Sattler, Christian (2015) On the complexities of polymorphic stream equation systems, isomorphism of finitary inductive types, and higher homotopies in univalent universes. PhD thesis, University of Nottingham.

ON THE COMPLEXITIES OF POLYMORPHIC STREAM EQUATION SYSTEMS, ISOMORPHISM OF FINITARY INDUCTIVE TYPES, AND HIGHER HOMOTOPIES IN UNIVALENT UNIVERSES

CHRISTIAN SATTLER, Dipl.–Math.

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy

December 2014

Abstract

This thesis is composed of three separate parts.

The first part deals with definability and productivity issues of equational systems defining polymorphic stream functions. The main result consists of showing such systems composed of only unary stream functions complete with respect to specifying computable unary polymorphic stream functions.

The second part deals with syntactic and semantic notions of isomorphism of finitary inductive types and associated decidability issues. We show isomorphism of so-called guarded types decidable in the set and syntactic model, verifying that the answers coincide.

The third part deals with homotopy levels of hierarchical univalent universes in homotopy type theory, showing that the n-th universe of n-types has truncation level strictly $n + 1$.

Acknowledgements

Foremost, I thank my supervisor Venanzio Capretta for his extensive support throughout my study. Without his continued determination and advice of wisdom, this thesis would not have existed in the present form.

The other members of the Functional Programming Laboratory have also provided ample opportunity for collaborations over these past years, including fruitful discussions with Florent Balestrieri, Paolo Capriotti, Ambrus Kaposi, Nuo Li, and particularly Nicolai Kraus. I thank Thorsten Altenkirch for regularly presenting interesting open research problems.

Of special significance outside of matters of research was the support always readily given by Nicolai Kraus.

Finally, I want to acknowledge my examiners, who have agreed, without knowing what lied ahead of them, to read through the version of the thesis submitted for the viva. Many of their helpful suggestions have been incorporated.

Contents

List of Figures

Chapter 0

Thesis Overview

This thesis is an outgrowth of my research during the three years of my study at the Functional Programming Laboratory of the University of Nottingham. From the beginning, the focus was not on a single project, but rather several projects of smaller size over a continuous time frame that seemed of interest at the time. This explains the decision of parting this thesis in three separate chapters, a collection of those projects I worked on that most definitely form a constituting whole one may call a thesis. Each chapter will contain a separate introduction relevant to the issues at hand. We thus will only give a brief overview here; proper references will be provided in the appropriate chapters.

0.1 Chapter One

0.1.1 Introduction

The first chapter deals with equational systems defining polymorphic stream functions. This research was originally started by Florent Balestrieri, who independently from previous work discovered a proof that productivity of such stream functions is undecidable based on a reducion to the generalized Collatz problem. Bending this construction, it turned out to be possible for any computable polymorphic stream function (with respect to an appropriate notion of computability) to be defined by such a stream equation system. This computational complexity of stream equation systems depended crucially on the availability of so-called zipping (interleaving) and projecting stream functions, concepts crucially depending on the fact that stream functions were allowed to have multiple parameters.

A natural question of interest was thus how definability and complexity of problems such as productivity change if we restrict our stream equation systems to purely unary functions. Using a fundamentally different encoding of Collatz iteration functions in terms of unary stream functions and an indirect translation of counter machines returning their result encoded in the total runtime until termination, we were able yet again to encode arbitrary recursive functions as so-called indexing functions of stream functions in unary systems, showing Π^0_2-completeness of productivity as a corollary. Yet another restriction to unary singleton systems finally yielded some positive results, showing that such stream functions must essentially have the semantics structure of Collatz iteration functions.

0.1.2 Contributions

Key contributions are as follows:

- We show that any computable unary polymorphic stream function can be defined in terms of a mutually corecursive polymorphic stream function equation system. This establishes a natural semantic notion of completeness for the above specification language and gives yet

another proof of Π^0_2-completeness (and thus undecidability) of the problem of determining productivity for such systems. This result is similar in spirit to previous work.

- We develop a novel method for encoding arbitrary computable unary polymorphic stream functions in terms of unary non-mutually corecursive polymorphic stream function equation systems. This enhances completeness and complexity theoretic results from the previous bullet point to an even more restricted setting.

- We show that unary singleton polymorphic stream function equation systems, i.e. those systems consisting of a single unary stream function definition, are limited in semantic expressibility so as to allow the semantic structure of the function to be determined in finite time.

0.1.3 Declaration of Authorship

The study of polymorphic stream function equation systems in the context of our work was originally undertaken by Florent Balestrieri, who introduced the notion of indexing function and found a proof of Π^0_2-completeness of productivity of stream function equation systems based on a reduction to the generalized Collatz problem [40], independent from the work of Endrullis et al. [16]. Subsequently, the present author noticed that Balestrieri's argument could be extended to give rise to the first bullet point above. The results of the second and third bullet points are due to the present author.

The results from the first two bullet points have previously been published as an article jointly with Florent Balestrieri at RTA 2012 [52].

0.2 Chapter Two

The second chapter deals with decidability of isomorphism of finitary inductive types. This problem was originally proposed by Thorsten Altenkirch in the set model. The similarly between recursive type equations using only finitary products and coproducts and mutual systems of polynomial equations in algebraic geometry suggested that an algebraic approach was in order. This turned out be workable for a certain subset of finitary inductive types we call guarded, a notion related to the behaviour of parametric type arguments. In that case, comparison of minimal polynomials for power series algebraic over a field of fractions of polynomials provides a finitary means of achieving the desired decidability. Using an elegant toolkit called the μ-calculus to facilitate decompositions into guarded and so-called indefinite parts, we hoped to treat each part separately. While separate decidability of isomorphism between guarded types and between indefinite types indeed turned out to be achievable, a flaw in our original argument stopped us from then combining these arguments. We are therefore forced to leave the general question open.

Another aspect is whether an isomorphism, if existing, can be realized as terms in the term model with constructors and eliminators for initial algebras. The proof that both halfs of an isomorphism form inverses should then ideally be performed in the equational theory of terms with respect to the universal properties of the involved type formers, including initial algebras. To avoid having to work with terms explicitely as much as possible, we have striven to work on the type layer. While this turned out to be feasible for the decomposition into guarded and indefinite parts, later aspects of the original set model argument unfortunately needed to be developed explicitely using term manipulation. To keep this as abstract as possible, we introduced a generalized traversal operation on regular functors that to our knowledge the literature only provided under much stronger assumptions on the ambient category. Working with internal propositional logic, equality predicates, induction-like principles, and algebraic structures, the argument for the guarded part from the set model may then be replicated internally.

0.2.1 Contribution

Stand-alone portions of this chapter deal with the following:

- We negatively answer a question by Altenkirch about second integrability of monomial quotient containers. This work overlaps with an independent corresponding result for combinatorial species.

- We establish traversability of regular functors in any bicartesian-closed category. Previous work either relied on properties of regular functors specific to set-like models or was restricted to traversals with respect to monads as opposed to applicative functors.

The key points of the main development are as follows:

- We present the notion of biased derivative, a generalization of the notion of derivative of a regular functor, and develop a methodology for applying this concept to obtain type-level syntactic decompositions.

- We define syntactic criteria for subclasses of regular functors called guarded and indefinite, with these concepts later turning out to separately enjoy more algebraic properties than the class of regular functors as a whole. With the defect of making use of only parts of the biased derivative methodology, we establish additive decomposition of regular functors into guarded and indefinite parts, with the indefinite part enjoying uniqueness.

- The set model: we show isomorphism of guarded types decidable using the language of containers [1] and the notion of algebraic power series. Isomorphism of indefinite types turns out to reduce to known problems in the theory of grammars using Parikh's theorem [46]. We discuss the problems encountered in combining these separate decidability properties via the above decomposition.

- The initial model: making use of the above work on traversability, we develop a framework for internalizing the reasoning about power series, specifically defining internal equality predicates, internal versions of structural induction over a restricted notion of internal propositions, the notion of sound and complete listing, and methods for data injection and extraction. This shows isomorphism of guarded types decidable in the syntactic model and establishes completeness of the set model with respect to this problem.

0.2.2 Declaration of Authorship

This chapter is original work by the present author, albeit stimulated by long and fruitful discussions with Nicolai Kraus. The decomposition of regular functors into guarded and indefinite parts was made precise in collaborative discussion with Nicolai Kraus, but for integration with the μ-calculus, we have restructured this aspect in a different fashion. It was Nicolai Kraus who noticed that there could not be a canonical decomposition into guarded and indefinite parts — there will always be a level of arbitrary choice for the guarded part. This observation meant we had to look for fresh arguments concerning mixed types, intermediately arriving at a certain flawed argument. Relately, work by the present author on the indefinite case turned out to be superseded by Parikh's theorem. We have thus chosen to omit its inclusion. However, it might still have its use in a future internal approach to indefinite types.

The stand-alone subsections on anti-derivatives of quotient containers and derivation of traversabilty for regular functors are original work of the present author.

Plans are underway for submission of a finalized development as a journal article jointly with Nicolai Kraus.

0.3 Chapter Three

0.3.1 Introduction

The third chapter deals with the emerging subject of homotopy type theory. This exciting research area promises to strengthen interdisciplinary links between computer science type theorists and everyday mathematical research.

A formerly open problem proposed at the Special Year on Univalent Foundations in Princeton asked: can we give a lower bound to the truncation level of univalent universes in a hierarchical order? Using an inductive argument involving higher loops in appropriately truncated higher universes, we were able to settle this question. The collaboration with Nicolai Kraus this chapter is reporting from has resulted in a journal article currently under consideration for publication in TOCL. All of our arguments are formalized in the experimental proof assistant Agda.

Originally, the plan was for this chapter to include more content, particularly about the $(\omega, 1)$-categorical interpretation of types. Based on this, using a potential theory of homotopy coherent diagrams to implement the Whitehead product from homotopy theory and utilize certain decade old results about the action of the Whitehead product on identity maps on spheres, one could explain why one of our original approaches, also presented in the chapter, failed to generalize to higher dimensions. Unfortunately, time constraints forces not yet sufficiently developed part to be omitted, although it certainly deserves further research. It is closely related to the suject of directed type theory, which amongst else is looking for a relation between the univalence axiom and parametricity.

