{id}+\mathsf{id})} \\
M(M(X+A)+MA+A) \\
\downarrow{\scriptstyle M[\mu_A\cdot M[r,\eta_A],\ \mathsf{id},\ \eta_A]} \\
M^2A \\
\downarrow{\scriptstyle \mu_A} \\
MA
\end{array}
\quad
\overset{\text{(Lemma A.5)}}{=}
\quad
\begin{array}{c}
\overline{M}(X+A) \\
\downarrow{\scriptstyle \sigma} \\
M(X+A) \\
\downarrow{\scriptstyle M(j+\mathsf{id})} \\
M(\overline{M}(X+A)+MA+A) \\
\downarrow{\scriptstyle M(\sigma+\mathsf{id}+\mathsf{id})} \\
M(M(X+A)+MA+A) \\
\downarrow{\scriptstyle M([M[r,\eta_A],\ M\eta_A]+\mathsf{id})} \\
M(M^2A+A) \\
\downarrow{\scriptstyle M[\mu_A,\eta_A]} \\
M^2A \\
\downarrow{\scriptstyle \mu_A} \\
MA
\end{array}
\quad
\overset{\text{(copr.+monads)}}{=}
$$

$$
\begin{array}{c}
\overline{M}(X+A) \\
\downarrow{\scriptstyle \sigma} \\
M(X+A) \\
\downarrow{\scriptstyle M(j+\mathsf{id})} \\
M(\overline{M}(X+A)+MA+A) \\
\downarrow{\scriptstyle M(\sigma+\mathsf{id}+\mathsf{id})} \\
M(M(X+A)+MA+A) \\
\downarrow{\scriptstyle M([M[r,\eta_A],\ M[r,\eta_A]\cdot M\mathsf{inr}]+\mathsf{id})} \\
M(M^2A+A) \\
\downarrow{\scriptstyle M[\mu_A,\eta_A]} \\
M^2A \\
\downarrow{\scriptstyle \mu_A} \\
MA
\end{array}
\quad
\overset{\text{(coproducts)}}{=}
\quad
\begin{array}{c}
\overline{M}(X+A) \\
\downarrow{\scriptstyle \sigma} \\
M(X+A) \\
\downarrow{\scriptstyle M(j+\mathsf{id})} \\
M(\overline{M}(X+A)+MA+A) \\
\downarrow{\scriptstyle M([\sigma,M\mathsf{inr}]+\mathsf{id})} \\
M(M(X+A)+MA+A) \\
\downarrow{\scriptstyle M(M[r,\eta_A]+\mathsf{id})} \\
M(M^2A+A) \\
\downarrow{\scriptstyle M[\mu_A,\eta_A]} \\
M^2A \\
\downarrow{\scriptstyle \mu_A} \\
MA
\end{array}
\quad
\overset{\text{(coproducts)}}{=}
$$

$$
\begin{array}{ll}
 & \overline{M}(X+A) \\
 & \Big\downarrow \sigma \\
 & M(X+A) \\
\overset{\text{(solution property of } r)}{=} & \Big\downarrow M[r,\eta_A] \\
 & M^2A \\
 & \Big\downarrow \mu_A \\
 & MA
\end{array}
$$

A.6 Proof of Lemma 5.11 (b)

We need the following lemmas:

Lemma A.6. *Let M be a monad idealised with $\overline{M}$, A and B be objects. The following diagram commutes:*

$$
\begin{array}{ccc}
\overline{M}(MB+A) & \xrightarrow{\sigma} & M(MB+A) \\
\Big\downarrow \mathsf{flatl} & & \Big\downarrow \mathsf{flatl} \\
\overline{M}(B+A) & \xrightarrow{\sigma} & M(B+A)
\end{array}
$$

Proof. It is a symmetric case of Lemma A.4. □

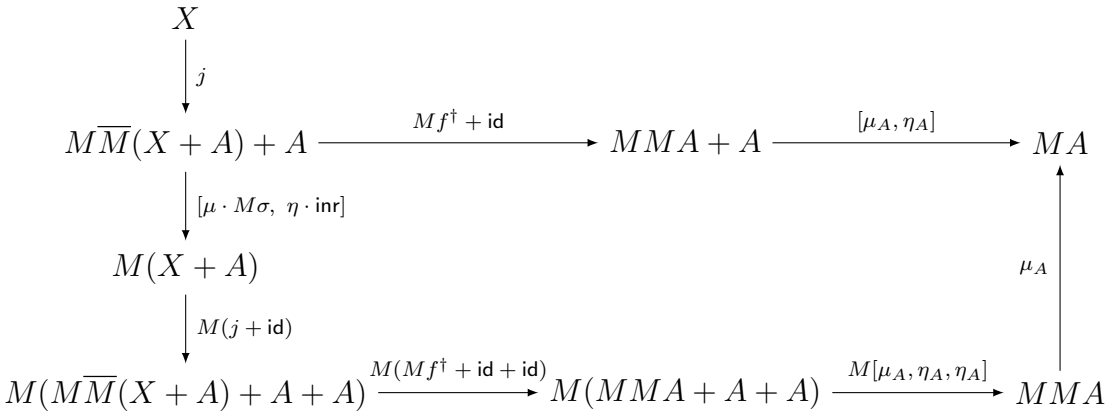
Lemma A.7. *Let M be a monad, A and B be objects, and $f : B \to MA$ be a morphism. Then, the following diagram commutes:*

$$
\begin{array}{ccccccc}
M(MB+A+A) & & \xrightarrow{\qquad\qquad M[\mu_A \cdot Mf,\ \eta_A,\ \eta_A] \qquad\qquad} & & & & M^2A \\
\Big\downarrow M(\mathsf{id}+[\mathsf{id},\mathsf{id}]) & & & & & & \Big\downarrow \mu_A \\
M(MB+A) & \xrightarrow{\mathsf{flatl}} & M(B+A) & \xrightarrow{M[f,\eta_A]} & M^2A & \xrightarrow{\mu_A} & MA
\end{array}
$$

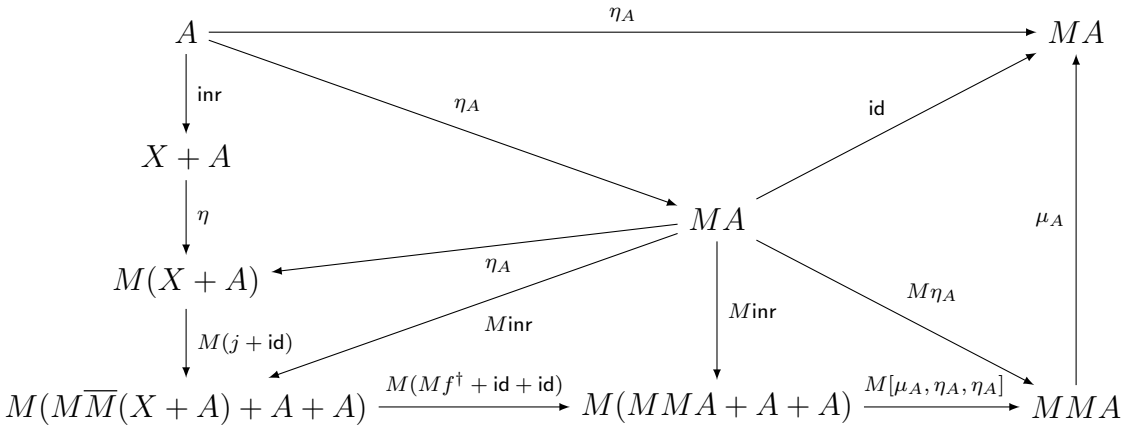
Proof. We calculate:

$$
\begin{aligned}
& \mu_A \cdot M[f, \eta_A] \cdot \mathsf{flatl} \cdot M(\mathsf{id} + [\mathsf{id}, \mathsf{id}]) \\
= \quad & (\text{ def. of } \mathsf{flatl} \ ) \\
& \mu_A \cdot M[f, \eta_A] \cdot \mu_{B+A} \cdot M[M\mathsf{inl}, M\mathsf{inr}] \cdot M(\mathsf{id} + \eta_A) \cdot M(\mathsf{id} + [\mathsf{id}, \mathsf{id}]) \\
= \quad & (\text{ naturality of } \mu \ ) \\
& \mu_A \cdot \mu_{MA} \cdot MM[f, \eta_A] \cdot M[M\mathsf{inl}, M\mathsf{inr}] \cdot M(\mathsf{id} + \eta_A) \cdot M(\mathsf{id} + [\mathsf{id}, \mathsf{id}]) \\
= \quad & (\text{ coproducts }) \\
& \mu_A \cdot \mu_{MA} \cdot M[M([f, \eta_A] \cdot \mathsf{inl}), M([f, \eta_A] \cdot \mathsf{inr})] \cdot M(\mathsf{id} + \eta_A) \cdot M(\mathsf{id} + [\mathsf{id}, \mathsf{id}]) \\
= \quad & (\text{ coproducts }) \\
& \mu_A \cdot \mu_{MA} \cdot M[Mf, M\eta_A] \cdot M(\mathsf{id} + \eta_A) \cdot M(\mathsf{id} + [\mathsf{id}, \mathsf{id}]) \\
= \quad & (\text{ monad laws }) \\
& \mu_A \cdot M\mu_A \cdot M[Mf, M\eta_A] \cdot M(\mathsf{id} + \eta_A) \cdot M(\mathsf{id} + [\mathsf{id}, \mathsf{id}]) \\
= \quad & (\text{ coproducts }) \\
& \mu_A \cdot M[\mu_A \cdot Mf, \ \mu_A \cdot M\eta_A] \cdot M(\mathsf{id} + \eta_A) \cdot M(\mathsf{id} + [\mathsf{id}, \mathsf{id}]) \\
= \quad & (\text{ monad laws }) \\
& \mu_A \cdot M[\mu_A \cdot Mf, \ \mathsf{id}] \cdot M(\mathsf{id} + \eta_A) \cdot M(\mathsf{id} + [\mathsf{id}, \mathsf{id}]) \\
= \quad & (\text{ coproducts }) \\
& \mu_A \cdot M[\mu_A \cdot Mf, \ \eta_A] \cdot M(\mathsf{id} + [\mathsf{id}, \mathsf{id}]) \\
= \quad & (\text{ coproducts }) \\
& \mu_A \cdot M[\mu_A \cdot Mf, \ \eta_A, \ \eta_A] \qquad \square
\end{aligned}
$$

It is left to prove that (i) $e^\ddagger$ is a solution of e, and that (ii) it is a unique solution.
(i) We need to show that the following diagram commutes:

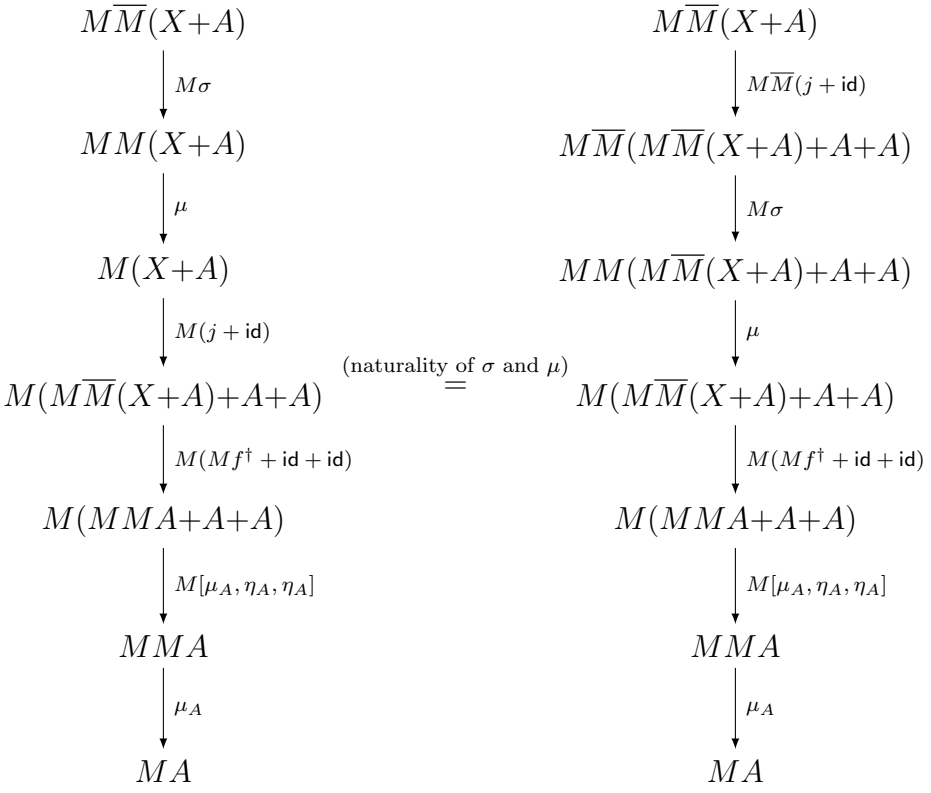


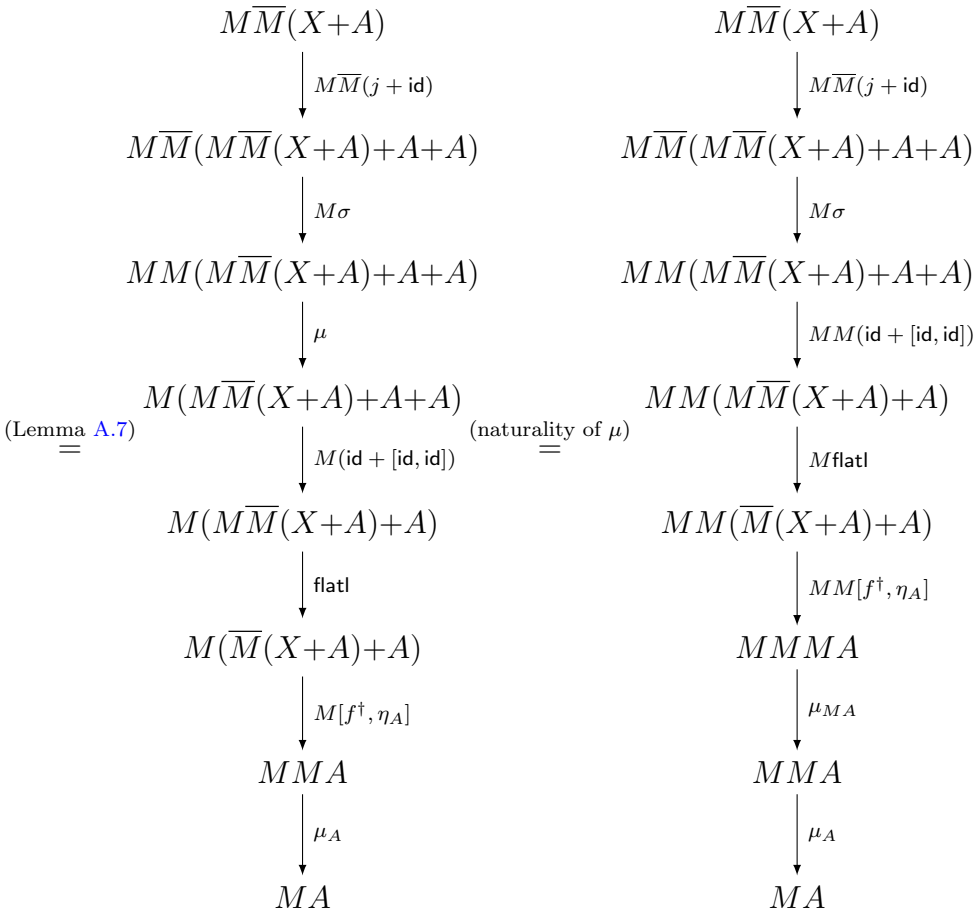
To see that the big heptagon commutes, we consider each component of the outer coproduct separately. The right-hand side component:

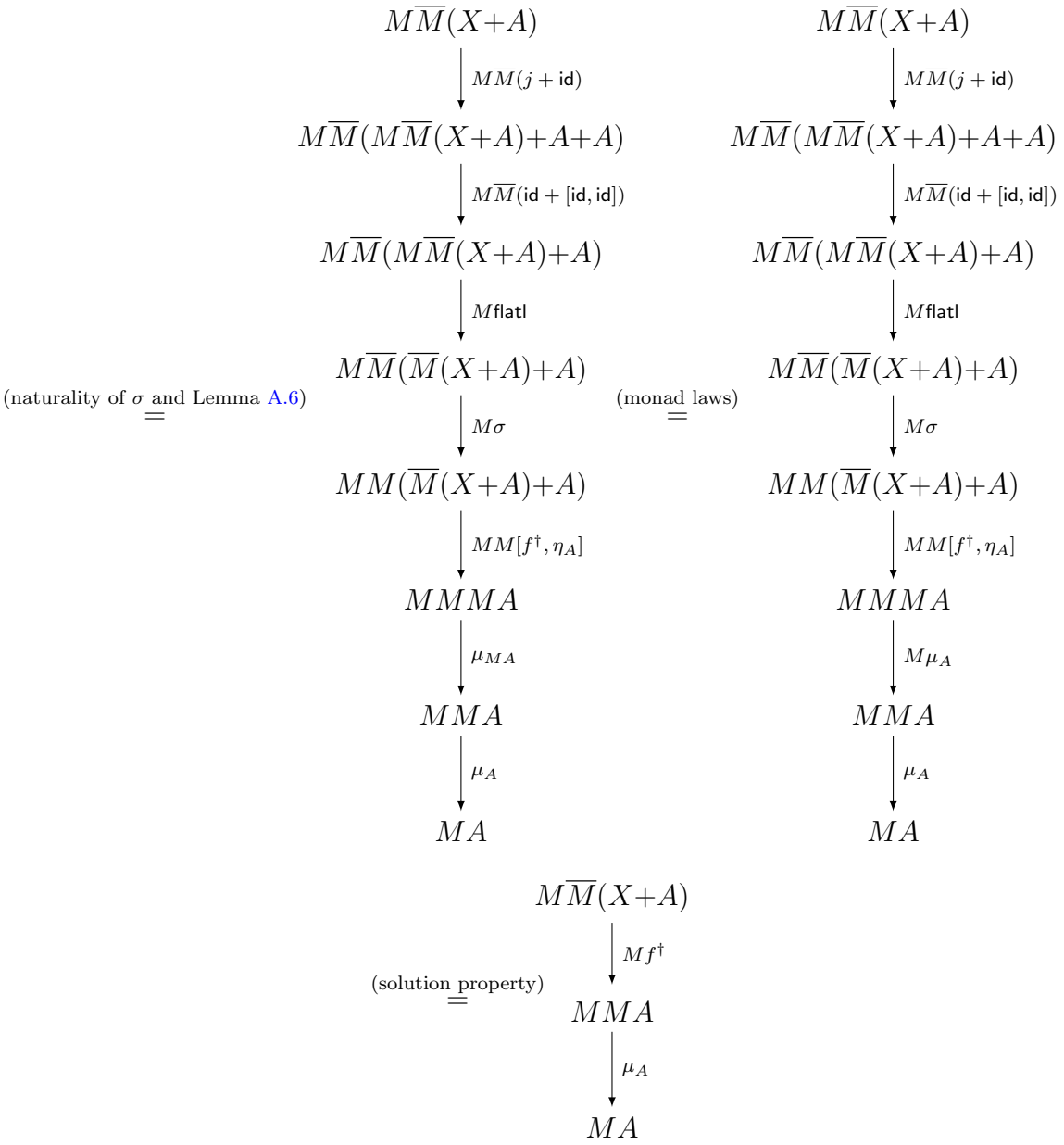


(Every part commutes from naturality, monad laws, or properties of coproducts.)

In case of the left-hand side component, we calculate as follows. We begin with the 'longer' path of the perimeter.



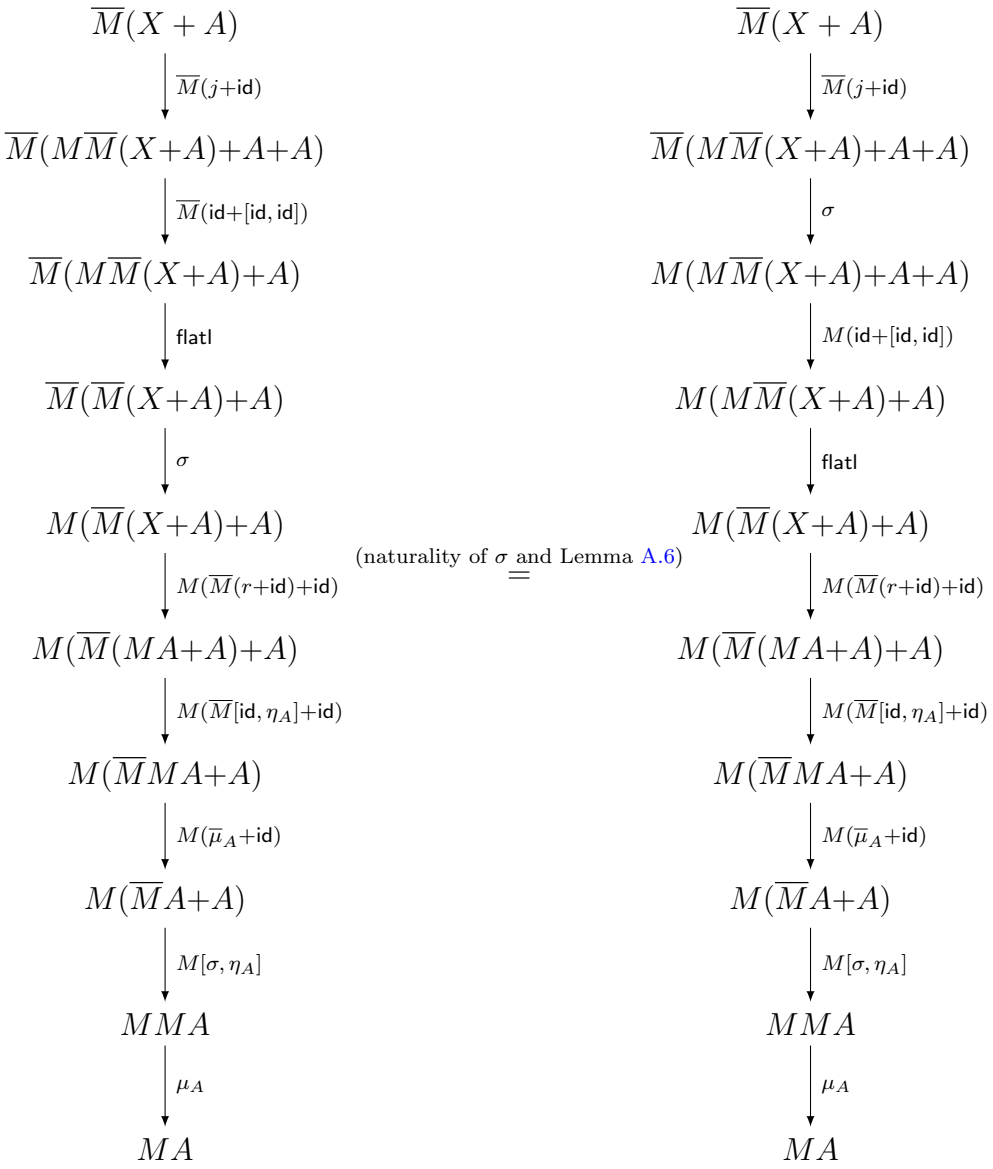

$M\overline{M}(X+A)$
$M\overline{M}(j+\mathsf{id})$
$M\overline{M}(M\overline{M}(X+A)+A+A)$
$M\sigma$
$MM(M\overline{M}(X+A)+A+A)$
μ
$M(M\overline{M}(X+A)+A+A)$
(Lemma A.7)
$M(\mathsf{id}+[\mathsf{id},\mathsf{id}])$
$M(M\overline{M}(X+A)+A)$
flatl
$M(\overline{M}(X+A)+A)$
$M[f^\dagger,\eta_A]$
MMA
μ_A
MA
$M\overline{M}(X+A)$
$M\overline{M}(j+\mathsf{id})$
$M\overline{M}(M\overline{M}(X+A)+A+A)$
$M\sigma$
$MM(M\overline{M}(X+A)+A+A)$
$MM(\mathsf{id}+[\mathsf{id},\mathsf{id}])$
(naturality of μ)
$MM(M\overline{M}(X+A)+A)$
$M\mathsf{flatl}$
$MM(\overline{M}(X+A)+A)$
$MM[f^\dagger,\eta_A]$
$MMMA$
μ_{MA}
MMA
μ_A
MA



(ii) For uniqueness, assume that r is a solution of e. Consider the following morphism:

$$k = \left(\overline{M}(X+A) \xrightarrow{\overline{M}(r+\mathsf{id})} \overline{M}(MA+A) \xrightarrow{\overline{M}[\mathsf{id},\eta_A]} \overline{M}MA \xrightarrow{\overline{\mu}_A} \overline{M}A \xrightarrow{\sigma} MA\right)$$

Below, we show that k is a solution of f, so $k = f^\dagger$.