0.3.2 Contributions

- We briefly develop a framework for naturally dealing with pointed types and the interaction between pointed type formers, in particular the loop space operator.
- We solve a left-over open problem concerning the homotopy level of higher universes in a univalent hierarchy, computing the truncation level of the n-th universe of n-types strictly as $n + 1$.

0.3.3 Declaration of Authorship

This chapter is an excerpt of joint work with Nicolai Kraus [36]. The suggestion of using the type of loops in a universe for solving the initial case is due to the present author. The critical suggestion of restricting the loops to lie in universes of truncated types is entirely due to Nicolai Kraus. This idea is what makes the proof go through in the higher cases, as presented by Nicolai Kraus at the Special Year on Univalent Foundations 2013. The accompanying Agda development, initially started by Nicolai Kraus, was completed and restructured in terms of the first bullet point by the present author. It makes use of parts of the community's Agda homotopy type theory library. [1]

[1] See `http://github.com/HoTT/HoTT-Agda/`.

Chapter 1

Polymorphic Stream Equation Systems: Productivity and Definability

1.1 Introduction

Streams over some set $\mathcal{D}$ are the basic example of a polymorphic coinductive data type, having been forced to serve as case study for almost every technique dealing with infinite data structures. Although being well-explored coalgebraic objects [10, 12, 51], they have recently re-emerged in the specific setting of term rewriting [15, 59]. They are usually introduced as the coinductive data type $\mathrm{Str}_{\mathcal{D}}$ generated by the constructor $\cdot :: \cdot : \mathcal{D} \times \mathrm{Str}_{\mathcal{D}} \to \mathrm{Str}_{\mathcal{D}}$ (cons), appending an element to the front of a stream, and come with destructors $\mathrm{head} : \mathrm{Str}_{\mathcal{D}} \to \mathcal{D}$ and $\mathrm{tail} : \mathrm{Str}_{\mathcal{D}} \to \mathrm{Str}_{\mathcal{D}}$, selecting and removing the front element, respectively. Algebraically, they can be characterized as an infinite term model parameterized by the value type $\mathcal{D}$ modulo observational equivalence on $\mathcal{D}$.

Since this work is concerned with computability and partiality, we choose to work in a semantics of partial streams, adding a bottom element $\bot$ to the underlying data type. Technically, such streams are just functions of type $\mathbb{N} \to \mathcal{D}_\bot$. [1] Note that with this terminology, if an element of a stream equals $\bot$, further elements can still be proper inhabitants of $\mathcal{D}$. Also, when speaking of computable (stream) functions, we always mean partial computable (stream) functions.

One of the simplest classes of functions on streams are the polymorphic stream functions, being parametric in the data type $\mathcal{D}$. This prohibits any kind of pattern matching or case distinction on the underlying data type, effectively restricting them to discarding, duplicating, and reordering of the input stream elements. This defines an indexing function which in the unary case has type $\mathbb{N} \to \mathbb{N}_\bot$, associating with each output stream index the corresponding input stream index to copy from, or $\bot$ if the output element is $\bot$. We can consider a polymorphic stream function f computable if this indexing function, denoted $\overline{f}$, is computable. What we call indexing function is a container morphism for streams in the terminology of Abott et al. [1].

We consider recursive stream equation systems for specifying polymorphic stream functions involving only stream constructors and destructors. As a representative example, consider the system

$$\begin{aligned} \mathrm{const}(s) &= \mathrm{head}(s) :: \mathrm{const}(s), \\ \mathrm{zip}_2(s,t) &= \mathrm{head}(s) :: \mathrm{zip}_2(t, \mathrm{tail}(s)), \\ \mathrm{hanoi}(s) &= \mathrm{zip}_2(\mathrm{hanoi}(\mathrm{tail}(s)), \mathrm{const}(s)). \end{aligned}$$

[1] We follow the usual convention that $\mathbb{N} = \{0, 1, \ldots\}$.

Through evaluation, which will be elaborated upon in the examples subsection, we find that

$$\mathrm{hanoi}(s) = \bot :: s(0) :: s(1) :: s(0) :: s(2) :: s(0) :: s(1) :: s(0) :: s(3) :: \dots.$$

The corresponding indexing function is

$$\overline{\mathrm{hanoi}}(k) = \max\{v \text{ such that } 2^v \text{ divides } k\}$$

where $\max\{\mathbb{N}\} := \bot$. To explain the naming, if $\mathcal{D}$ is instantiated with the set of disks of an infinite Tower of Hanoi and $s \in \mathrm{Str}_{\mathcal{D}}$ is a list of the disks sorted by increasing size, then $\mathrm{tail}(\mathrm{hanoi}(s))$ is a walkthrough for coinductively solving the puzzle, the k-th stream element being the disk to be moved in the k-th step, with the smallest disk always moving in the same direction [26].

The key point to stress is that polymorphism is a severe restriction. Constructing examples less trivial than the above seems out of reach: the reader is invited to try to encode the Fibonacci sequence as the indexing function of an equation in a polymorphic system. Still, we retain undecidability of productivity even in the unary setting as corollary of Theorem 1.2.

As a first taste, Proposition 1.1 states that our limited systems are nevertheless still sufficient to define every computable polymorphic stream function. Although the construction requires some imagination, the simulation of counter machines is quite direct and mainly intended to give the reader some intuition for the long road towards the proof of our key result, Theorem 1.2, which improves upon this by restricting systems to unary stream functions without mutual recursion. This is the main contribution of our work, which seems surprising giving the crippled expressiveness of the syntax.

We go on to show in Theorem 1.4 that recursive unary singleton systems are crippled enough to allow for an effective description of their semantic behaviour to be deduced in finite time.

See Simonsen [53] for a good survey of complexity analysis on stream rewriting, including the kind generalized by our developments. We hope that our Turing-complete unary recursive systems, in their simplicity, may be used as a computational model in further reduction proofs (e.g., of complexity results) not only in rewriting theory. We conclude by remarking that all our proofs are constructive, i.e. algorithmically implementable.

1.2 Syntax and Semantics

1.2.1 Streams and Indexing Functions

In our terminology, a *domain* is a directed-complete partial order, i.e. a partially ordered set such that every directed subset has a least upper bound. *Maps* or *morphisms* (or even functions) between domains D and E are given by Scott continuous functions from one to the other, i.e. functions on the underlying sets that preserve directed subsets and their least upper bounds. Recall that domains with this notion of morphisms form a complete and cocomplete category with exponentials, in particular we have the internal language of bicartesian-closed categories available to us when speaking about sum, product, and arrow types of domains. We say that such a domain D *has bottom* if it has a least element, which then will be denoted $\bot \in D$. The subcategory of domains with bottom and strict morphisms preserving the least element is no longer cartesian-closed, but the associated forgetful functor has a left adjoint mapping a domain D to a new domain $D_\bot$ with a synthetic bottom element added. Given a map $f : A \to B$ between a domain A and a domain with bottom B, this adjunction justifies the convention of implicitly extending f to $A_\bot$ by setting $f(\bot) := \bot$ when needed. Finally, we note that we often view sets as flat domain without further notice.

Given a domain D, the domain of *streams* over D is defined as $\mathrm{Str}_D := \mathbb{N} \to D$. The notion of *partial streams* over D is an abbreviation for streams over $D_\bot$. A partial stream $s : \mathrm{Str}_{D_\bot}$ is *total* if it lifts through the embedding $D \to D_\bot$, i.e. if $s(k) \neq \bot$ for every $k : \mathbb{N}$. [2] The stream

[2] A more precise choice of words might be *structure-total* for the elements of such a stream, i.e. the actual data, might come with its own notion of totality, which remains unaffected. Since we are only interested in data purely polymorphically (i.e., not at all), we choose the shorter alternative.

former Str acts as a continuous (i.e. limit preserving) functor on domains, with its action on a morphism $g : D \to E$ given by $\mathrm{Str}(g) : \mathrm{Str}_D \to \mathrm{Str}_E$ with $\mathrm{Str}(g)(s) = g \circ s$. A *polymorphic stream function* f of arity set [3] A is a natural transformation from Str^A to $\mathrm{Str}_{(\cdot)_\perp}$, i.e. a family of maps

$$f_D : \mathrm{Str}_D^A \to \mathrm{Str}_{D_\perp}$$

taking an A-indexed family of streams over D and returning a partial stream over D with the family being *parametric* in the domain D. Explicitly, the latter condition means that for all domain morphisms $g : D \to E$, we have

$$\mathrm{Str}(g_\perp) \circ f_D = f_E \circ \mathrm{Str}^A(g)$$

over the type $\mathrm{Str}_D^A \to \mathrm{Str}_{E_\perp}$. Here, we capture possible partiality of the stream function by having $E_\perp$ in the output type. We synthesize a fresh bottom rather than requiring the domain to come with a presupplied least element in order to facilitate separate notions of partiality induced by non-termination of stream functions as opposed to partial input data. We call a polymorphic stream function f *total* if it lifts through the natural embedding $\mathrm{Str} \to \mathrm{Str}_{(\cdot)_\perp}$, i.e. if it only has total streams as values.

Note that we have chosen to disallow individual domain elements (as opposed to streams of them) as arguments of stream functions in order to later arrive at a single-sorted syntactic theory (for ease of specification and handling of grammar), This is not restrictive in any effective way: finitely many domain arguments can be encoded as a prefix of an extra dummy stream argument.

Even though a polymorphic stream function is an operation on streams, its parametricity property enables us to express it as a single stream on a particular domain, pairs of indexes and positions. To see this, consider a stream function of arity A and abbreviate $I_A := A \times \mathbb{N}$. Abstractly and after uncurrying, we may view a polymorphic stream functions of arity A as a natural transformation from the functor internally represented by I_A to the functor $\mathrm{Str}_{(\cdot)_\perp}$. By an internal version of the Yoneda Lemma, it follows that such natural transformations are isomorphic to $\mathrm{Str}_{(I_A)_\perp}$.

In detail, the direction from left to right is simply given by instantiating the domain to I_A and applying the identity map. For a polymorphic stream function f of arity A, this amounts to defining $\overline{f} := f_{I_A}(j \mapsto (i, j))_{i \in A} : \mathrm{Str}_{(I_A)_\perp}$ which we call the *indexing function* of f as it tells us for any given output position which input position it is derived from.

For the direction from right to left, consider $\overline{f} : \mathrm{Str}_{(I_A)_\perp}$. Given a domain D and an argument of streams $g : I_A \to D$, we return the application of $\mathrm{Str}(g_\perp)$ to $\overline{f}$. Clearly, with $D := I_A$ and $g := \mathrm{id}_{I_A}$, we get back the the original value $\overline{f}$.

For the other order of composition, consider the following diagram illustrating an instance of naturality of f':

$$\begin{array}{ccc} I_A \to I_A & \xrightarrow{f'_{I_A}} & \mathrm{Str}_{(I_A)_\perp} \\ \downarrow{\scriptstyle s \mapsto g \circ s} & & \downarrow{\scriptstyle \mathrm{Str}(g_\perp)} \\ I_A \to D & \xrightarrow{f'_D} & \mathrm{Str}_{D_\perp} \end{array} \qquad \begin{array}{ccc} \mathrm{id}_{I_A} & \longmapsto & \overline{f} \\ \downarrow & & \downarrow \\ g & \longmapsto & f'_D(g) \cdots\cdots \mathrm{Str}(g_\perp)(g) \end{array}$$

Starting with an arbitrary polymorphic stream function f of arity A and its uncurry-adjusted version f', we may compute $\overline{f}$ as shown in the top row. Given a domain D and an argument of streams $g : I_A \to D$, naturality of f' ensures that applying $\mathrm{Str}(g_\perp)$ to $\overline{f}$ reproduces the original value of f' at argument g.

Under the above representation of polymorphic stream functions as indexing functions, it is clear that a polymorphic stream function is total if and only if its corresponding indexing

[3] Here, A is usually chosen to be $[n] := \{0, \ldots, n-1\}$ for some $n \in \mathbb{N}$, in which case we identify $X^{[n]}$ and $X^n = X \times \ldots \times X$ in the obvious way. However, the treatment is a bit less redundant if generalized to arbitrary A.

function is total as a partial stream. This correspondence also allows us to transfer in a rather canonical manner the notion of computability to polymorphic stream functions from their indexing functions. [4]

In the remainder of the chapter, we will identify $[1] \times \mathbb{N}$ with $\mathbb{N}$, both to increase readability of the presentation and later on because of the specific focus on unary stream functions.

It is worth noting that indexing functions represent an inversion of the usual notions of input and output for stream functions. This contravariance is reflected in the fact that

$$\overline{f \circ g} = \overline{g} \circ \overline{f}$$

for polymorphic $f_D, g_D : \mathrm{Str}_D \to \mathrm{Str}_D$, a fact heavily utilized in the rest of this chapter.

Basic examples of polymorphic stream functions are the tail operation

$$\begin{aligned} &\mathrm{tail}_D : \mathrm{Str}_D \to \mathrm{Str}_D, \\ &\mathrm{tail}_D(s) = i \mapsto s(i+1) \end{aligned}$$

with indexing function $\overline{\mathrm{tail}}(k) = k + 1$ and the combined head-cons operation

$$\begin{aligned} &(\mathrm{head}(\cdot) :: \cdot)_D : \mathrm{Str}_D \times \mathrm{Str}_D \to \mathrm{Str}_D, \\ &\mathrm{head}(s) :: t = i \mapsto \begin{cases} s(0) & \text{if } i = 0, \\ t(i-1) & \text{else} \end{cases} \end{aligned}$$

with indexing function

$$\overline{(\mathrm{head}(\cdot) :: \cdot)}(k) = \begin{cases} (0,0) & \text{if } k = 0, \\ (1, k-1) & \text{else.} \end{cases}$$

The reason for stating head and cons as a combined operation — in contrast to the introduction — is our desire to avoid an extra sort for individual domain elements in our syntactic treatment (see below). We are justified in merging these because head and cons are the only primitive stream operation to produce and consume individual domain elements, respectively. Note though that it is possible in principle to extend the framework of indexing functions to polymorphic stream functions that additionally also take a finite number of domain elements as arguments by slightly changing the definition of I_A to an arbitrary coproduct of copies of $\mathbb{N}$ and $[1]$.

1.2.2 Stream Equation Systems

Let us specify a simple scheme for stating recursive systems of syntactic equations giving rise to polymorphic stream functions. Fix an environment consisting of an equation index set E and, for each $e : E$, a set $\mathbf{A}_e$ designating the intended arity of the stream function defined by equation e. *Stream terms* $\textsc{StrTerm}_A$ of arity A are inductive generated as follows:

$$\frac{a \in A}{\textsc{param}(a) \in \textsc{StrTerm}_A} \qquad \frac{e \in E \qquad \sigma_i \in \textsc{StrTerm}_A \text{ for } i \in \mathbf{A}_e}{\textsc{call}(e, \vec{\sigma}) \in \textsc{StrTerm}_A}$$

$$\frac{\sigma \in \textsc{StrTerm}_A}{\textsc{tail}(\sigma) \in \textsc{StrTerm}_A} \qquad \frac{\sigma, \sigma' \in \textsc{StrTerm}_A}{\textsc{headcons}(\sigma, \sigma') \in \textsc{StrTerm}_A}$$

A *(stream equation) system* then consists of, for each $e : E$, a stream term of arity $\mathbf{A}_e$.

[4] Computability of elements of $\mathrm{Str}_{(I_A)_\perp}$ boils down to the standard notion of computability on $\mathbb{N} \to \mathbb{N}_\perp$ via some computable bijection $A \times \mathbb{N} \simeq \mathbb{N}$.

In practice, we will rarely be that formal and write down such systems as simple lists of equations of the form $f_e(\vec{s}) = \sigma$ where σ might be generated by the following grammar:

$$\begin{array}{rll} \sigma ::= & s_i & \text{stream parameter with } i \in A, \\ \mid & \mathrm{tail}(\sigma) & \text{stream stripped of its first element,} \\ \mid & \mathrm{head}(\sigma) :: \sigma' & \text{first element of a stream prepended to stream (head-cons),} \\ \mid & f_e(\vec{\sigma}) & \text{recursive call with } e \in E \text{ and arguments } \sigma_i \text{ for } i \in \mathbf{A}_e. \end{array}$$

The actual names chosen for individual syntactic stream functions or parameters is, of course, irrelevant. In each specific context, we will choose appropriate names so as to maximize readability. We further explicitly state that there are no further restrictions such as guardedness on the form of the stream terms σ since we specifically deal with ill-defined equations using our (domain theoretic) partiality semantics. A system is called *unary* if all of its defined stream functions are unary, i.e. $|\mathbf{A}_e| = 1$ for all $e \in E$.

By a standard application of the Kleene fixed-point theorem [21], a stream equation system gives, for each fixed domain D, rise to stream functions of corresponding number and arities, the least fixpoint of the given system of equations when the tail and head-cons operations are interpreted according to the previous section. It is clear that the resulting stream functions are indeed polymorphic: the syntactic specification did not make any mention of properties of the underlying domain, so the fixpoint construction will remain parametric. Because we will consider different variants of operational semantics derived from this, we will go through this process in detail. An equation in the system is called *productive* if the polymorphic stream function it defines is total. In what follows, we will usually use the same symbols to denote syntactic occurrences and their semantic counterparts: their meaning will always be clear from the context.

We can also view such a system as an *executable specification*, giving rise to a notion of operational semantics allowing us to explicitly construct the least fixpoint alluded to above. For this, we formalize the computation of stream elements using a functional relation $\cdot \to \cdot$ on pairs $\sigma \,!\, k$ of stream terms σ and indices $k \in \mathbb{N}$ by setting

$$\begin{aligned} \mathrm{tail}(\sigma) \,!\, k &\to \sigma \,!\, k+1, \\ \mathrm{head}(\sigma) :: \sigma' \,!\, k &\to \begin{cases} \sigma \,!\, 0 & \text{if } k = 0, \\ \sigma' \,!\, k-1 & \text{else,} \end{cases} \\ f_e((\sigma_i)_{i \in \mathbf{A}_e}) \,!\, k &\to \rho[\sigma_i / s_i]_{i \in \mathbf{A}_e} \,!\, k \end{aligned}$$

where $f_e(\vec{s}) = \rho$ is an equation in the system. This is effectively equivalent to introducing a rewriting system on constructs of the form $\mathrm{head}(\mathrm{tail}^k(\sigma))$ based on the rules

$$\begin{aligned} \mathrm{head}(\mathrm{head}(\sigma) :: \sigma') &\to \mathrm{head}(\sigma), \\ \mathrm{tail}(\mathrm{head}(\sigma) :: \sigma') &\to \sigma' \end{aligned}$$

and call-inlining with a deterministic outermost rewriting strategy. For a given domain D, we can now compute $f_{e,D}$ via

$$f_{e,D}(\vec{t})(k) = \begin{cases} t_w(i) & \text{if } f_e(\vec{s}) \,!\, k \to^* s_w \,!\, i, \\ \bot & \text{else} \end{cases}$$

for $e \in E$, verifying that this is indeed the smallest solution to the given specification and fulfills the parametricity property. From this, we can express the corresponding indexing function as

$$\overline{f_e}(k) = \begin{cases} (w, r) & \text{if } f_e(\vec{s}) \,!\, k \to^* s_i \,!\, r, \\ \bot & \text{else.} \end{cases}$$

Retrospectively, this consolidates our definition of productivity with its usual connotation, namely that each finite prefix of a stream (or the result of a stream function called with productive arguments) be constructible through finite evaluation. On a side note, this description of

indexing functions makes explicit the obvious fact that the defined stream functions are always computable.