$\overline{M}(X+A)$
$\overline{M}(j+\mathsf{id})$
$\overline{M}(M\overline{M}(X+A)+A+A)$
$\overline{M}(\mathsf{id}+[\mathsf{id},\mathsf{id}])$
$\overline{M}(M\overline{M}(X+A)+A)$
flatl
$\overline{M}(\overline{M}(X+A)+A)$
σ
$M(\overline{M}(X+A)+A)$
$M(\overline{M}(r+\mathsf{id})+\mathsf{id})$
$M(\overline{M}(MA+A)+A)$
$M(\overline{M}[\mathsf{id},\eta_A]+\mathsf{id})$
$M(\overline{M}MA+A)$
$M(\overline{\mu}_A+\mathsf{id})$
$M(\overline{M}A+A)$
$M[\sigma,\eta_A]$
MMA
μ_A
MA
(naturality of σ and Lemma A.6)
=
$\overline{M}(X+A)$
$\overline{M}(j+\mathsf{id})$
$\overline{M}(M\overline{M}(X+A)+A+A)$
σ
$M(M\overline{M}(X+A)+A+A)$
$M(\mathsf{id}+[\mathsf{id},\mathsf{id}])$
$M(M\overline{M}(X+A)+A)$
flatl
$M(\overline{M}(X+A)+A)$
$M(\overline{M}(r+\mathsf{id})+\mathsf{id})$
$M(\overline{M}(MA+A)+A)$
$M(\overline{M}[\mathsf{id},\eta_A]+\mathsf{id})$
$M(\overline{M}MA+A)$
$M(\overline{\mu}_A+\mathsf{id})$
$M(\overline{M}A+A)$
$M[\sigma,\eta_A]$
MMA
μ_A
MA

$$
\begin{array}{l}
\overline{M}(X+A) \\
\quad\Big\downarrow \overline{M}(j+\mathsf{id}) \\
\overline{M}(M\overline{M}(X+A)+A+A) \\
\quad\Big\downarrow \sigma \\
\overline{M}(M\overline{M}(X+A)+A+A) \\
\quad\Big\downarrow M[\mu_A \cdot M(\sigma \cdot \overline{\mu}_A \cdot \overline{M}[r,\eta_A]),\ \eta_A,\ \eta_A] \\
MMA \\
\quad\Big\downarrow \mu_A \\
MA
\end{array}
$$

$$
\begin{array}{l}
\overline{M}(X+A) \\
\quad\Big\downarrow \overline{M}(j+\mathsf{id}) \\
\overline{M}(M\overline{M}(X+A)+A+A) \\
\quad\Big\downarrow \overline{M}[\mu_A \cdot M\mu_A \cdot MM[r,\eta_A] \cdot M\sigma,\ \eta_A,\ \eta_A] \\
\overline{M}MA \\
\quad\Big\downarrow \overline{\mu}_A \\
\overline{M}A \\
\quad\Big\downarrow \sigma \\
MA
\end{array}
$$

(Lemma A.7) $=$

(naturality of σ and σ module morphism) $=$

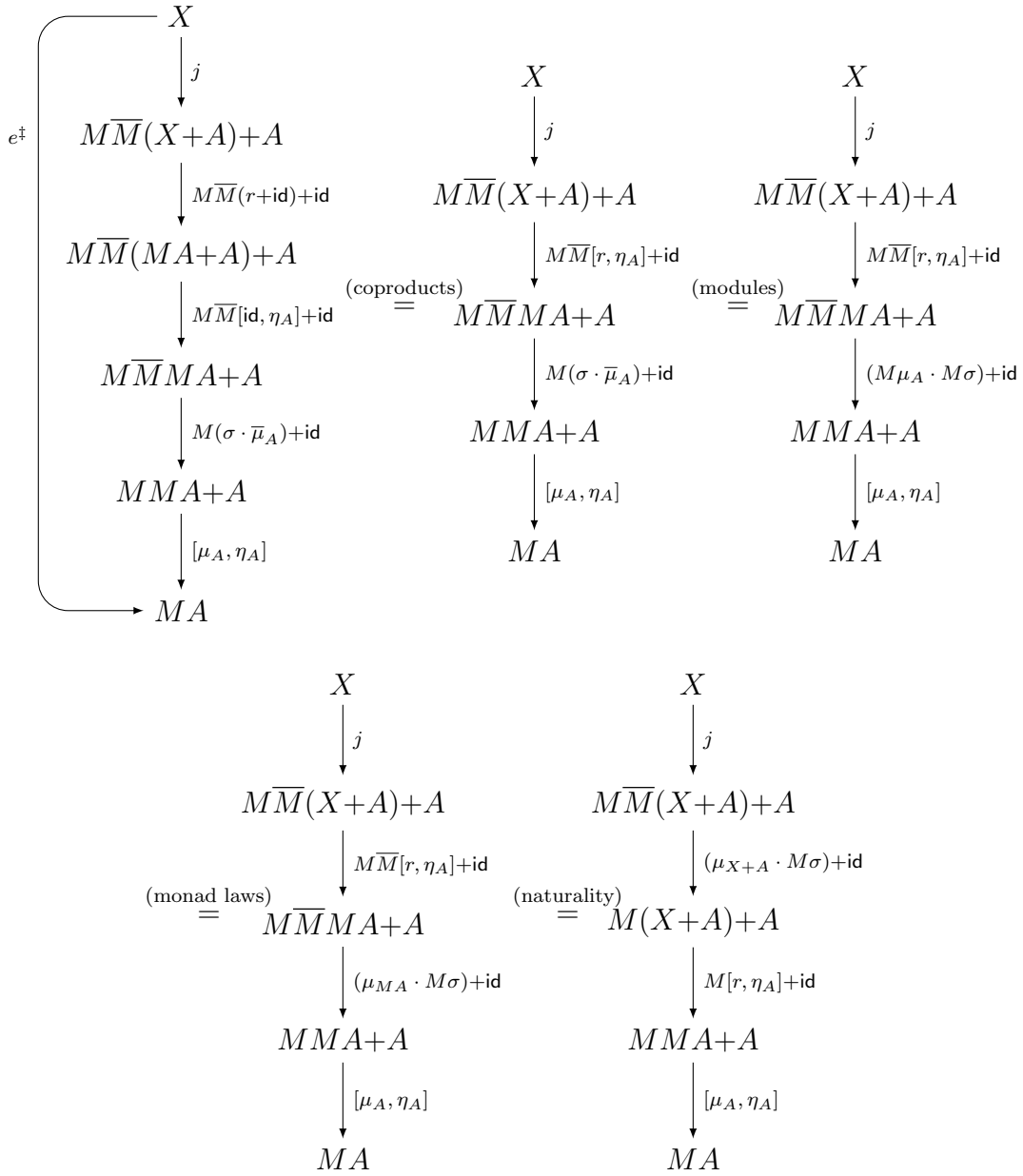
$$
\begin{array}{l}
\overline{M}(X+A) \\
\quad\Big\downarrow \overline{M}(j+\mathsf{id}) \\
\overline{M}(M\overline{M}(X+A)+A+A) \\
\quad\Big\downarrow \overline{M}[\mu_A \cdot \mu_{MA} \cdot MM[r,\eta_A] \cdot M\sigma,\ \eta_A,\ \eta_A] \\
\overline{M}MA \\
\quad\Big\downarrow \overline{\mu}_A \\
\overline{M}A \\
\quad\Big\downarrow \sigma \\
MA
\end{array}
$$

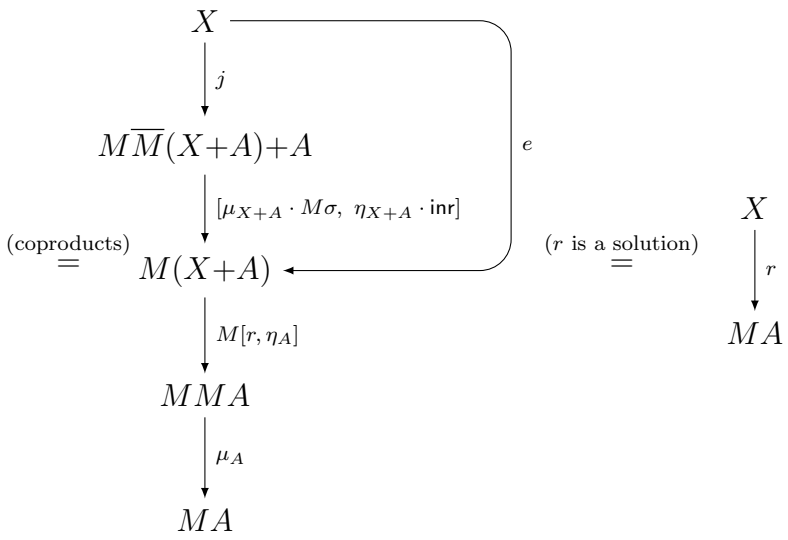
(monad laws) $=$

$$
\begin{array}{l}
\overline{M}(X{+}A) \\
\quad\downarrow{\scriptstyle \overline{M}(j{+}\mathsf{id})} \\
\overline{M}(M\overline{M}(X{+}A){+}A{+}A) \\
\quad\downarrow{\scriptstyle \begin{array}{l} \overline{M}[\mu_A \cdot M[r,\eta_A] \cdot \mu_{X+A} \cdot M\sigma,\ \mu_A \cdot M\eta_A \cdot \eta_A,\ \eta_A] \\ = \overline{M}[\mu_A \cdot M[r,\eta_A] \cdot \mu_{X+A} \cdot M\sigma,\ \mu_A \cdot M[r, M\eta_A] \cdot M\mathsf{inr} \cdot \eta_A,\ \eta_A] \\ = \overline{M}[\mu_A \cdot M[r,\eta_A] \cdot \mu_{X+A} \cdot M\sigma,\ \mu_A \cdot M[r,\eta_A] \cdot \eta_{X+A} \cdot \mathsf{inr},\ \eta_A] \end{array}} \\
\overset{\text{(monad laws)}}{=} \\
\overline{M}MA \\
\quad\downarrow{\scriptstyle \overline{\mu}_A} \\
\overline{M}A \\
\quad\downarrow{\scriptstyle \sigma} \\
MA
\end{array}
$$

$$
\begin{array}{l}
\overline{M}(X{+}A) \\
\quad\downarrow{\scriptstyle \overline{M}(j{+}\mathsf{id})} \\
\overline{M}(M\overline{M}(X{+}A){+}A{+}A) \\
\quad\downarrow{\scriptstyle \overline{M}([\mu_{X+A} \cdot M\sigma,\ \eta_{MA} \cdot \eta_A] + \mathsf{id})} \\
\overline{M}(M(X+A)+A) \\
\quad\downarrow{\scriptstyle \overline{M}(M[r,\eta_A] + \mathsf{id})} \\
\overline{M}(MMA{+}A) \\
\overset{\text{(coproducts)}}{=} \quad\downarrow{\scriptstyle \overline{M}(\mu_A + \mathsf{id})} \\
\overline{M}(MA{+}A) \\
\quad\downarrow{\scriptstyle \overline{M}[\mathsf{id},\eta_A]} \\
\overline{M}MA \\
\quad\downarrow{\scriptstyle \overline{\mu}_A} \\
\overline{M}A \\
\quad\downarrow{\scriptstyle \sigma} \\
MA
\end{array}
\qquad
\begin{array}{l}
\overline{M}(X{+}A) \\
\quad\downarrow{\scriptstyle \overline{M}(r + \mathsf{id})} \\
\overline{M}(MA{+}A) \\
\quad\downarrow{\scriptstyle \overline{M}[\mathsf{id},\eta_A]} \\
\overset{(r \text{ is a solution})}{=} \quad \overline{M}MA \\
\quad\downarrow{\scriptstyle \overline{\mu}_A} \\
\overline{M}A \\
\quad\downarrow{\scriptstyle \sigma} \\
MA
\end{array}
$$