1.2.3 Examples

Using this syntax, we can specify the *interleaving* function zip_n for $n > 0$ as

$$\mathrm{zip}_n(s_0, \ldots, s_{n-1}) = \mathrm{head}(s_0) :: \mathrm{zip}_n(s_1, \ldots, s_{n-1}, \mathrm{tail}(s_0)).$$

It is productive with indexing function $\overline{\mathrm{zip}_n}(k) = (k \bmod n, \lfloor k/n \rfloor)$. Let us prove this statement in detail by induction. In the base case, we have

$$\begin{aligned}\mathrm{zip}_n(s_0, \ldots, s_{n-1}) \,!\, 0 &\to \mathrm{head}(s_0) :: \mathrm{zip}_n(s_1, \ldots, s_{n-1}, \mathrm{tail}(s_0)) \,!\, 0 \\ &\to s_0 \,!\, 0\end{aligned}$$

proving $\overline{\mathrm{zip}_n}(0) = (0, 0) = (0 \bmod n, \lfloor 0/n \rfloor)$. In the induction step, we have

$$\begin{aligned}\mathrm{zip}_n(s_0, \ldots, s_{n-1}) \,!\, k+1 &\to \ \mathrm{head}(s_0) :: \mathrm{zip}_n(s_1, \ldots, s_{n-1}, \mathrm{tail}(s_0)) \,!\, k+1 \\ &\to \ \mathrm{zip}_n(s_1, \ldots, s_{n-1}, \mathrm{tail}(s_0)) \,!\, k \\ &\to^* \begin{cases} \mathrm{tail}(s_0) \,!\, \lfloor k/n \rfloor & \text{if } k \equiv -1 \mod n, \\ s_{(k \bmod n)+1} \,!\, \lfloor k/n \rfloor & \text{else} \end{cases} \\ &\to^* s_{(k+1) \bmod n} \,!\, \lfloor (k+1)/n \rfloor,\end{aligned}$$

proving $\overline{\mathrm{zip}_n}(k) = (k \bmod n, \lfloor k/n \rfloor)$ implies $\overline{\mathrm{zip}_n}(k+1) = ((k+1) \bmod n, \lfloor (k+1)/n \rfloor)$.

As a kind of inverse to interleaving, the *projection* function proj_n is defined as

$$\mathrm{proj}_n(s) = \mathrm{head}(s) :: \mathrm{proj}_n(\mathrm{tail}^n(s)).$$

It is easily shown to be productive with indexing function $\overline{\mathrm{proj}_n}(k) = nk$. For convenience, we also define shifted projections $\mathrm{proj}_{n,i}(s) := \mathrm{proj}_n(\mathrm{tail}^i(s))$ for $i < n$ with $\overline{\mathrm{proj}_{n,i}}(k) = nk + i$. Note that, for all domains D, we have

$$\mathrm{proj}_{n,i,D}(\mathrm{zip}_{n,D}(s_0, \ldots, s_{n-1})) = s_i$$

for $s_0, \ldots, s_{n-1} \in \mathrm{Str}_D$ as well as

$$s = \mathrm{zip}_{n,D}\left((\mathrm{proj}_{n,i,D}(s))_{i \in [n]}\right)$$

for $s : \mathrm{Str}_D$.

We also have the *constant function*

$$\mathrm{const}(s) = \mathrm{head}(s) :: \mathrm{const}(s),$$

repeating the first stream element of its argument, with $\overline{\mathrm{const}}(k) = 0$.

We are now ready to return to the example from the introduction. Recall that

$$\mathrm{hanoi}(s) = \mathrm{zip}_2(\mathrm{hanoi}(\mathrm{tail}(s)), \mathrm{const}(s)).$$

We will prove that

$$\overline{\mathrm{hanoi}}(2^v(2m+1)) = v$$

by induction on v. In the base case, we have

$$\begin{aligned}\mathrm{hanoi}(s) \,!\, 2k+1 &\to \ \mathrm{zip}_2(\mathrm{hanoi}(\mathrm{tail}(s)), \mathrm{const}(s)) \,!\, 2k+1 \\ &\to^* \mathrm{const}(s) \,!\, k \\ &\to^* s \,!\, 0,\end{aligned}$$

proving $\overline{\text{hanoi}}(2k+1) = 0$. In the induction step, we have

$$\begin{aligned}\text{hanoi}(s) \,!\, 2^{v+1}(2m+1) &\to \text{zip}_2(\text{hanoi}(\text{tail}(s)), \text{const}(s)) \,!\, 2^{v+1}(2m+1)\\ &\to^* \text{hanoi}(\text{tail}(s)) \,!\, 2^v(2m+1)\\ &\to^* \text{tail}(s) \,!\, v\\ &\to s \,!\, v+1,\end{aligned}$$

proving $\overline{\text{hanoi}}(2^v(2m+1)) = v$ implies $\overline{\text{hanoi}}(2^{v+1}(2m+1)) = v+1$. Now, the interesting artifact is the first stream position,

$$\begin{aligned}\text{hanoi}(s) \,!\, 0 &\to \text{zip}_2(\text{hanoi}(\text{tail}(s)), \text{const}(s)) \,!\, 0\\ &\to^+ \text{hanoi}(\text{tail}(s)) \,!\, 0,\end{aligned}$$

leading to an infinite loop

$$\text{hanoi}(s) \,!\, 0 \to^+ \text{hanoi}(\text{tail}(s)) \,!\, 0 \to^+ \text{hanoi}(\text{tail}^2(s)) \,!\, 0 \to^+ \ldots,$$

showing that $\overline{\text{hanoi}}(0) = \bot$.

After having established some intuition for computing indexing functions, let us prove a lemma which will be of much use further on.

Lemma 1.1. *Given $h \geq 1$ and a stream equation of the form*

$$f(s) = \text{head}(\text{tail}^{a_0}(s)) :: \ldots :: \text{head}(\text{tail}^{a_{h-1}}(s)) :: f(v(s))$$

with an arbitrary unary stream function v, then $\overline{f}(k) = \overline{v}^{\lfloor k/h \rfloor}(a_{k \bmod h})$ for $k \in \mathbb{N}$.

Proof. Since this is our first technical result about indexing functions, we will be explicit in every detail. The proof is by induction on $k \in \mathbb{N}$. For $k < h$, we have

$$\begin{aligned}f(s) \,!\, k &\to \text{head}(\text{tail}^{a_0}(s)) :: \ldots :: \text{head}(\text{tail}^{a_{h-1}}(s)) :: f(v(s)) \,!\, k\\ &\to^k \text{tail}^{a_k}(s) \,!\, 0\\ &\to^{a_k} s \,!\, a_k,\end{aligned}$$

yielding

$$\overline{f}(k) = a_k = \overline{v}^0(a_k) = \overline{v}^{\lfloor k/h \rfloor}(a_{k \bmod h}).$$

For $k \geq h$, we have

$$\begin{aligned}f(s) \,!\, k &\to \text{head}(\text{tail}^{a_0}(s)) :: \ldots :: \text{head}(\text{tail}^{a_{h-1}}(s)) :: f(v(s)) \,!\, k\\ &\to^h f(v(s)) \,!\, k-h.\end{aligned}$$

By induction hypothesis, it follows that

$$\begin{aligned}\overline{f}(k) &= (\overline{f \circ v})(k-h)\\ &= \overline{v}(\overline{f}(k-h))\\ &= \overline{v}(\overline{v}^{\lfloor (k-h)/h \rfloor}(a_{(k-h) \bmod h}))\\ &= \overline{v}(\overline{v}^{\lfloor k/h \rfloor - 1}(a_{k \bmod h}))\\ &= \overline{v}^{\lfloor k/h \rfloor}(a_{k \bmod h}),\end{aligned}$$

where we exploited contravariance of the indexing operation in the second step. □

1.3 Definability

Our first result consists of the insight that the above stream equations of simple form, incorporating only the stream constructor and destructors and recursion, already allow for the definition of every computable polymorphic stream function. In the following, we will only consider single argument functions since we can easily fuse multiple arguments into a single one using instances of zip and proj. The argument can be seen as a generalization of the idea expressed in the proof of Theorem 4.4 in Endrullis et al. [17] that recognizing productivity of a form of system similar to ours (with a unit stream data type instead of abstract polymorphism) is of complexity Π^0_2. Since our key contribution concerns the even more general result of unary definability, the main purpose of the following exposition is to serve as a contrast to the next section.

Recall that for a single argument stream function, the indexing function has type $\mathbb{N} \to \mathbb{N}_\perp$. We will give our proof in the form of a reduction, transforming a counter machine [44] representing an arbitrary computable function $\phi : \mathbb{N} \to \mathbb{N}_\perp$ into a corresponding system with an equation having ϕ as indexing function.

Definition 1.1. *A* counter machine *is given by a tuple (L, I) where L is the length of the program and I is a list $I_0, \ldots, I_{L-1}$ of instructions* $\mathrm{inc}(r)$ *with $r \in \mathbb{N}$, denoting increment of register r, and* $\mathrm{jzdec}(r, l)$ *with $r \in \mathbb{N}$ and $l \in [L+1]$, denoting a jump to instruction l if register r is zero and a decrement of r otherwise.*

The semantics of such a machine is as follows: the *state* $(P, R) \in S := [L+1] \times \mathbb{N}^{(\mathbb{N})}$ consists of the value P of its instruction pointer and the values R of its registers, where $\mathbb{N}^{(\mathbb{N})}$ denotes the set of functions $\mathbb{N} \to \mathbb{N}$ with finite support, i.e. with only finitely many values non-zero. Such a tuple is called *terminal* if $P = L$, denoting the machine has exited with output $R(1)$. For $R \in \mathbb{N}^{(\mathbb{N})}$ and $r, v \in \mathbb{N}$, the result of replacing the r-th entry of R with v will be denoted $R[r \leftarrow v]$. Representing execution, we define a functional *next* relation $\to$ over S on non-terminal elements by

$$(P, R) \to \begin{cases} (P+1, R[r \leftarrow R(r)+1]) & \text{if } I_P = \mathrm{inc}(r), \\ (P+1, R[r \leftarrow R(r)-1]) & \text{if } I_P = \mathrm{jzdec}(r, l), R(r) \neq 0, \\ (l, R) & \text{if } I_P = \mathrm{jzdec}(r, l), R(r) = 0. \end{cases}$$

The *result function* $\mathrm{result}_{(L,I)} : S \to \mathbb{N}_\perp$ is defined as

$$\mathrm{result}_{(L,I)}(t) = \begin{cases} R(1) & \text{for } t \to^* (L, R) \text{ terminal}, \\ \perp & \text{else.} \end{cases}$$

The *initial state* for a given input $i \in \mathbb{N}$ is given by

$$\mathrm{init}(i) := (0, (i, 0, \ldots)).$$

With this machinery, we can now define the associated computable function of the counter machine as

$$\phi_{(L,I)} = \mathrm{result}_{(L,I)} \circ \mathrm{init} : \mathbb{N} \to \mathbb{N}_\perp.$$

Proposition 1.1. *Given a counter machine (L, I), there is a stream equation system defining a unary stream function with indexing function $\phi_{(L,I)}$.*

Definition 1.2. *Let $p_0 < p_1 < \ldots$ be an ordered listing of all the primes. We encode a register state $R \in \mathbb{N}^{(\mathbb{N})}$ as a single non-zero natural number using $\widehat{R} := \prod_{r \in \mathbb{N}, R(r) \neq 0} p_r^{R(r)}$.*

Proof of Proposition 1.1. Our goal is to mutually define unary stream functions $f_0, \ldots, f_{L-1}$ such that for a sequence of machine states $(P, R) \to (P', R')$, we have $f_P(s) \,!\, \widehat{R} \to^+ f_{P'}(s) \,!\, \widehat{R'}$, effectively simulating the execution of the counter machine. It follows that whenever the counter

machine terminates with $(P, R) \to^* (L, R')$ terminal, then $\overline{f_P}(\widehat{R}) = \overline{f_L}(\widehat{R'})$, and $\overline{f_P}(\widehat{R}) = \bot$ otherwise. Defining f_L to extract the value of register 1, i.e. the exponent of $p_1 = 3$, from the encoding we set

$$f_L(s) = \mathrm{zip}_3(f_L(\mathrm{tail}(s)), \mathrm{const}(s), \mathrm{const}(s)).$$

A straightforward induction on $R(1)$ shows that

$$\overline{f_L}(\widehat{R}) = R(1) = \mathrm{result}_{(L,I)}(L, R) \neq \bot$$

for $R \in \mathbb{N}^{(\mathbb{N})}$. Together, this means

$$f_P(\widehat{R}) = \mathrm{result}_{(L,I)}(P, R)$$

for any state $(P, R) \in S$.

For each instruction I_P with $P \in \{0, \ldots, L-1\}$, we mutually define a corresponding stream function f_P reproducing the action of I_P on the register encoding. If $I_P = \mathrm{inc}(r)$, let

$$f_P(s) = \mathrm{proj}_{p_r}(f_{P+1}(s))$$

and note that

$$\begin{aligned} f_P(s) \,!\, \widehat{R} &\to^+ f_{P+1}(s) \,!\, p_r\widehat{R} \\ &= f_{P+1}(s) \,!\, R[r \leftarrow \widehat{R(r)} + 1] \end{aligned}$$

for $R \in \mathbb{N}^{(\mathbb{N})}$, simulating an increment. If $I_P = \mathrm{jzdec}(r, l)$, let

$$f_P(s) = \mathrm{zip}_{p_r}\left(\begin{cases} f_{P+1}(s) & \text{if i} = 0, \\ \mathrm{proj}_{p_r,i}(f_l(s)) & \text{else} \end{cases}\right)_{i \in [p_r]}.$$

Given $R \in \mathbb{N}^{(\mathbb{N})}$ with $R(r) \neq 0$, we know $\widehat{R}$ is divisible by p_r and hence

$$\begin{aligned} f_P(s) \,!\, \widehat{R} &\to^+ f_{P+1}(s) \,!\, \widehat{R}/p_r \\ &= f_{P+1}(s) \,!\, R[r \leftarrow \widehat{R(r)} - 1], \end{aligned}$$

simulating a decrement. For $R(r) = 0$, we have

$$\begin{aligned} f_P(s) \,!\, \widehat{R} &\to^+ f_l(s) \,!\, p_r \left\lfloor \widehat{R}/p_r \right\rfloor + (\widehat{R} \bmod p_r) \\ &= f_l(s) \,!\, \widehat{R}, \end{aligned}$$

simulating a jump. Here, we exploited properties of the indexing functions of proj and zip noted earlier.

Finally, we need a stream function to produce the initial register encoding. This will be accomplished by

$$u(s) = \mathrm{head}(\mathrm{tail}(s)) :: u(\mathrm{proj}_{p_0}(s)).$$

By Lemma 1.1, we have

$$\overline{u}(i) = \overline{\mathrm{proj}_{p_0}}^{\,i}(1) = p_0^i = (\widehat{i, 0, \ldots}).$$

Defining $q(s) = u(f_0(s))$, we have

$$\overline{q}(i) = \overline{f_0}((\widehat{i, 0, \ldots})) = \mathrm{result}_{(L,I)}(\mathrm{init}(i))$$

for $i \in \mathbb{N}$, and thus $\overline{q} = \phi_{(L,I)}$. The equations for the stream functions $f_0, \ldots, f_L, u, q, \mathrm{const}$ plus finitely many instances of zip and proj hence represent a stream equation system with the denotation of q having $\phi_{(L,I)}$ as indexing function. □

Note that Minsky [44] shows that counter machines with only two registers are enough to achieve Turing-completeness. However, when representing computable functions, this requires an extra level of encoding of input values and decoding of output values, which can also be achieved by defining suitable stream functions.

1.4 Unary Definability

The defining feature of the previous construction was its reliance on interleaving for implementing conditional execution, effectively dispatching different cases, identified by their residues of the register encoding modulo a prime, to arbitrarily different handlers. Noting that zip_p with p prime were the only non-unary stream functions in the construction, we are left to reflect on the computational consequences of only allowing unary stream functions to be defined. Note that allowing interleaving is synonymous to allowing non-unary stream functions since we can use interleaving to merge any number of stream arguments into a single one. In order to prove an even more general definability result in the unary setting, entirely different techniques need to be developed, separating conditional execution and unbounded looping into orthogonal concepts.

Let us give an explicit definition of unary systems. Fix an equation index set E. *Unary stream terms* UStrTerm are inductive generated as follows:

$$\frac{}{\textsc{param} \in \textsc{UStrTerm}} \qquad \frac{e \in E \qquad \sigma \in \textsc{UStrTerm}}{\textsc{call}(e, \sigma) \in \textsc{UStrTerm}}$$

$$\frac{\sigma \in \textsc{UStrTerm}}{\textsc{tail}(\sigma) \in \textsc{UStrTerm}} \qquad \frac{\sigma, \sigma' \in \textsc{UStrTerm}}{\textsc{headcons}(\sigma, \sigma') \in \textsc{UStrTerm}}$$

A *unary (stream equation) system* then consists of a family $(\delta_e)_{e \in E}$ of unary stream term.

Again, in practice we will be more informal and write unary systems in the form $f_e(s) = \sigma$ where σ might be generated by the following grammar:

$\sigma ::=$	s		stream parameter,
	$\mid$	$\mathrm{tail}(\sigma)$	stream stripped of its first element,
	$\mid$	$\mathrm{head}(\sigma) :: \sigma'$	first element of a stream prepended to stream (head-cons),
	$\mid$	$f_e(\sigma)$	recursive stream function call with $e \in E$.

1.4.1 Collatz Functions and If-Programs

Definition 1.3. *A function $g : \mathbb{N} \to \mathbb{N}$ is called a* Collatz function *if there is $n > 0$ such that g is affine on each equivalence class modulo n, i.e. there are coefficients $a_i, b_i \in \mathbb{N}$ for $i = 0, \ldots, n-1$ such that $g(nq + i) = a_i q + b_i$ for $q \in \mathbb{N}$. In this case, n is called a* modulus *of g.*

The naming stems from the famous conjecture first proposed by Collatz in 1937, asking whether the function

$$\mathrm{collatz} : \mathbb{N} \to \mathbb{N},$$
$$n \mapsto \begin{cases} n/2 & \text{for } n \text{ even}, \\ 3n+1 & \text{for } n \text{ odd} \end{cases}$$

will map each positive integer to 1 after finitely many applications. Despite its deceivingly simple form, it has been resisting all attempts of resolution [41]. In recent years, the equivalent of this conjecture for the above generalized notion of Collatz functions, i.e. deciding whether iterations of a given generalized Collatz function eventually map each natural number input to e.g. zero, has been proved (algorithmically) undecidable [40].

Lemma 1.2. *Given a Collatz function g, we can construct a non-mutually recursive unary system defining a stream function v such that $\overline{v} = g$.*

Before going into the details of the proof, note that, although it is clear from the work cited above on reducing computation to generalized Collatz problems that the above encoding of

Collatz functions already enables an embedding of full computational power into unary stream equations, what is not at all obvious is whether we can actually define every computable unary stream function through a purely unary system.

Proof. Let modulus $n > 0$ and coefficients $a_i, b_i \in \mathbb{N}$ be as in the above definition. Define a stream function

$$\mathrm{add}(s) = \mathrm{head}(\mathrm{tail}^{na_0+0}(s)) :: \ldots :: \mathrm{head}(\mathrm{tail}^{na_{n-1}+(n-1)}(s)) :: \mathrm{add}(\mathrm{tail}^n(s)).$$

Lemma 1.1 shows that

$$\overline{\mathrm{add}}(k) = \left\lfloor \frac{k}{n} \right\rfloor \cdot n + (na_{k \bmod n} + (k \bmod n)) = k + na_{k \bmod n}$$

for $k \in \mathbb{N}$. The role of this function is to act as a crude replacement conditional for the unavailable zip, adding different constants depending on the equivalence class of the stream index modulo n.

Next, define a stream function

$$u(s) = \mathrm{head}(\mathrm{tail}^{nb_0+0}(s)) :: \ldots :: \mathrm{head}(\mathrm{tail}^{nb_{n-1}+(n-1)}(s)) :: u(\mathrm{add}(s)).$$

Using Lemma 1.1, we derive

$$\overline{u}(nq + i) = \overline{\mathrm{add}}^q(nb_i + i) = (nb_i + i) + q \cdot na_i = ng(k) + i$$

for $q \in \mathbb{N}$. This function is an approximation to g, the only difference being that the output indices come pre-multiplied by n.