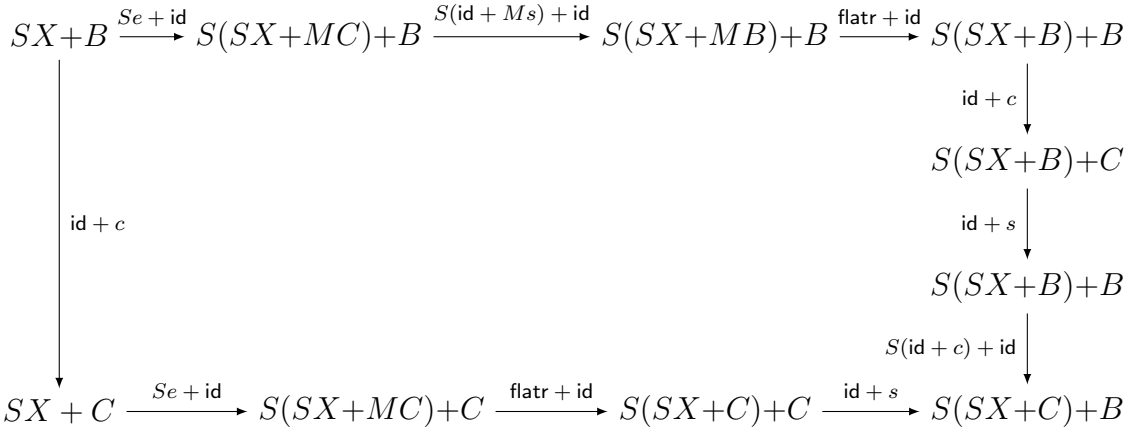
Since $f^\dagger = k$, we substitute k for $f^\dagger$ in the definition of $e^\ddagger$. We calculate:


X
j
e♯
M M̄(X+A)+A
M M̄(r+id)+id
M M̄(MA+A)+A
M M̄[id, ηA]+id
M M̄MA+A
M(σ · μ̄A)+id
MMA+A
[μA, ηA]
MA
(coproducts)
=
M M̄[r, ηA]+id
M(σ · μ̄A)+id
(modules)
(M μA · Mσ)+id
(monad laws)
(μMA · Mσ)+id
(naturality)
(μX+A · Mσ)+id
M[r, ηA]+id

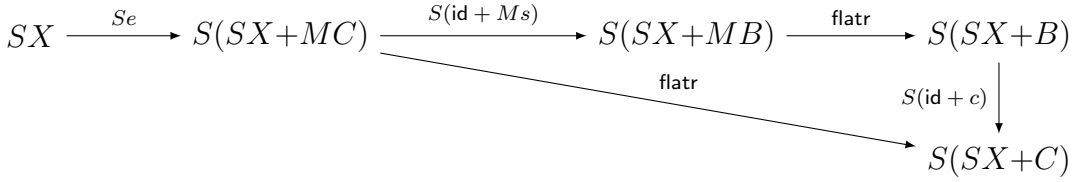


A.7 Calculation from the proof of Theorem 5.25

Unfolding the definitions of ◎ and |-|, we want to show that the following diagram commutes:



We proceed component-wisely. The right component is trivial. The left component is given by the following diagram:



(naturality of flatr and $c \cdot s = \mathsf{id}$.)

A.8 Calculation from the proof of Theorem 5.26

In this section, for brevity, we define:

$$\mathsf{red} = [T\mathsf{inl},\ \eta^T \cdot \mathsf{inr}] : TX + A \to T(X + A)$$

First, we need a couple of auxiliary lemmata:

Lemma A.8. *For all objects A and B, the following diagram commutes:*

$$\begin{array}{ccc} S(A + MB) & \xrightarrow{f * (\mathsf{id} + m)} & T(A + TB) \\ \downarrow{\scriptstyle \mathsf{flatr}} & & \downarrow{\scriptstyle \mathsf{flatr}} \\ S(A + B) & \xrightarrow{f} & T(A + B) \end{array}$$

Proof. We calculate:

$$\begin{array}{ll} & f \cdot \mathsf{flatr} \\ = & \quad (\text{ def. of } \mathsf{flatr} \ ) \\ & f \cdot \overline{\mu} \cdot S[M\mathsf{inl}, M\mathsf{inr}] \cdot S(\eta^M + \mathsf{id}) \\ = & \quad (\ \langle m, f \rangle \text{ is a module morphism }) \\ & \mu^T \cdot (f * m) \cdot S[M\mathsf{inl}, M\mathsf{inr}] \cdot S(\eta^M + \mathsf{id}) \\ = & \quad (\ \text{coproducts }) \\ & \mu^T \cdot T[T\mathsf{inl}, T\mathsf{inr}] \cdot (f * (m + m)) \cdot S(\eta^M + \mathsf{id}) \\ = & \quad (\ m \text{ is a monad morphism }) \\ & \mu^T \cdot T[T\mathsf{inl}, T\mathsf{inr}] \cdot (f * (\eta^T + m)) \\ = & \quad (\ \text{coproducts }) \\ & \mu^T \cdot T[T\mathsf{inl}, T\mathsf{inr}] \cdot T(\eta^T + \mathsf{id}) \cdot (f * (\mathsf{id} + m)) \\ = & \quad (\ \text{def. of } \mathsf{flatr} \ ) \\ & \mathsf{flatr} \cdot (f * (\mathsf{id} + m)) \end{array}$$

□

Lemma A.9. *For all morphism $h_0 : A \to C$ and $h_1 : B \to C$, the following diagram commutes:*

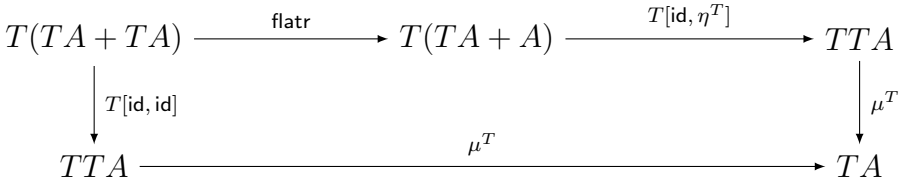
$$\begin{array}{ccc} TA + B & & \\ \downarrow{\scriptstyle \mathsf{red}} & \searrow^{[Th_0,\, \eta^T \cdot h_1]} & \\ T(A + B) & \xrightarrow{T[h_0, h_1]} & TB \end{array}$$

Proof. We calculate:

$$
\begin{aligned}
& T[h_0, h_1] \cdot \mathsf{red} \\
= \quad & \text{(def. of } \mathsf{red} \text{)} \\
& T[h_0, h_1] \cdot [T\mathsf{inl}, T\mathsf{inr}] \cdot (\mathsf{id} + \eta^T) \\
= \quad & \text{(coproducts)} \\
& [Th_0, Th_1] \cdot (\mathsf{id} + \eta^T) \\
= \quad & \text{(coproducts)} \\
& [Th_0,\ Th_1 \cdot \eta^T] \\
= \quad & \text{(naturality of } \eta^T \text{)} \\
& [Th_0,\ \eta^T \cdot h_1]
\end{aligned}
$$

□

Lemma A.10. *For all objects A, the following diagram commutes:*

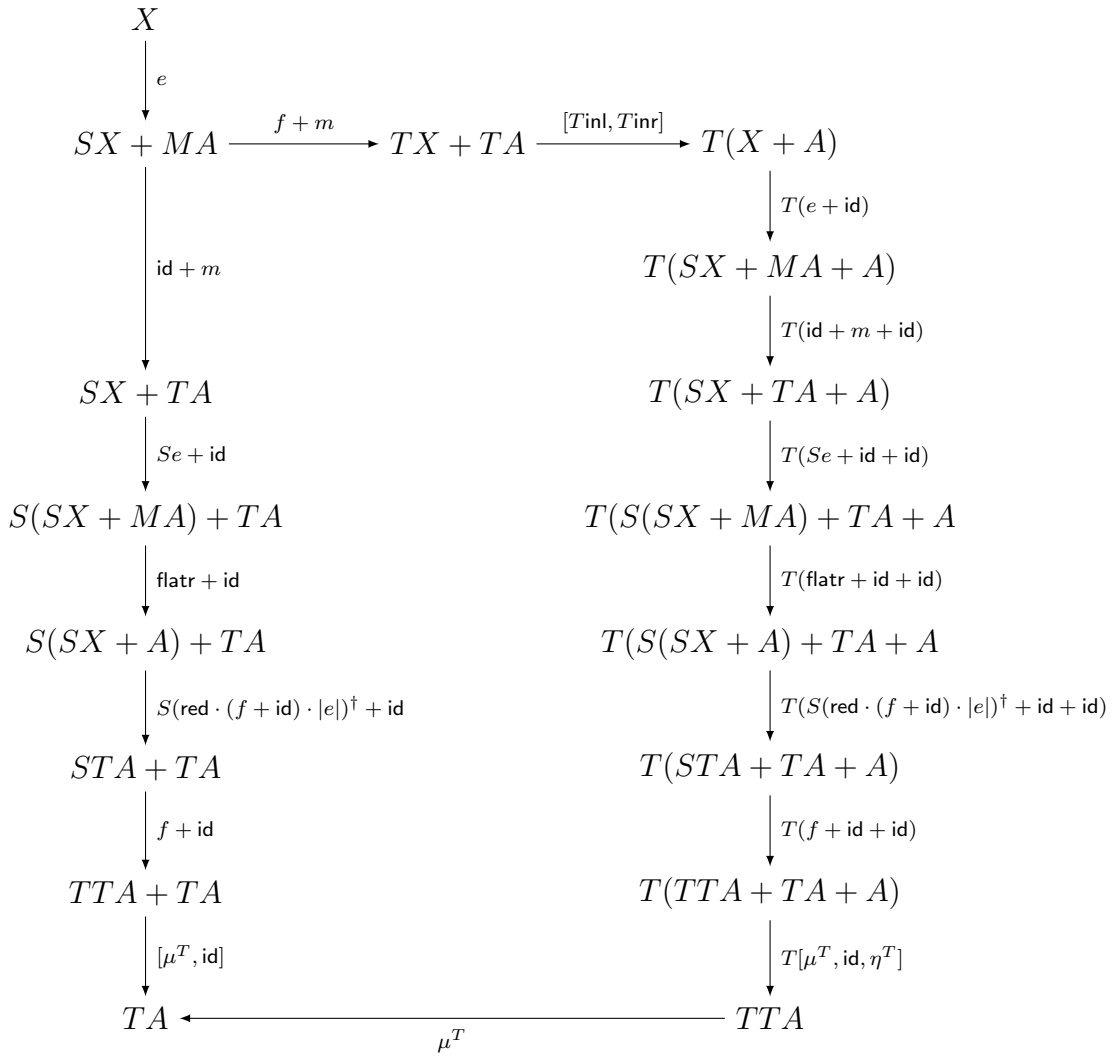


Proof. We calculate:

$$
\begin{aligned}
& \mu^T \cdot T[\mathsf{id}, \eta^T] \cdot \mathsf{flatr} \\
= \quad & \text{(def. of } \mathsf{flatr} \text{)} \\
& \mu^T \cdot T[\mathsf{id}, \eta^T] \cdot \mu^T \cdot T[T\mathsf{inl}, T\mathsf{inr}] \cdot T(\eta^T + \mathsf{id}) \\
= \quad & \text{(naturality of } \mu^T \text{)} \\
& \mu^T \cdot \mu^T \cdot TT[\mathsf{id}, \eta^T] \cdot T[T\mathsf{inl}, T\mathsf{inr}] \cdot T(\eta^T + \mathsf{id}) \\
= \quad & \text{(monad laws)} \\
& \mu^T \cdot T\mu^T \cdot TT[\mathsf{id}, \eta^T] \cdot T[T\mathsf{inl}, T\mathsf{inr}] \cdot T(\eta^T + \mathsf{id}) \\
= \quad & \text{(coproducts)} \\
& \mu^T \cdot T[\mu^T \cdot T[\mathsf{id}, \eta^T] \cdot T\mathsf{inl} \cdot \eta^T,\ \mu^T \cdot T[\mathsf{id}, \eta^T] \cdot T\mathsf{inr}] \\
= \quad & \text{(coproducts)} \\
& \mu^T \cdot T[\mu^T \cdot \eta^T,\ \mu^T \cdot T\eta^T] \\
= \quad & \text{(monad laws)} \\
& \mu^T \cdot T[\mathsf{id}, \mathsf{id}]
\end{aligned}
$$

□

We need to show that the following diagram commutes:



It is enough to show that the big square commutes. We proceed component-wisely. The left component:

$$
\begin{aligned}
& \mu^T_A \cdot f_{TA} \cdot S(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger \cdot \mathsf{flatr} \cdot Se \\
= \quad & \text{(solution)} \\
& \mu^T_A \cdot f_{TA} \cdot S\mu^T_A \cdot ST[(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger, \eta^T_A] \cdot S\mathsf{red} \cdot S(f + \mathsf{id}) \cdot S|e| \cdot \mathsf{flatr} \cdot Se \\
= \quad & \text{(def. of |-|)} \\
& \mu^T_A \cdot f_{TA} \cdot S\mu^T_A \cdot ST[(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger, \eta^T_A] \\
& \qquad \cdot S\mathsf{red} \cdot S(f + \mathsf{id}) \cdot S(\mathsf{flatr} + \mathsf{id}) \cdot S(Se + \mathsf{id}) \cdot \mathsf{flatr} \cdot Se \\
= \quad & \text{(naturality of } f \text{)} \\
& \mu^T_A \cdot T\mu^T_A \cdot TT[(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger, \eta^T_A] \cdot T\mathsf{red} \cdot T(f + \mathsf{id}) \cdot T(\mathsf{flatr} + \mathsf{id}) \cdot T(Se + \mathsf{id})
\end{aligned}
$$

$$\cdot f_{SX+A} \cdot \mathsf{flatr} \cdot Se$$

$=$ (Lemma A.8)

$$\mu^T_A \cdot T\mu^T_A \cdot TT[(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger, \eta^T_A] \cdot T\mathsf{red} \cdot T(f + \mathsf{id}) \cdot T(\mathsf{flatr} + \mathsf{id}) \cdot T(Se + \mathsf{id})$$
$$\cdot \mathsf{flatr} \cdot (f * (\mathsf{id} + m_A)) \cdot Se$$

$=$ (naturality of f)

$$\mu^T_A \cdot T\mu^T_A \cdot TT[(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger, \eta^T_A] \cdot T\mathsf{red} \cdot T(f + \mathsf{id}) \cdot T(\mathsf{flatr} + \mathsf{id}) \cdot T(Se + \mathsf{id})$$
$$\cdot \mathsf{flatr} \cdot T(\mathsf{id} + m_A) \cdot Te \cdot f_X$$

$=$ (Lemma A.9)

$$\mu^T_A \cdot T\mu^T_A \cdot T[T(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger,\ \eta^T_{TA} \cdot \eta^T_A] \cdot T(f + \mathsf{id}) \cdot T(\mathsf{flatr} + \mathsf{id}) \cdot T(Se + \mathsf{id})$$
$$\cdot \mathsf{flatr} \cdot T(\mathsf{id} + m_A) \cdot Te \cdot f_X$$

$=$ (naturality of flatr)

$$\mu^T_A \cdot T\mu^T_A \cdot T[T(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger,\ \eta^T_{TA} \cdot \eta^T_A] \cdot \mathsf{flatr} \cdot T(f + \mathsf{id}) \cdot T(\mathsf{flatr} + \mathsf{id})$$
$$\cdot T(Se + \mathsf{id}) \cdot T(\mathsf{id} + m_A) \cdot Te \cdot f_X$$

$=$ (monad laws)

$$\mu^T_A \cdot \mu^T_A \cdot T[T(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger,\ \eta^T_{TA} \cdot \eta^T_A] \cdot \mathsf{flatr} \cdot T(f + \mathsf{id}) \cdot T(\mathsf{flatr} + \mathsf{id})$$
$$\cdot T(Se + \mathsf{id}) \cdot T(\mathsf{id} + m_A) \cdot Te \cdot f_X$$

$=$ (Lemma A.8)

$$\mu^T_A \cdot \mu^T_A \cdot T[T(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger,\ \eta^T_{TA}] \cdot T(f + \mathsf{id}) \cdot T(\mathsf{flatr} + \mathsf{id}) \cdot T(Se + \mathsf{id})$$
$$\cdot T(\mathsf{id} + m_A) \cdot Te \cdot f_X$$

$=$ (monad laws)

$$\mu^T_A \cdot T\mu^T_A \cdot T[T(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger,\ \eta^T_{TA}] \cdot T(f + \mathsf{id}) \cdot T(\mathsf{flatr} + \mathsf{id}) \cdot T(Se + \mathsf{id})$$
$$\cdot T(\mathsf{id} + m_A) \cdot Te \cdot f_X$$

$=$ (coproducts)

$$\mu^T_A \cdot T[\mu^T_A \cdot T(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger,\ \mu^T_A \cdot \eta^T_{TA}] \cdot T(f + \mathsf{id}) \cdot T(\mathsf{flatr} + \mathsf{id}) \cdot T(Se + \mathsf{id})$$
$$\cdot T(\mathsf{id} + m_A) \cdot Te \cdot f_X$$

$=$ (monad laws)

$$\mu^T_A \cdot T[\mu^T_A \cdot T(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger,\ \mathsf{id}] \cdot T(f + \mathsf{id}) \cdot T(\mathsf{flatr} + \mathsf{id}) \cdot T(Se + \mathsf{id})$$
$$\cdot T(\mathsf{id} + m_A) \cdot Te \cdot f_X$$

$=$ (naturality of f)

$$\mu^T_A \cdot T[\mu^T_A \cdot f \cdot S(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger,\ \mathsf{id}] \cdot T(\mathsf{flatr} + \mathsf{id}) \cdot T(Se + \mathsf{id}) \cdot T(\mathsf{id} + m_A)$$

$$\cdot Te \cdot f_X$$

$=$ (coproducts)

$$\mu_A^T \cdot T[\mu_A^T, \mathsf{id}] \cdot T(f + \mathsf{id}) \cdot T(S(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger + \mathsf{id}) \cdot T(\mathsf{flatr} + \mathsf{id}) \cdot T(Se + \mathsf{id}) \cdot T(\mathsf{id} + m_A) \cdot Te \cdot f_X$$

$=$ (coproducts)

$$\mu_A^T \cdot T[\mu_A^T, \mathsf{id}, \eta_A^T] \cdot T(f + \mathsf{id} + \mathsf{id}) \cdot T(S(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger + \mathsf{id} + \mathsf{id}) \cdot T(\mathsf{flatr} + \mathsf{id} + \mathsf{id}) \cdot T(Se + \mathsf{id} + \mathsf{id}) \cdot T(\mathsf{id} + m_A + \mathsf{id}) \cdot T(e + \mathsf{id}) \cdot T\mathsf{inl} \cdot f_X$$

The right component:

m_A

$=$ (monad laws)

$\mu_A^T \cdot T\eta_A^T \cdot m_A$

$=$ (coproducts)

$$\mu_A^T \cdot T[\mu_A^T, \mathsf{id}, \eta_A^T] \cdot T(f + \mathsf{id} + \mathsf{id}) \cdot T(S(\mathsf{red} \cdot (f + \mathsf{id}) \cdot |e|)^\dagger + \mathsf{id} + \mathsf{id}) \cdot T(\mathsf{flatr} + \mathsf{id} + \mathsf{id}) \cdot T(Se + \mathsf{id} + \mathsf{id}) \cdot T(\mathsf{id} + m_A + \mathsf{id}) \cdot T(e + \mathsf{id}) \cdot T\mathsf{inr} \cdot m_A$$

Appendix B

Strict ideals as predicates

One way to think about an idealised monad M is that $\overline{M}$ is a subset of computations represented by M, especially when the associated natural transformation $\sigma : \overline{M} \to M$ is monic. In case of cims, it is a subset of guarded (or 'observable', 'productive') computations.

On the logical side, a subset can be viewed as a predicate (closed under composition with any other computation). From the perspective of categorical logic [66], predicates are often modelled by fibrations. In this appendix, we give some preliminary results regarding strict ideals. We show that the category of idealised monads in which the module is a strict ideal is fibred over the category of monads and monad morphisms. Then, we sketch some results regarding strict ideals of monads induced by distributivity laws, such as different kinds of resumptions introduced in this dissertation.

B.1 Fibres over monads

Definition B.1. Let $\langle M, \eta, \mu \rangle$ be a monad. For an endofunctor $\overline{M}$, a natural transformation $\sigma : \overline{M} \to M$ with monomorphic components is called a *subfunctor* of M. We call σ a *strict ideal* of M if there exists a natural transformation (a *restriction* of μ to the strict ideal) $\overline{\mu} : \overline{M}M \to \overline{M}$ such that the following diagram commutes:

$$\begin{array}{ccc} \overline{M}M & \xrightarrow{\sigma M} & M^2 \\ \downarrow{\scriptstyle \overline{\mu}} & & \downarrow{\scriptstyle \mu} \\ \overline{M} & \xrightarrow{\sigma} & M \end{array}$$

In such a situation we say that the pair $\langle M, \sigma \rangle$ is a strictly idealised monad, and call σ an *ideal injection*.

For two strictly idealised monads $\langle M, \sigma^M \rangle$ and $\langle N, \sigma^N \rangle$, a monad morphism $r : M \to N$ is *idealised* if it preserves the ideals, that is, there exists $\overline{r}$ such that $r \cdot \sigma^M = \sigma^N \cdot \overline{r}$, for a natural transformation $\overline{r} : \overline{M} \to \overline{N}$. We denote the category of idealised monads and idealised monad morphisms as $i\textsc{Mnd}$.

Note that due to the fact that σ^M is monic, the natural transformation $\overline{r}$ in the definition above is always unique.

Remark B.2. Every monad is idealised by itself, that is when the strict ideal is given by $\mathsf{id}_M : M \to M$. Moreover, in a category with the initial object 0 and monic coproduct injections, every monad is idealised by $!_M : 0 \to M$. Those are the maximal and minimal ideals respectively, in a sense that they form the respective right and left adjoints to the obvious forgetful functor $U^{\mathrm{IMnd}} : i\textsc{Mnd} \to \textsc{Mnd}$:

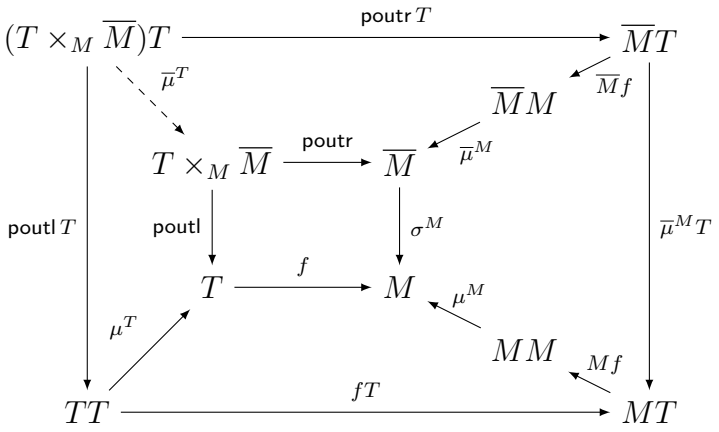
$$\begin{array}{lcll} F^{\mathrm{IMnd}} : \textsc{Mnd} \to i\textsc{Mnd} & & U^{\mathrm{IMnd}} : i\textsc{Mnd} \to \textsc{Mnd} & & \textsc{Mnd} \to i\textsc{Mnd} \\ F^{\mathrm{IMnd}} M = \langle M, \mathsf{id} \rangle & \dashv & U^{\mathrm{IMnd}} \langle M, \sigma \rangle = M & \dashv & M \mapsto \langle M, !^M \rangle \\ F^{\mathrm{IMnd}} f = f & & U^{\mathrm{IMnd}} f = f & & f \mapsto f \end{array}$$

Assume that in the base category $\mathbb{C}$ the ideals have pointwise pullbacks along monad morphisms. We can define the base change of an ideal as usually:

Theorem B.3. *Ideals are stable under pullbacks along monad morphisms. In detail, let M be a monad idealised by $\sigma^M : \overline{M} \to M$, and let $f : T \to M$ be a monad morphism. The left projection out of the pullback of these two natural transformations $\mathsf{poutl} : T \times_M \overline{M} \to T$ is an ideal of T.*

Proof. Monomorphisms are stable under pullbacks, so poutl is monic. To define the restriction of the monad multiplication, consider the following diagram, where the

perimeter is the pullback daigram of $T \times_M \overline{M} \to T$ composed with T.



(B.1)

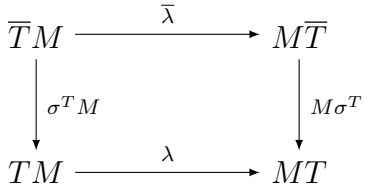
The perimeter, the pentagon on the bottom, and the hexagon on the right all commute. This means that the morphisms $f \cdot \mu^T \cdot \mathsf{poutl}\, T$ and $\sigma^M \cdot \overline{\mu}^M \cdot \overline{M}f \cdot \mathsf{poutr}\, T$ are equal. Thus, from the universal property of the inner pullback, there exists a unique $\overline{\mu}^T$ that makes the diagram commute. The trapezoid on the left is the desired restriction property. □

Corollary B.4. *One can immediately see that* $U^{\mathrm{IMnd}} : i\textsc{Mnd} \to \textsc{Mnd}$ *is a subobject-like fibration. More specifically, since there is at most one morphism between two strictly idealised monads in a fibre* U_M^{IMnd}*, it is a preorder-category fibration.*

B.2 Distributive laws and idealisations

Now, we consider monads that arise from ideal-preserving distributive laws $\lambda : TM \to MT$, like the resumption monads MS^* and MS^∞ discussed in Chapters 4 and 5 respectively. In the setting of strict ideals, the appropriate definitions simplify as follows.

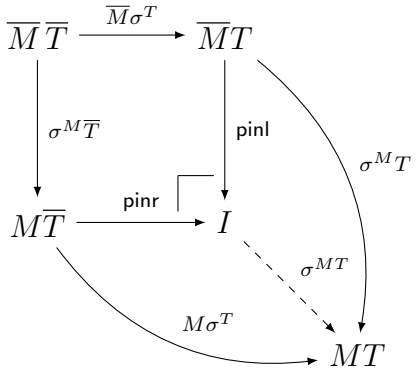
Let M and T be strictly idealised monads such that M preserves monomorphisms. Let $\lambda : TM \to MT$ be a distributive law between monads that preserves the ideal of T. In the case of strict ideals, this means that there exists a natural transformation $\overline{\lambda} : \overline{T}M \to M\overline{T}$ such that the following diagram commutes:



Moreover, since M preserves monomorphisms, $\overline{\lambda}$ is unique with such a property.

In such a setting, we know (see Example 2.60 and Theorem 4.13) that MT is idealised with $\overline{M}T + M\overline{T}$, but this module is not necessarily a strict ideal. Intuitively, a value of the type $\overline{M}\,\overline{T}$ is both in components of the coproduct, so the associated module morphism σ cannot be injective. Under some additional assumptions, we can obtain a strict ideal of MT as an appropriate quotient of $\overline{M}T + M\overline{T}$ obtained via a pushout.

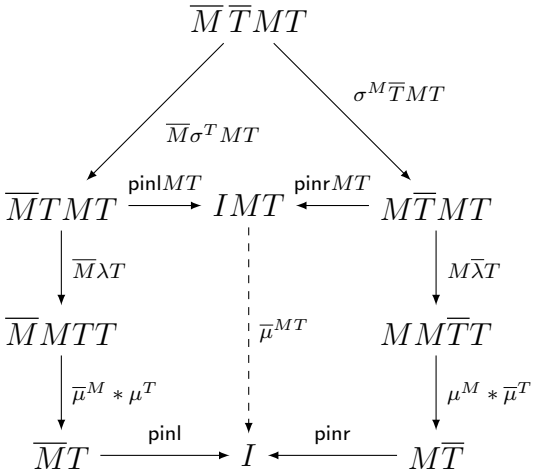
Assume that the span $M\overline{T} \xleftarrow{\sigma^M\overline{T}} \overline{M}\,\overline{T} \xrightarrow{\overline{M}\sigma^T} \overline{M}T$ has a pushout $M\overline{T} \xrightarrow{\mathsf{pinl}} I \xleftarrow{\mathsf{pinr}} \overline{M}T$. Using the universal property of the pushout, we define a natural transformation $\sigma^{MT} : I \to MT$ as the unique arrow that makes the following diagram commute (the outer edges commute from the naturality of σ^M):



(B.2)

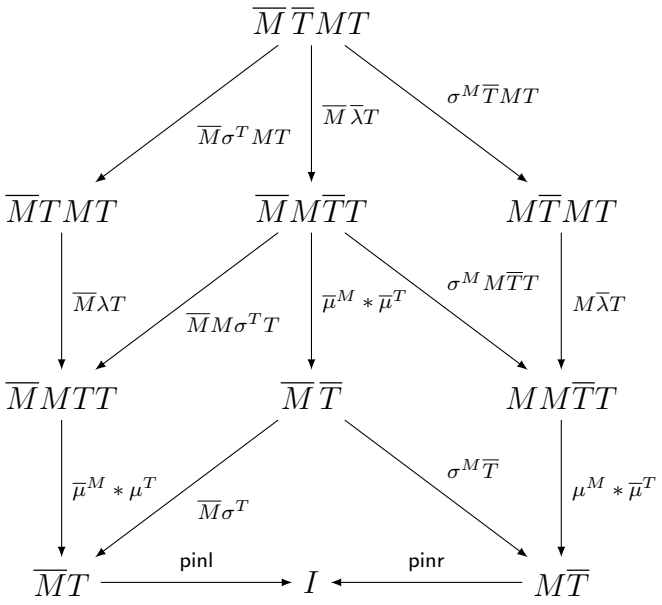
Theorem B.5. *With the definitions given above, if the natural transformation $\sigma^{MT} : I \to MT$ is monic, then it idealises the monad MT induced by the distributive law λ.*

Proof. The restriction of the multiplication of the monad MT is given by the pushout mediator:

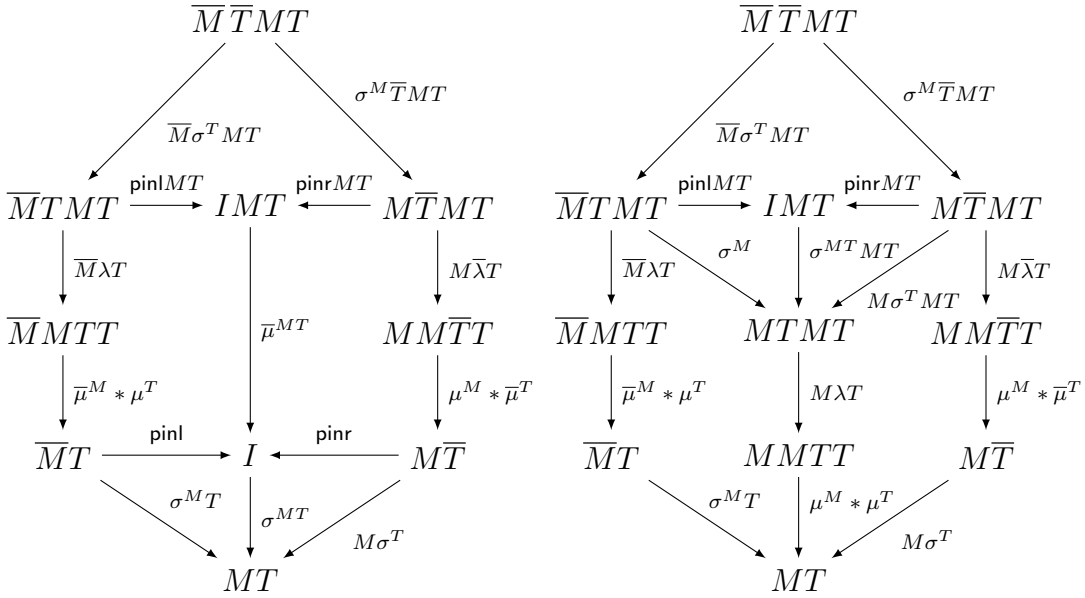


The fact that the perimeter of the diagram above commutes can be read from the

following commutative diagram:



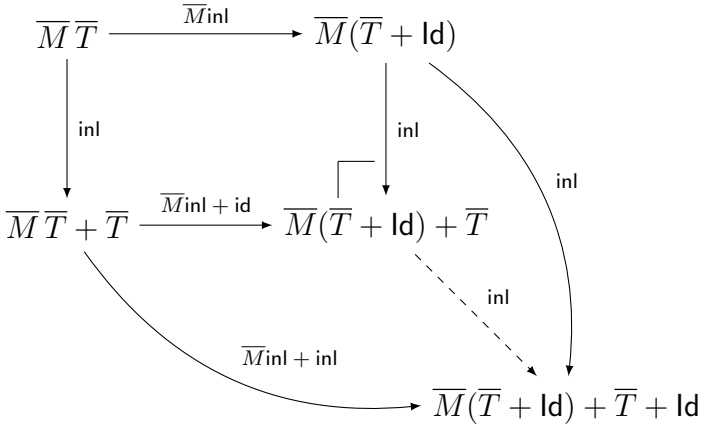
We need to check that $\overline{\mu}^{MT}$ is a restriction of μ^{MT} to the subfunctor $\overline{M}\,\overline{T}$, that is $\sigma^{MT} \cdot \overline{\mu}^{MT} = \mu^{MT} \cdot \sigma^{MT}MT = (\mu^M * \mu^T) \cdot M\lambda T \cdot \sigma^{MT}MT$. The following two diagrams commute:



This means that $\sigma^{MT} \cdot \overline{\mu}^{MT}$ and $(\mu^M * \mu^T) \cdot M\lambda T \cdot \sigma^{MT}MT$ are both mediators of the same morphisms, hence they are equal. □

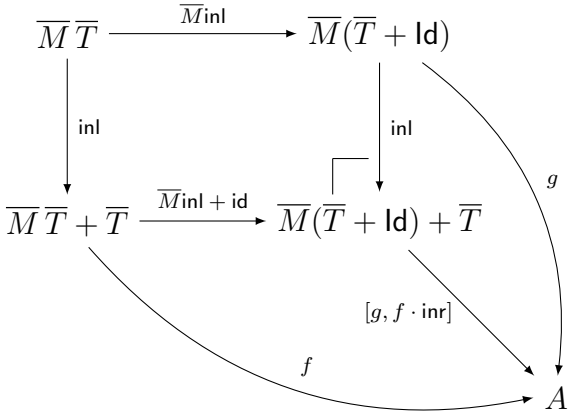
Below, we give a couple of sufficient conditions when the Theorem B.5 applies, that is, when the pushout mediator σ^{MT} is monic. The first one is when both involved monads are ideal.

Theorem B.6. *Assume that the base category $\mathbb{B}$ has monic coproduct injections. Let M and T be ideal monads. Then, the pushout diagram (B.2) specialises to:*



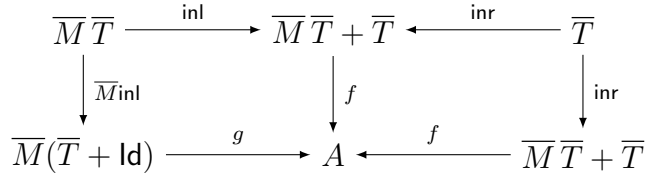
By Theorem B.5 we obtain that MT is strictly idealised with $I = \overline{M}(\overline{T}+\mathsf{Id})+\mathsf{Id}$.

Proof. We need to show that the top-left square is indeed a pushout. For two morphisms f and g, we define their pushout mediator as $[g,\, f\cdot\mathsf{inr}]$ as in the following diagram:



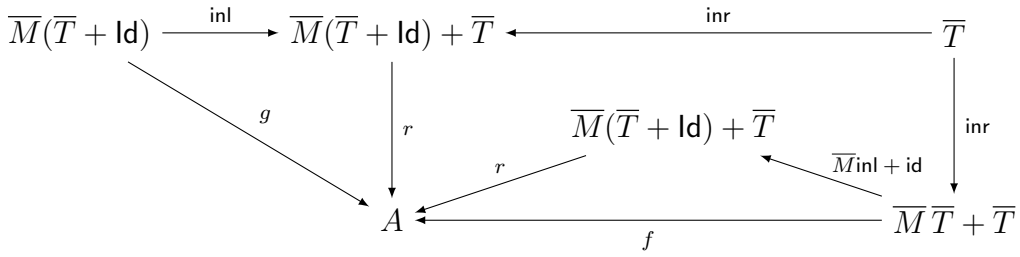
(B.3)

We need to show that this diagram commutes. The only non-trivial part is to show that $f = [g,\, f\cdot\mathsf{inr}]\cdot(\overline{M}\mathsf{inl}+\mathsf{id}) = [g\cdot\overline{M}\mathsf{inl},\, f\cdot\mathsf{inr}]$. We show that f is the appropriate coproduct mediator:



The left square is the outer edge of the diagram (B.3), the right square is trivial.

To show the uniqueness of the pushout mediator, substitute a morphism $r : \overline{M}(\overline{T}+\mathsf{Id})+\overline{T} \to A$ for $[g, f\cdot\mathsf{inr}]$ in the diagram (B.3) and assume that the diagram commutes. Then, the following diagram commutes:



This means that r is the coproduct mediator of g and $f\cdot\mathsf{inr}$. From the uniqueness of mediators, $r = [g, f\cdot\mathsf{inr}]$. □

Another situation in which MT is strictly idealised with I is when the base category has a robust calculus of subobjects, namely, it is a topos. We also need to assume that M and σ^M have some additional properties:

Lemma B.7. *Assume that the base category $\mathbb{B}$ is a topos, M preserves monomorphisms, and that the following naturality square is a pullback:*

$$\begin{array}{ccc} \overline{M}\,\overline{T} & \xrightarrow{\overline{M}\sigma^T} & \overline{MT} \\ \downarrow{\scriptstyle \sigma^M\overline{T}} & & \downarrow{\scriptstyle \sigma^M T} \\ M\overline{T} & \xrightarrow{M\sigma^T} & MT \end{array} \tag{B.4}$$

Then, I is an ideal of MT.