We fix this by defining a stream function

$$\mathrm{div}(s) = \underbrace{\mathrm{head}(s) :: \ldots :: \mathrm{head}(s)}_{n \text{ times}} :: \mathrm{div}(\mathrm{tail}(s)).$$

A trivial application of Lemma 1.1 verifies that $\overline{\mathrm{div}}(k) = \lfloor k/n \rfloor$ for $k \in \mathbb{N}$. Finally defining

$$v(s) = u(\mathrm{div}(s))$$

yields the Collatz function semantics we want in that $\overline{v} = \overline{\mathrm{div}} \circ \overline{u} = g$. □

For instance, the original Collatz function would be encoded as $\mathrm{collatz} = \overline{v}$ as follows:

$$\begin{aligned}
\mathrm{add}(s) &= \mathrm{head}(\mathrm{tail}^2(s)) :: \mathrm{head}(\mathrm{tail}^{13}(s)) :: \mathrm{add}(\mathrm{tail}^2(s)),\\
u(s) &= \mathrm{head}(s) :: \mathrm{head}(\mathrm{tail}^9(s)) :: u(\mathrm{add}(s)),\\
\mathrm{div}(s) &= \mathrm{head}(s) :: \mathrm{head}(s) :: \mathrm{div}(\mathrm{tail}(s)),\\
v(s) &= u(\mathrm{div}(s)).
\end{aligned}$$

For further illustration, let us evaluate stream position 3 of $v(s)$:

$$\begin{aligned}
v(s)\,!\,3 &\to^* u(\mathrm{div}(s))\,!\,3\\
&\to^* u(\mathrm{add}(\mathrm{div}(s)))\,!\,1\\
&\to^* \mathrm{add}(\mathrm{div}(s))\,!\,9\\
&\to^* \mathrm{add}(\mathrm{tail}^2(\mathrm{div}(s)))\,!\,7\\
&\to^* \ldots\\
&\to^* \mathrm{add}(\mathrm{tail}^8(\mathrm{div}(s)))\,!\,1\\
&\to^* \mathrm{div}(s)\,!\,21\\
&\to^* \mathrm{div}(\mathrm{tail}(s))\,!\,19
\end{aligned}$$

$$\begin{aligned}&\to^* \dots\\&\to^* \operatorname{div}(\operatorname{tail}^{10}(s))\,!\,1\\&\to^* s\,!\,10.\end{aligned}$$

This is consistent with $\operatorname{collatz}(3) = 3 \cdot 3 + 1 = 10$.

The role of Collatz functions in our setting is to serve as an intermediate between indexing functions and the known world of computability. To make the latter link clearer, we will show how Collatz functions relate semantically to different register machine models under the prime factorization register encoding introduced in the previous section.

Definition 1.4. *The inductive set of* IF-programs *is generated by concatenation* $\mathbf{A}_0 \dots \mathbf{A}_{n-1}$*, increments* $\operatorname{inc}(r)$*, decrements* $\operatorname{dec}(r)$*, and conditional clauses* $\operatorname{ifz}(r, \mathbf{A}, \mathbf{B})$*, where* $n \in \mathbb{N}$*,* $r \in \mathbb{N}$ *designates a register, and* $\mathbf{A}_0, \dots, \mathbf{A}_{n-1}, \mathbf{A}, \mathbf{B}$ *are* IF*-programs.*

Although it is quite clear intuitively what the semantic effects of running an IF-program $\mathbf{A}$ on some register state $R \in \mathbb{N}^{(N)}$ are, we will formally introduce an associated semantics function $\chi_{\mathbf{A}} : \mathbb{N}^{(\mathbb{N})} \to \mathbb{N}^{(\mathbb{N})}$ defined structurally as follows:

$$\begin{aligned}\chi_{\mathbf{A}_0 \dots \mathbf{A}_{n-1}} &= \chi_{\mathbf{A}_{n-1}} \circ \dots \circ \chi_{\mathbf{A}_0},\\ \chi_{\operatorname{inc}(r)}(R) &= R[r \leftarrow R(r) + 1],\\ \chi_{\operatorname{dec}(r)}(R) &= R[r \leftarrow \max(R(r) - 1, 0)],\\ \chi_{\operatorname{ifz}(r,\mathbf{A},\mathbf{B})}(R) &= \begin{cases}\chi_{\mathbf{A}}(R) & \text{if } R(r) = 0,\\ \chi_{\mathbf{B}}(R) & \text{else.}\end{cases}\end{aligned}$$

Note that a decrement on a zero-valued register is ignored.

We will reuse the prime factorization register encoding

$$\begin{aligned}\widehat{\cdot} &: \mathbb{N}^{(\mathbb{N})} \to \mathbb{N} \setminus \{0\},\\ \widehat{R} &= \prod_{r \in \mathbb{N}, R(r) \neq 0} p_r^{R(r)}\end{aligned}$$

from the previous section. Translated to this setting, the semantics function of an IF-program $\mathbf{A}$ takes the form $\widehat{\chi}_{\mathbf{A}} := \widehat{\cdot} \circ \chi_{\mathbf{A}} \circ \widehat{\cdot}^{-1}$. Although this is an endofunction on the positive integers, to make the following treatment more uniform, we will extend it to the natural numbers by setting $\widehat{\chi}_{\mathbf{A}}(0) := 0$.

Lemma 1.3. *Given an if-program* $\mathbf{A}$*, its semantics* $\widehat{\chi}_{\mathbf{A}} : \mathbb{N} \to \mathbb{N}$ *on the register encoding is a Collatz function.*

Proof. By induction on the structure of $\mathbf{A}$, noting that:

- The concatenation of finitely many Collatz functions of moduli $m_0, \dots, m_{n-1}$ is a Collatz function of modulus $m_0 \cdot \dots \cdot m_{n-1}$.
- Given a register state $R \in \mathbb{N}^{(\mathbb{N})}$, an increment of register r corresponds to multiplication of $\widehat{R}$ with p_r, a Collatz function of modulus 1.
- Decrement of register r corresponds to division of $\widehat{R}$ by p_r if the former is divisible by p_r, and no change otherwise. This is a Collatz function of modulus p_r.
- Let $\widehat{\chi_{\mathbf{A}}}$ and $\widehat{\chi_{\mathbf{B}}}$ be Collatz functions of moduli $m_{\mathbf{A}}$ and $m_{\mathbf{B}}$, respectively. A conditional clause $\operatorname{ifz}(r, \mathbf{A}, \mathbf{B})$ corresponds first to case distinction depending on whether $\widehat{R}$ is divisible by p_r and subsequent application of either $\widehat{\chi_{\mathbf{A}}}$ or $\widehat{\chi_{\mathbf{B}}}$. This is a Collatz function of modulus the least common multiple of $p_r, m_{\mathbf{A}}, m_{\mathbf{B}}$.

□

1.4.2 Iteration-Programs and Their Encoding

Unsurprisingly, the expressive power of IF-programs by themselves is quite limited. To achieve computational completeness, we need an unbounded looping construct. The following definition intends to provide a minimal such model, enabling us to concentrate on the essential details of the conversion from Turing-complete programs to stream equation systems.

Definition 1.5. *An* ITERATION-program $\mathbf{P}$ *is a tuple* $(\mathbf{Body_P}, \mathsf{input_P}, \mathsf{output_P}, \mathsf{loop_P})$ *consisting of an* IF*-program* $\mathbf{Body_P}$ *called the* body *of* $\mathbf{P}$ *and designated and mutually distinct* input*,* output *and* loop *registers* $\mathsf{input_P}, \mathsf{output_P}, \mathsf{loop_P} \in \mathbb{N}$.

The semantics of such a program is a computable function $\phi_{\mathbf{P}} : \mathbb{N} \to \mathbb{N}_\perp$ defined as follows: given an input $i \in \mathbb{N}$, the register state $R_0 \in \mathbb{N}^{(\mathbb{N})}$ is initialized with $R_0(\mathsf{input_P}) := i$, $R_0(\mathsf{loop_P}) := 1$, and $R_0(r) := 0$ for $r \neq \mathsf{input_P}, \mathsf{loop_P}$. We iteratively execute the body of $\mathbf{P}$, yielding $R_{n+1} := \chi_{\mathbf{Body_P}}(R_n)$ for $n \in \mathbb{N}$. For given input i, if there is n such that $R_n(\mathsf{loop_P}) = 0$, then $\mathbf{P}$ is called *terminating* with *iteration count* $\operatorname{count} P(i) := n$ and output $\phi_{\mathbf{P}}(i) := R_n(\mathsf{output_P})$ for the smallest such n. Otherwise, we set $\operatorname{count}_{\mathbf{P}}(i) := \perp$ and $\phi_{\mathbf{P}}(i) := \perp$.

Intuitively, an ITERATION-program is just a WHILE-program [48] with a single top-level loop, a well-studied concept in theoretical computer science bearing resemblance to the normal form theorem for μ-recursive functions [34, 54] except that we do not even allow primitive recursion inside the loop.

Theorem 1.1. *Given a computable function* $\phi : \mathbb{N} \to \mathbb{N}_\perp$*, there is an* ITERATION*-program* $\mathbf{P}$ *with semantics* $\phi_{\mathbf{P}} = \phi$.

Proof. This is a folklore theorem [23], see Böhm and Jacopini [9] and Perkowska [48] for more details. □

The reason behind our choice for this computationally complete machine model is that we already have the machinery to simulate a single execution of the body of such a machine via Collatz functions as indexing functions of stream equations using our prime factorization exponential encoding on the register state.

We will now investigate how to translate a top-level unbounded looping construct into the recursive stream equation setting. For this, we require a further technical result about stream equations of a certain shape:

Lemma 1.4. *Consider a stream equation system containing equations for stream functions* f, u, v*. Assume the equation for* f *has the form*

$$f(s) = \operatorname{head}(\operatorname{tail}^{a_0}(s)) :: \ldots :: \operatorname{head}(\operatorname{tail}^{a_{h-1}}(s)) :: \operatorname{tail}^h(u(f(v(s))))$$

where $h \geq 1$.

Fix $k \in \mathbb{N}$ *and choose* $c(k) \in \mathbb{N}$ *minimal such that* $d(k) := \overline{u}^{c(k)}(k) \in [h]_\perp$*. If such a* $c(k)$ *exists and* $d(k) \neq \perp$*, then* $\overline{f}(k) = \overline{v}^{c(k)}(a_{d(k)})$*, otherwise* $\overline{f}(k) = \perp$.

Proof. The proof is by induction on $c(k)$ if existent. At the base, $c(k) = 0$ is equivalent to $k < h$. In this case,

$$\begin{aligned} f(s) \,!\, k &\to^{k+1} \operatorname{tail}^{a_k}(s) \,!\, 0 \\ &\to^{a_k} \quad s \,!\, a_k, \end{aligned}$$

i.e. $\overline{f}(k) = a_k = \overline{v}^{c(k)}(a_{d(k)})$.