Proof. First, note that if M preserves monomorphisms, then so does $\overline{M}$. In a topos, the intersection of two subobjects $A \xrightarrow{a} S$ and $B \xrightarrow{b} S$ is defined as their pullback $A \xleftarrow{\mathsf{poutl}} A \times_S B \xrightarrow{\mathsf{poutr}} B$, while their union as a pushout of poutl and poutr. In our case, because the diagram (B.4) is a pullback square and pullbacks are unique up to isomorphism, the intersection of $M\overline{T} \xrightarrow{M\sigma^T} MT$ and $\overline{M}T \xrightarrow{\sigma^M T} MT$ is equal to $\overline{M}\,\overline{T}$. Their union is then given by the pushout of $M\overline{T} \xleftarrow{\sigma^M} \overline{M}\,\overline{T} \xrightarrow{\overline{M}\sigma^T} \overline{M}T$, that is I. □

Note that every topos admits an epi-mono factorisation system, which suggests that we can use the factorisation of σ^{MT} as $\sigma^{MT} = I \xrightarrow{e} J \xrightarrow{m} MT$ to further quotient I as J and take m as the strict ideal. However, it is not clear how one can define the restriction of μ^{MT} to J. A natural transformations that guarantee pullbacks like (B.4) are called *sub-cartesian*. In detail:

Definition B.8. A natural transformation $g : G \to H$ is called sub-cartesian if for every monic $f : A \to B$ the naturality square $g_B \cdot Gf = Hf \cdot g_A$ is a pullback square.

A good source of sub-cartesian natural transformations are so-called *sound* endofunctors on Set:

Definition B.9. Let C_1 be a constant endofunctor on Set, such that $C_1 X = 1$. Let C_0 be a functor such that

$$\begin{cases} C_0\emptyset = \emptyset \\ C_0 X = 1 \quad \text{where } X \neq \emptyset \end{cases}$$

An endofunctor G on Set is *sound* if every natural transformation $C_0 \to G$ has a unique extension to $C_1 \to G$.

Lemma B.10. *The following hold:*

- *Let G and H be sound endofunctors. Every natural transformation $g : G \to H$ with monic components is sub-cartesian [10].*
- *Every sound endofunctor preserves monomorphisms [114].*

Therefore, if M and $\overline{M}$ are sound, then σ^M is sub-cartesian (since it has monic components) and both $M\sigma^T$ and $\overline{M}\sigma^T$ are monic, which is enough (via Lemma B.7) to show that I is indeed an ideal.

Being sound is not a rare property to ask for. It is in accordance with the intuition that endofunctors on Set are data structures, while natural transformations manipulate only the values, leaving the structure intact. For example, all containers in the sense of Abbott, Altenkirch, and Ghani [1] are sound.

Bibliography

[1] Michael Abbott, Thorsten Altenkirch, and Neil Ghani. Categories of containers. In Andrew D. Gordon, editor, *Foundations of Software Science and Computational Structures, 6th International Conference, FOSSACS 2003 Held as Part of the Joint European Conference on Theory and Practice of Software, ETAPS 2003, Warsaw, Poland, April 7-11, 2003, Proceedings*, volume 2620 of *Lecture Notes in Computer Science*, pages 23–38. Springer, 2003.

[2] Samson Abramsky. Retracing some paths in process algebra. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR*, volume 1119 of *Lecture Notes in Computer Science (LNCS)*, pages 1–17. Springer, 1996.

[3] Samson Abramsky, Simon Gay, and Rajagopal Nagarajan. Interaction categories and the foundations of typed concurrent programming. In M. Broy, editor, *Proceedings of the 1994 Marktoberdorf Summer School on Deductive Program Design*, pages 35–113. Springer-Verlag, 1996.

[4] Samson Abramsky and Radha Jagadeesan. New foundations for the geometry of interaction. *Information and Computation*, 111(1):53–119, 1994.

[5] Peter Aczel. *Non-well-founded Sets*. Number 14 in Lecture Notes. Center for the Study of Language and Information, Stanford University, 1988.

[6] Peter Aczel, Jiří Adámek, Stefan Milius, and Jiří Velebil. Infinite trees and completely iterative theories: a coalgebraic view. *Theoretical Computer Science*, 300(1-3):1–45, 2003.

[7] Peter Aczel and Nax Paul Mendler. A final coalgebra theorem. In David H. Pitt, David E. Rydeheard, Peter Dybjer, Andrew M. Pitts, and Axel Poigné, editors, *Category Theory and Computer Science, Manchester, UK, September 5-8, 1989, Proceedings*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365. Springer, 1989.

[8] Jiří Adámek. Introduction to coalgebra. *Theory and Applications of Categories*, 14:157–199, 2005.

[9] Jiří Adámek, Stephen L. Bloom, and Stefan Milius. On algebras with iteration. *Journal of Logic and Computation*, 18(6):1047–1085, 2008.

[10] Jiří Adámek, Heinz Peter Gumm, and Vera Trnková. Presentation of Set functors: A coalgebraic perspective. *Journal of Logic and Computation*, 20(5):991–1015, 2010.

[11] Jiří Adámek, Stefan Milius, Nathan Bowler, and Paul Blain Levy. Coproducts of monads on set. In *LICS*, pages 45–54. IEEE, 2012.

[12] Jiří Adámek, Stefan Milius, and Jiří Velebil. On rational monads and free iterative theories. *Electronic Notes in Theoretical Computer Science*, 69:23–46, 2002.

[13] Jiří Adámek, Stefan Milius, and Jiří Velebil. Free iterative theories: A coalgebraic view. *Mathematical Structures in Computer Science*, 13(2):259–320, 2003.

[14] Jiří Adámek, Stefan Milius, and Jiří Velebil. Infinite trees and completely iterative theories: a coalgebraic view. *Theoretical Computer Science*, 300(1-3):1–45, 2003.

[15] Jiří Adámek, Stefan Milius, and Jiří Velebil. Elgot algebras. *Logical Methods in Computer Science*, 2(5), 2006.

[16] Jiří Adámek, Stefan Milius, and Jiří Velebil. Iterative reflections of monads. *Mathematical Structures in Computer Science*, 20(3):419–452, 2010.

[17] Danel Ahman and Tarmo Uustalu. Update monads: Cointerpreting directed containers. In Ralph Matthes and Aleksy Schubert, editors, *TYPES*, volume 26 of *LIPIcs*, pages 1–23. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[18] Benedikt Ahrens. Modules over relative monads for syntax and semantics. *Mathematical Structures in Computer Science*, FirstView:1–35, 12 2014.

[19] Pierre America and Jan J. M. M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *Journal of Computer and System Sciences*, 39(3):343–375, 1989.

[20] Davide Ancona. Soundness of object-oriented languages with coinductive big-step semantics. In James Noble, editor, *ECOOP 2012 - Object-Oriented Programming - 26th European Conference, Beijing, China, June 11-16, 2012. Proceedings*, volume 7313 of *Lecture Notes in Computer Science*, pages 459–483. Springer, 2012.

[21] André Arnold, P Naudin, and Maurice Nivat. On semantics of nondeterministic recursive program schemes. In Maurice Nivat and John C Reynolds, editors, *Algebraic Methods in Semantics*, pages 1–33. Cambridge University Press, New York, NY, USA, 1986.

[22] Robert Atkey, Neil Ghani, Bart Jacobs, and Patricia Johann. Fibrational induction meets effects. In Lars Birkedal, editor, *Foundations of Software Science and Computational Structures*, volume 7213 of *Lecture Notes in Computer Science (LNCS)*, pages 42–57. Springer, 2012.

[23] Roland Carl Backhouse, Marcel Bijsterveld, Rik van Geldrop, and Jaap van der Woude. Categorical fixed point calculus. In David H. Pitt, David E. Rydeheard, and Peter Johnstone, editors, *Category Theory and Computer Science*, volume 953 of *Lecture Notes in Computer Science (LNCS)*, pages 159–179. Springer, 1995.

[24] Michael Barr. Coequalizers and free triples. *Mathematische Zeitschrift*, 116(4):307–322, 1970.

[25] Michael Barr and Charles F. Wells. *Toposes, Triples, and Theories.* Grundlehren der mathematischen Wissenschaften. Springer-Verlag, 1985.

[26] Jonathan M. Beck. *Triples, algebras and cohomology.* PhD thesis, Columbia University, 1967.

[27] Jonathan M. Beck. Distributive laws. In *Seminar on Triples and Categorical Homology Theory*, volume 80 of *Lecture Notes in Mathematics*, pages 119–140. Springer Berlin / Heidelberg, 1969. 10.1007/BFb0083084.

[28] Pietro Cenciarelli and Eugenio Moggi. A syntactic approach to modularity in denotational semantics. In *Category Theory and Computer Science*, Amsterdam, The Netherlands, 1993.

[29] Koen Claessen. A poor man's concurrency monad. *Journal of Functional Programming*, 9(3):313–323, 1999.

[30] Koen Claessen and John Hughes. QuickCheck: a lightweight tool for random testing of Haskell programs. In Martin Odersky and Philip Wadler, editors, *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000.*, pages 268–279. ACM, 2000.

[31] Bruno Courcelle. Fundamental properties of infinite trees. *Theoretical Computer Science*, 25:95–169, 1983.

[32] Jaco W. de Bakker and Jeffery I. Zucker. Processes and the denotational semantics of concurrency. *Information and Control*, 54(1/2):70–120, 1982.

[33] Eduardo J. Dubuc. *Kan extensions in enriched category theory.* Lecture notes in mathematics. Springer, Berlin, 1970.

[34] R. Kent Dybvig and Robert Hieb. Engines from continuations. *Computer Languages*, 14(2):109–123, 1989.

[35] Samuel Eilenberg and John C. Moore. Adjoint functors and triples. *Illinois Journal of Mathematics*, 9(3):381–398, 09 1965.

[36] Calvin C. Elgot. Monadic computation and iterative algebraic theories. In *Logic Colloquium '73, Proc., Bristol 1973, 175-230*, 1975.

[37] Calvin C. Elgot, Stephen L. Bloom, and Ralph Tindell. On the algebraic structure of rooted trees. *Journal of Computer and System Sciences*, 16:362–399, 1978.

[38] Martín Hötzel Escardó. A metric model of PCF. Presented at the Workshop on Realizability Semantics and Applications (associated to the Federated Logic Conference), Trentino, Italy, 1999.

[39] Andrzej Filinski and Kristian Støvring. Inductive reasoning about effectful data types. In *International Conference on Functional Programming*, pages 97–110. ACM, 2007.

[40] Martinus Maria Fokkinga. *Law and Order in Algorithmics.* PhD thesis, University of Twente, Enschede, February 1992.

[41] Steven E. Ganz, Daniel P. Friedman, and Mitchell Wand. Trampolined style. In Didier Rémi and Peter Lee, editors, *ICFP*, pages 18–27. ACM, 1999.

[42] Neil Ghani, Christoph Lüth, and Federico De Marchi. Coalgebraic monads. *Electr. Notes Theor. Comput. Sci.*, 65(1):71–91, 2002.

[43] Neil Ghani, Christoph Lüth, Federico De Marchi, and John Power. Algebras, coalgebras, monads and comonads. *Electronic Notes in Theoretical Computer Science*, 44(1):128–145, 2001.

[44] Neil Ghani and Tarmo Uustalu. Coproducts of ideal monads. *Theoretical Informatics and Applications*, 38(4):321–342, 2004.

[45] Jeremy Gibbons. An unbounded spigot algorithm for the digits of π. *The American Mathematical Monthly*, 113(4):318–328, 2006.

[46] Jean-Yves Girard. Geometry of interaction I: Interpretation of system F. In *Logic Colloquium '88*, Studies in logic and the foundations of mathematics, pages 221–260. North-Holland Pub. Co., 1989.

[47] Roger Godement. *Topologie Algébrique et Théorie des Faisceaux*. Hermann, 1958.

[48] Joseph A. Goguen, James W. Thatcher, Eric G. Wagner, and Jesse B. Wright. Initial algebra semantics and continuous algebras. *Journal of the ACM (JACM)*, 24(1):68–95, 1977.

[49] Sergey Goncharov. Trace semantics via generic observations. In Reiko Heckel and Stefan Milius, editors, *Algebra and Coalgebra in Computer Science*, volume 8089 of *Lecture Notes in Computer Science*, pages 158–174. Springer Berlin Heidelberg, 2013.

[50] Sergey Goncharov and Lutz Schröder. A coinductive calculus for asynchronous side-effecting processes. In Olaf Owe, Martin Steffen, and Jan Arne Telle, editors, *FCT*, volume 6914 of *Lecture Notes in Computer Science (LNCS)*, pages 276–287. Springer, 2011.

[51] Tatsuya Hagino. A Typed Lambda Calculus with Categorical Type Constructors. In D. H. Pitt, A. Poigne, and D. E. Rydeheard, editors, *Category Theory and Computer Science*, pages 140–57. Springer-Verlag Lecture Notes in Computer Science 283, 1988.

[52] Peter Hancock and Anton Setzer. Interactive programs in dependent type theory. In Peter Clote and Helmut Schwichtenberg, editors, *CSL*, volume 1862 of *Lecture Notes in Computer Science (LNCS)*, pages 317–331. Springer, 2000.

[53] Peter Hancock and Anton Setzer. Interactive programs and weakly final coalgebras (extended version). In T. Altenkirch, M. Hofmann, and J. Hughes, editors, *Dependently typed programming*, number 04381 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2004. Available via http://drops.dagstuhl.de/opus/volltexte/2005/176/.

[54] William L. Harrison. The essence of multitasking. In Michael Johnson and Varmo Vene, editors, *AMAST*, volume 4019 of *Lecture Notes in Computer Science (LNCS)*, pages 158–172. Springer, 2006.

[55] Ichiro Hasuo and Bart Jacobs. Traces for coalgebraic components. *Mathematical Structures in Computer Science*, 21(2):267–320, 2011.

[56] Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3(4), 2007.

[57] Christopher T. Haynes and Daniel P. Friedman. Engines build process abstractions. In *LISP and Functional Programming*, pages 18–24, 1984.

[58] Ralf Hinze. Adjoint folds and unfolds. In Claude Bolduc, Jules Desharnais, and Béchir Ktari, editors, *Mathematics of Program Construction, 10th International Conference, MPC 2010, Québec City, Canada, June 21-23, 2010. Proceedings*, volume 6120 of *Lecture Notes in Computer Science*, pages 195–228. Springer, 2010.

[59] Ralf Hinze. Kan extensions for program optimisation or: Art and dan explain an old trick. In Jeremy Gibbons and Pablo Nogueira, editors, *Mathematics of Program Construction - 11th International Conference, MPC 2012, Madrid, Spain, June 25-27, 2012. Proceedings*, volume 7342 of *Lecture Notes in Computer Science*, pages 324–362. Springer, 2012.

[60] Ralf Hinze, Nicolas Wu, and Jeremy Gibbons. Unifying structured recursion schemes. In Morrisett and Uustalu [90], pages 209–220.

[61] André Hirschowitz and Marco Maggesi. Modules over monads and linearity. In Daniel Leivant and Ruy J. G. B. de Queiroz, editors, *WoLLIC*, volume 4576 of *Lecture Notes in Computer Science (LNCS)*, pages 218–237. Springer, 2007.

[62] André Hirschowitz and Marco Maggesi. Modules over monads and initial semantics. *Information and Computation*, 208(5):545–564, 2010.

[63] Peter J. Huber. Homotopy theory in general categories. *Mathematische Annalen*, 144:361–385, 1961.

[64] John Hughes. Why functional programming matters. *The Computer Journal*, 32(2):98–107, 1989.

[65] Martin Hyland, Gordon D. Plotkin, and John Power. Combining effects: Sum and tensor. *Theoretical Computer Science*, 357(1-3):70–99, 2006.

[66] Bart Jacobs. *Categorical Logic and Type Theory*. Number 141 in Studies in Logic and the Foundations of Mathematics. North Holland, Amsterdam, 1999.

[67] Bart Jacobs. Coalgebraic walks, in quantum and turing computation. In Martin Hofmann, editor, *Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, volume 6604 of *Lecture Notes in Computer Science*, pages 12–26. Springer, 2011.

[68] Bart Jacobs and Jan J.M.M. Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 62:222–259, 1997.

[69] Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. In Dirk Pattinson and Lutz Schröder, editors, *Coalgebraic Methods in Computer Science - 11th International Workshop, CMCS 2012, Colocated with ETAPS 2012, Tallinn, Estonia, March 31 - April 1, 2012, Revised Selected Papers*, volume 7399 of *Lecture Notes in Computer Science*, pages 109–129. Springer, 2012.

[70] Daniel M. Kan. Adjoint functors. *Transactions of the American Mathematical Society*, 87(2):pp. 294–329, 1958.

[71] Gregory M. Kelly. A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on. *Bulletin of the Australian Mathematical Society*, 22:1–83, 8 1980.

[72] Oleg Kiselyov. Iteratees. In Tom Schrijvers and Peter Thiemann, editors, *FLOPS*, volume 7294 of *Lecture Notes in Computer Science (LNCS)*, pages 166–181. Springer, 2012.

[73] Heinrich Kleisli. Every standard construction is induced by a pair of adjoint functors. *Proceedings of the American Mathematical Society*, 16(3):544–546, 06 1965.

[74] Neelakantan R. Krishnaswami and Nick Benton. Ultrametric semantics of reactive programs. In *LICS*, pages 257–266. IEEE Computer Society, 2011.

[75] Joachim Lambek. A fixpoint theorem for complete categories. *Mathematische Zeitschrift*, 103(2):151–161, 1968.

[76] Joachim Lambek and Philip J. Scott. *Introduction to Higher-Order Categorical Logic*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1988.

[77] Xavier Leroy and Hervé Grall. Coinductive big-step operational semantics. *Information and Computation*, 207(2):284–304, 2009.

[78] Saunders Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer, 1998.

[79] Ernie Manes and Philip S. Mulry. Monad compositions I: General constructions and recursive distributive laws. *Theory and Applications of Categories*, 18(7):172–208, 2007.

[80] Federico De Marchi, Neil Ghani, and Christoph Lüth. Solving algebraic equations using coalgebra. *RAIRO - Theoretical Informatics and Applications*, 37(4):301–314, 2003.

[81] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[82] Erik Meijer, Maarten M. Fokkinga, and Ross Paterson. Functional programming with bananas, lenses, envelopes and barbed wire. In John Hughes, editor, *FPCA*, volume 523 of *Lecture Notes in Computer Science*, pages 124–144. Springer, 1991.

[83] Markus Michelbrink and Anton Setzer. State dependent io-monads in type theory. *Electronic Notes in Theoretical Computer Science*, 122:127–146, 2005.

[84] Stefan Milius. Completely iterative algebras and completely iterative monads. *Information and Computation*, 196:1–41, 2005.

[85] Stefan Milius and Lawrence S. Moss. The category-theoretic solution of recursive program schemes. *Theoretical Computer Science*, 366(1-2):3–59, 2006.

[86] Robin Milner. Processes: A mathematical model of computing agents. In *Logic colloquium '73*, Studies in logic and the foundations of mathematics, pages 157–173. North-Holland Pub. Co., 1975.

[87] Robin Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[88] Eugenio Moggi. An Abstract View of Programming Languages. Technical report, Edinburgh University, 1989.

[89] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.

[90] Greg Morrisett and Tarmo Uustalu, editors. *ACM SIGPLAN International Conference on Functional Programming, ICFP'13, Boston, MA, USA - September 25 - 27, 2013*. ACM, 2013.

[91] Lawrence S. Moss. Parametric corecursion. *Theoretical Computer Science*, 260(1-2):139–163, 2001.

[92] Keiko Nakata. Resumption-based big-step and small-step interpreters for while with interactive I/O. In Olivier Danvy and Chung-chieh Shan, editors, *Proceedings IFIP Working Conference on Domain-Specific Languages, DSL 2011, Bordeaux, France, 6-8th September 2011.*, volume 66 of *EPTCS*, pages 226–235, 2011.

[93] Keiko Nakata and Tarmo Uustalu. Trace-based coinductive operational semantics for while. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *TPHOLs*, volume 5674 of *Lecture Notes in Computer Science (LNCS)*, pages 375–390. Springer, 2009.

[94] Keiko Nakata and Tarmo Uustalu. A Hoare logic for the coinductive trace-based big-step semantics of While. In Andrew D. Gordon, editor, *Programming Languages and Systems, 19th European Symposium on Programming, ESOP 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6012 of *Lecture Notes in Computer Science*, pages 488–506. Springer, 2010.

[95] Evelyn Nelson. Iterative algebras. *Theoretical Computer Science*, 25:67–94, 1983.

[96] Hanne Riis Nielson and Flemming Nielson. *Semantics with applications: a formal introduction*. John Wiley & Sons, Inc., New York, NY, USA, 1992.

[97] Nikolaos S. Papaspyrou. A resumption monad transformer and its applications in the semantics of concurrency. In *In Proceedings of the 3rd Panhellenic Logic Symposium, Anogia*, 2001.

[98] Simon Peyton Jones. Tackling the Awkward Squad: monadic input/output, concurrency, exceptions, and foreign-language calls in Haskell.

[99] Simon Peyton Jones et al. The Haskell 98 language and libraries: The revised report. *Journal of Functional Programming*, 13(1):i–xii,1–255, Jan 2003.

[100] Maciej Piróg and Jeremy Gibbons. Tracing monadic computations and representing effects. In James Chapman and Paul Blain Levy, editors, *MSFP*, volume 76 of *EPTCS*, pages 90–111, 2012.

[101] Maciej Piróg and Jeremy Gibbons. Monads for behaviour. *Electronic Notes in Theoretical Computer Science*, 298:309–324, 2013. Mathematical Foundations of Programming Semantics (MFPS XXIX).

[102] Maciej Piróg and Jeremy Gibbons. The coinductive resumption monad. *Electronic Notes in Theoretical Computer Science*, 308:273–288, 2014. Mathematical Foundations of Programming Semantics (MFPS XXX).

[103] Gordon D. Plotkin. A powerdomain construction. *SIAM Journal on Computing*, 5(3):452–487, 1976.

[104] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.

[105] Jan J. M. M. Rutten. A note on coinduction and weak bisimilarity for While programs. *Theoretical Informatics and Applications*, 33(4/5):393–400, 1999.

[106] Ondřej Rypáček. *Distributive laws in programming structures.* PhD thesis, University of Nottingham, 2010.

[107] Davide Sangiorgi. On the origins of bisimulation and coinduction. *ACM Transactions on Programming Languages and Systems*, 31(4), 2009.

[108] David A. Schmidt. Trace-based abstract interpretation of operational semantics. *Lisp and Symbolic Computation*, 10(3):237–271, 1998.

[109] Michael B. Smyth. Topology. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science: Background - Mathematical Structures (Volume 1)*, pages 641–761. Clarendon Press, Oxford, 1992.

[110] Ross Street. The formal theory of monads. *Journal of Pure and Applied Algebra*, 2(2):149–168, 1972.

[111] Wouter Swierstra. *A Functional Specification of Effects.* PhD thesis, University of Nottingham, November 2008.

[112] Wouter Swierstra and Thorsten Altenkirch. Beauty in the beast. In *Proceedings of the ACM SIGPLAN Workshop on Haskell, Haskell 2007, Freiburg, Germany, September 30, 2007*, pages 25–36, 2007.

[113] Miki Tanaka. *Pseudo-Distributive Laws and a Unified Framework for Variable Binding.* PhD thesis, University of Edinburgh, 2005.

[114] Věra Trnková. On descriptive classification of Set-functors. I. *Commentationes Mathematicae Universitatis Carolinae*, 12:143–174, 1971.

[115] Daniele Turi and Jan J. M. M. Rutten. On the foundations of final coalgebra semantics. *Mathematical Structures in Computer Science*, 8(5):481–540, 1998.

[116] Tarmo Uustalu. Generalizing substitution. *RAIRO—Theoretical Informatics and Applications*, 37:315–336, 10 2003.

[117] Tarmo Uustalu and Varmo Vene. Primitive (co)recursion and course-of-value (co)iteration, categorically. *Informatica, Lith. Acad. Sci.*, 10(1):5–26, 1999.

[118] Tarmo Uustalu, Varmo Vene, and Alberto Pardo. Recursion schemes from comonads. *Nord. J. Comput.*, 8(3):366–390, 2001.

[119] Philip Wadler. Comprehending monads. In *LISP and Functional Programming*, pages 61–78, 1990.

[120] Philip Wadler. Monads for functional programming. In Johan Jeuring and Erik Meijer, editors, *Advanced Functional Programming, First International Spring School on Advanced Functional Programming Techniques, Båstad, Sweden, May 24-30, 1995, Tutorial Text*, volume 925 of *Lecture Notes in Computer Science*, pages 24–52. Springer, 1995.

[121] Mitchell Wand. Final algebra semantics and data type extensions. *Journal of Computer and System Sciences*, 19(1):27 – 44, 1979.

Index

Notation

Particular categories

Index