Now assume $k \geq h$. Note that

$$\begin{aligned} f(s) \,!\, k &\to^{h+1} \operatorname{tail}^h(u(f(v(s)))) \,!\, k - h \\ &\to^{h} \quad u(f(v(s))) \,!\, k. \end{aligned}$$

If $\overline{u}(k) = \bot$, then $c(k) = 1$, $d(k) = \bot$, and $\overline{f}(k) = \bot$. In the remainder, we will assume $\overline{u}(k) \neq \bot$. Then,

$$f(s) \,!\, k \to^+ f(v(s)) \,!\, \overline{u}(k),$$

and $\overline{f}(k) = \overline{v}(\overline{f}(\overline{u}(k)))$.

If $c(k)$ is defined, then $c(k) = c(\overline{u}(k)) + 1$ and we can apply the induction hypothesis: if $d(k) = d(\overline{u}(k)) \neq \bot$, then

$$\overline{v}(\overline{f}(\overline{u}(k))) = \overline{v}(\overline{v}^{c(\overline{u}(k))} a_{d(k)}) = \overline{v}^{c(k)}(a_{d(k)}),$$

otherwise $\overline{v}(\overline{f}(\overline{u}(k))) = \overline{v}(\bot) = \bot$.

If $c(k)$ is undefined, then so is $c(\overline{u}(k))$, and with a second induction we can construct an infinite sequence

$$f(s) \,!\, k \to^+ f(v(s)) \,!\, \overline{u}(k) \to^+ f(v^2(s)) \,!\, \overline{u}^2(k) \to^+ \ldots$$

showing non-termination and $\overline{f}(k) = \bot$. □

The inquiring reader will notice that this lemma can be seen as a generalization of Lemma 1.1 with u defined in a particular way, namely

$$u(s) = \underbrace{\mathrm{head}(s) :: \ldots :: \mathrm{head}(s)}_{h \text{ times}} :: s.$$

Using Lemmata 1.2 and 1.3, we can translate the encoded iteration step function $\widehat{\chi}_{\mathbf{Body_P}} : \mathbb{N} \to \mathbb{N}$ of an ITERATION-program $\mathbf{P}$ to an indexing function of a stream equation for some u. We would like to use this stream function u as it appears in Lemma 1.4 in a way such that the minimal choice of $c(k)$ corresponds to the iteration count of $\mathbf{P}$. Unfortunately, the equivalent of the stopping condition in the lemma, that the index be smaller than some constant h, corresponds to $\widehat{R} < h$ for the register state $R \in \mathbb{N}^{(\mathbb{N})}$, a statement which does not have a natural meaning in terms of the registers of R individually, forestalling us from expressing the condition $p_{\mathsf{loop}_\mathbf{P}} \mid \widehat{R}$ corresponding to the termination condition $R(\mathsf{loop}_\mathbf{P}) = 0$. A second problem comes from our desire to somehow extract the value of $R(\mathsf{output}_\mathbf{P})$ after termination. But since at this point of time $\widehat{R}$ is limited to a finite set of values, there is no direct way of realizing this.

What we can do is extract the iteration count for particularly nicely behaving programs.

Lemma 1.5. *Given an* ITERATION*-program* $\mathbf{Q}$ *such that whenever* $\mathbf{Q}$ *terminates, all its registers are zero-valued, i.e.* $\chi_{\mathbf{Body_Q}}^{\mathrm{count}_\mathbf{Q}(i)} = (0, 0, \ldots)$ *for terminating input* $i \in \mathbb{N}$*, there is a non-mutually recursive unary system defining a stream function* w *such that* $\overline{w} = \mathrm{count}_\mathbf{Q}$.

Proof. In anticipation of applying Lemma 1.4, we extend this system with a new equation

$$q(s) = \mathrm{head}(s) :: \mathrm{head}(s) :: \mathrm{tail}^2(v(q(\mathrm{tail}(s)))).$$

Given an input $i \in \mathbb{N}$ and corresponding initial register state $R \in \mathbb{N}^{(\mathbb{N})}$, the termination condition in Lemma 1.4 can equivalently be expressed as follows:

$$\begin{aligned} \overline{v}^{c(\widehat{R})}(\widehat{R}) < 2 &\iff \widehat{\chi}_{\mathbf{Body_Q}}^{c(\widehat{R})}(\widehat{R}) = 1 \\ &\iff \chi_{\mathbf{Body_Q}}^{c(\widehat{R})}(R) = (0, 0, \ldots). \end{aligned}$$

Now, by our assumption on the behaviour of $\mathbf{Q}$, the first point in time all registers are zero equals the first point in time the loop register attains zero. But by our definition of the iteration count, this just means that $c(\widehat{R}) = \mathrm{count}_\mathbf{Q}(i)$, and Lemma 1.4 shows that

$$\overline{q}(\widehat{R}) = \overline{\mathrm{tail}}^{c(\widehat{R})}(0) = c(\widehat{R}) = \mathrm{count}_\mathbf{Q}(i).$$

All that remains is to produce the initial register state $R_{(i)}$ with only $R_{(i)}(\mathsf{input}_{\mathbf{Q}}) = i$ and $R_{(i)}(\mathsf{loop}_{\mathbf{Q}}) = 1$ non-zero. For this, we define

$$r(s) = \mathrm{head}(\mathrm{tail}^{p_{\mathsf{loop}_{\mathbf{Q}}}}(s)) :: r(\mathrm{proj}_{p_{\mathsf{input}_{\mathbf{Q}}}}(s))$$

and utilize Lemma 1.1 to prove that

$$\overline{r}(i) = \overline{\mathrm{proj}_{p_{\mathsf{input}_{\mathbf{Q}}}}}^{i}(p_{\mathsf{loop}_{\mathbf{Q}}}) = p^{i}_{\mathsf{input}_{\mathbf{Q}}} \cdot p_{\mathsf{loop}_{\mathbf{Q}}} = \widehat{R_{(i)}}.$$

Defining $w(s) = r(q(s))$, we verify that

$$\overline{w}(i) = \overline{q}(\overline{r}(i)) = \overline{q}(\widehat{R_{(i)}}) = \mathrm{count}_{\mathbf{Q}}(R_{(i)}).$$

□

Unfortunately, the set of possible iteration count functions constitutes only a small part of the set of all computable functions. Intuitively, this is because even very small values can be the result of prohibitively expensive operations. However, this range can still be seen as containing Turing-complete fragments under certain encodings. This is what we exploit in the next step by shifting the role of the output register to the iteration count under a particular such encoding. The trick is to have each possible output value correspond to infinitely many iteration counts in a controlled way such that after having computed the result, by being self-aware of the current iteration count, we can consciously terminate the loop at one of these infinitely many counts, no matter how long the computation took.

Lemma 1.6. *Given an* ITERATION*-program* $\mathbf{P}$*, there is an* ITERATION*-program* $\mathbf{Q}$ *such that for every input* i *natural,* $\mathbf{Q}$ *terminates if and only if* $\mathbf{P}$ *terminates, and furthermore if* $\mathbf{P}$ *terminates with output* $o \in \mathbb{N}$*, then* $\mathbf{Q}$ *terminates after exactly* $(3m+1) \cdot 3^{o+1}$ *iterations with all registers zero-valued where* $m \in \mathbb{N}$ *depends on* i*.*

Proof. Let $r_0, \ldots, r_{k-1} \in \mathbb{N}$ denote all the registers occurring in $\mathbf{Body}_{\mathbf{P}}$ except for $\mathsf{output}_{\mathbf{P}}$ and $\mathsf{loop}_{\mathbf{P}}$ (but including $\mathsf{input}_{\mathbf{P}}$). We choose $\mathsf{loop}_{\mathbf{Q}}$ as a fresh natural number distinct from all previously mentioned registers. Both programs will have the same input register, i.e. $\mathsf{input}_{\mathbf{Q}} := \mathsf{input}_{\mathbf{P}}$. The output register of $\mathbf{Q}$ is irrelevant since we aim to have all registers reset at termination.

The body of $\mathbf{Q}$ is listed in Figure 1.1. Note that the body of $\mathbf{P}$ is textually inserted at line 14. Register names main-phase, run-time, mod-three, swap-phase, copy also designate fresh natural numbers. To enhance readability, we used some lyrical freedom with the syntax: for example, **if** $R(\mathsf{output}_{\mathbf{P}}) \neq 0$ **then A else B end if** translates to $\mathrm{ifz}(\mathsf{output}_{\mathbf{P}}, \mathbf{B}, \mathbf{A})$. Since the program is somewhat complex, we will describe its function in great detail.

Execution of $\mathbf{Q}$, i.e. iterated execution of $\mathbf{Body}_{\mathbf{Q}}$ until decrement of $\mathsf{loop}_{\mathbf{Q}}$, is split into two main phases, as signalled by the flag register main-phase. The first phase, when main-phase has value 0 (lines 2–22), is dedicated to simulating the original program $\mathbf{P}$ while keeping track of the total iteration count in a dedicated register run-time. At the end of this phase, after $\mathbf{P}$ has exited with result $o \in \mathbb{N}$ in register $\mathsf{output}_{\mathbf{P}}$, we want the total iteration count to equal $3m+1$ for some arbitrary $m \in \mathbb{N}$. The second phase, when main-phase has value 1 (lines 23–44) is dedicated to tripling the total iteration count o times, plus an additional tripling to reset run-time, so that the total iteration count becomes $(3m+1) \cdot 3^{o+1}$.

In detail, the first phase (lines 2–22) contains three separate components:

- Lines 3–4 are executed only at the beginning of the first iteration and initialize the loop register of $\mathbf{P}$ (note that the input register of $\mathbf{P}$ does not need to be initialized as $\mathsf{input}_{\mathbf{P}} = \mathsf{input}_{\mathbf{Q}}$) and mod-three (see below).
- Lines 6-12 keep track not only of the current iteration count by incrementing run-time once per iteration, but also of how many iterations modulo 3 we are afar from meeting the $(3m+1)$-condition in a dedicated register mod-three.

```
 1. if R(main-phase) = 0 then
 2.     if R(run-time) = 0 then
 3.         inc(loop_P)
 4.         inc(mod-three)
 5.     end if
 6.     inc(run-time)
 7.     if R(mod-three) = 0 then
 8.         inc(mod-three)
 9.         inc(mod-three)
10.         inc(mod-three)
11.     end if
12.     dec(mod-three)
13.     if R(loop_P) ≠ 0 then
14.         Body_P
15.     else if R(r_0) ≠ 0 then
16.         dec(r_0)
17.     [...]
18.     else if R(r_{n-1}) ≠ 0 then
19.         dec(r_{n-1})
20.     else if R(mod-three) = 0 then
21.         inc(main-phase)
22.     end if
23. else if R(swap-phase) = 0 then
24.     dec(run-time)
25.     inc(copy)
26.     if R(run-time) = 0 then
27.         inc(swap-phase)
28.     end if
29. else
30.     dec(copy)
31.     if R(output_P) ≠ 0 then
32.         inc(run-time)
33.         inc(run-time)
34.         inc(run-time)
35.         if R(copy) = 0 then
36.             dec(swap-phase)
37.             dec(output_P)
38.         end if
39.     else if R(copy) = 0 then
40.         dec(swap-phase)
41.         dec(main-phase)
42.         dec(loop_Q)
43.     end if
44. end if
```

Figure 1.1: The body of **Q** from Lemma 1.6

- Lines 13–22 execute the body of **P** once per iteration until termination is signalled by $\mathsf{loop}_{\mathbf{P}}$ being set to zero (lines 13–14). In subsequent iterations, the registers used in **P** are incrementally reset (lines 15–19). Finally, we wait up to two iterations for the iteration count (including the current iteration) to have the proper remainder modulo 3 (line 20), and proceed to the second phase (lines 21).

After the final iteration of this phase, the iteration count is $3m + 1$ for some $m \in \mathbb{N}$ and the only possibly non-zero registers are main-phase and swap-phase of value 1 and $\mathsf{output}_{\mathbf{P}}$ of value o. We duly note that if **P** does not terminate, then neither does **Q**.

In similar detail, the second phase (lines 23–44) contains two alternately executed subphases (lines 24–28 and 30–43) responsible for shifting the iteration count back and forth between the registers run-time and copy. The current subphase is indicated by the flag register swap-phase:

- The first subphase in lines 24–28 started with register values $[\mathsf{swap\text{-}phase} : 1, \mathsf{run\text{-}time} : x \geq 1, \mathsf{copy} : 0]$ will end, after x iterations, with values $[\mathsf{swap\text{-}phase} : 0, \mathsf{run\text{-}time} : 0, \mathsf{copy} : x]$.
- The second subphase in lines 30–43 started with register values $[\mathsf{swap\text{-}phase} : 0, \mathsf{run\text{-}time} : 0, \mathsf{copy} : x \geq 1]$ will end, after x iterations, depending on the value of register $\mathsf{output}_{\mathbf{P}}$,
  - if non-zero, with $[\mathsf{swap\text{-}phase} : 1, \mathsf{run\text{-}time} : 3x, \mathsf{copy} : 0]$ and $\mathsf{output}_{\mathbf{P}}$ decremented,
  - if zero, with all registers zero and $\mathsf{loop}_{\mathbf{Q}}$ decremented to zero in the last iteration.

Taken together, we deduce that starting (the first subphase) with $[\mathsf{swap\text{-}phase} : 1, \mathsf{run\text{-}time} : x \geq 1, \mathsf{copy} : 0]$ and $\mathsf{output}_{\mathbf{P}}$ non-zero, after $2x$ iterations, the effective changes will be tripling of run-time and decrement of $\mathsf{output}_{\mathbf{P}}$. In particular, if x and hence run-time denoted the iteration count before these iterations, run-time will again denote the iteration count after these iterations. After following this reasoning o times, the iteration count and value of run-time will be $(3m + 1) \cdot 3^o$ while $\mathsf{output}_{\mathbf{P}}$ attains zero. One last instance of each phase, costing $2 \cdot (3m+1) \cdot 3^o$ iterations, yield a total iteration count of $(3m+1) \cdot 3^{o+1}$ with all registers having been cleared. □

1.4.3 Proof of the Main Result

Theorem 1.2. *A unary polymorphic stream function is definable by a non-mutually recursive unary system if and only if its indexing function is computable.*

Proof. We need only consider the reverse implication. Let a computable function $\phi : \mathbb{N} \to \mathbb{N}_\perp$ be represented as an ITERATION-program $\mathbf{P}$, i.e. $\phi = \phi_P$, and let $\mathbf{Q}$ be the modified ITERATION-program as defined in Lemma 1.6. By Lemma 1.5, there is a non-mutually recursive unary system defining a stream function w such that $\overline{w}(i) = \mathrm{count}_{\mathbf{Q}}(i) = (3m+1) \cdot 3^{\phi_{\mathbf{P}}(i)+1}$ for all $i \in \mathbb{N}$ and some $m \in \mathbb{N}$ depending on i. As stated at the beginning, the latter expression is taken to mean $\perp$ if $\phi_{\mathbf{P}}(i) = \perp$.

Our strategy for extracting the final output value $\phi_{\mathbf{P}}(i)$ from this expression is by iterating a second program, adding a tail for each time the stream index is divisible by 3. In particular, from Lemma 1.2, it is clear how to give a non-mutually recursive unary system defining u such that

$$\overline{u}(k) = \begin{cases} k/3 & \text{if } 3 \mid k, \\ 0 & \text{else} \end{cases}$$

since this is a Collatz function. But note that we can alternatively directly define

$$\begin{aligned} u(s) = {} & \mathrm{head}(s) :: \mathrm{head}(s) :: \mathrm{head}(s) :: \\ & \mathrm{head}(\mathrm{tail}(s)) :: \mathrm{head}(s) :: \mathrm{head}(s) :: \mathrm{tail}^3(u(\mathrm{head}(s) :: \mathrm{tail}^2(s))) \end{aligned}$$

using only a single equation. For either choice, we define

$$v(s) = \mathrm{head}(s) :: \mathrm{tail}(u(v(\mathrm{tail}(s)))).$$

A second application of Lemma 1.4 shows that $\overline{v}((3m+1) \cdot 3^{i+1}) = i+1$ for $i, m \in \mathbb{N}$. This function is of almost as critical importance as w as it repeats each natural number output infinitely many times in a controlled way and reverses the iteration count result encoding of program $\mathbf{Q}$.

We now have all the parts necessary for concluding our venture. Defining

$$f(s) = w(v(\mathrm{head}(s) :: s)),$$

we see that

$$\begin{aligned} \overline{f}(i) &= \max(\overline{v}(\overline{w}(i)) - 1, 0) \\ &= \max(\overline{v}((3m+1) \cdot 3^{\phi_{\mathbf{P}}(i)+1}) - 1, 0) \\ &= \max((\phi_{\mathbf{P}}(i) + 1) - 1, 0) \\ &= \phi_{\mathbf{P}}(i) \end{aligned}$$

with our usual convention regarding $\perp$, proving $\overline{f} = \phi_{\mathbf{P}}$. □

1.5 Unary Singleton Systems

In an effort to restrict the shape of our systems even further so that we finally arrive at some positive results, let us now consider unary systems consisting of a single equation, i.e. systems $f(s) = \sigma$ where σ is a unary stream term in s with calls only to f. As it turns out, given a unary system of size one, we can completely determine the structure of its indexing function in finite time. The proof of this amazing fact is somewhat involved, requiring a certain apparatus. We start with the first important step:

Theorem 1.3. *Given a single unary stream function equation u, with no references to other stream functions, we can decide if $\overline{u}(k) = \perp$ for $k \in \mathbb{N}$, i.e. we can compute $\overline{u}(k) \in \mathbb{N} \cup \{\perp\}$ in finite time.*

Proof. We can assume the definition of u has the form

$$u(s) = \rho[s] = \mathrm{head}(\alpha_0[s]) :: \ldots :: \mathrm{head}(\alpha_{h-1}[s]) :: \mathrm{tail}^t(u(\beta[s]))$$

where $\rho[s], \alpha_0[s], \ldots, \alpha_{h-1}[s], \beta[s]$ are stream terms in s with calls only to u.

Define $p := h - t$. First, we deal with the case $p \leq 0$, in which u is non-productive at indices $k \geq h$:

$$u(s) \,!\, k \to^+ u(\beta[s]) \,!\, k - p \to^+ u(\beta[\beta[s]]) \,!\, k - 2p \to^+ \ldots$$

For indices $k < h$, we iteratively check if any computation $u(s) \,!\, k$ goes through without other references to $u(\ldots) \,!\, k'$ for any yet unresolved $k' \in \mathbb{N}$, i.e. not $u(s) \,!\, k \to^+ u(\ldots)k'$. Since there are only finitely many indices smaller than h, this coinductive process eventually terminates, leaving us with those indices $k < h$ at which u is unproductive, having explicitly constructed loops for these.

In the following, we assume $p > 0$. We define a partial order $\preceq$ on $\mathbb{N}$, setting $a \preceq b$ if $t \leq a \leq b$ and $a \equiv b \mod p$. The intuition is that, given $a \prec b$, we have a reduction $u(s) \,!\, b \to^+ u(\ldots) \,!\, a$, i.e. $\overline{u}(a) = \bot$ implies $\overline{u}(b) = \bot$. We note this order has finite width, the maximum size of a set of mutually incomparable elements being h. We extend the order to the set $\mathbb{P}(\mathbb{N})$ of subsets of $\mathbb{N}$ by setting $A \preceq B$ if for all $b \in B$ there is $a \in A$ such that $a \preceq b$, noting that transitivity is preserved. We set $A \prec B$ if $A \preceq B$ but not $B \preceq A$. Since $(\mathbb{N}, \preceq)$ has finite width, we can derive well-foundedness of $(\mathbb{P}(\mathbb{N}), \prec)$ using a well-known result from the theory of well-quasi-orderings [39]. [5]

We now give a terminating algorithm for calculating $\overline{u}(k)$ for arbitrary $k \in \mathbb{N}$, enabling us to decide productivity at a given index. For elegance, we state the algorithm in functional form, taking as arguments a stream term with calls only to u, an index to compute at, and a *history* parameter $H \in \mathbb{P}(\mathbb{N})$ with H finite, keeping track of the indices k for which we are currently trying to compute $u(\ldots) \,!\, k$. In the following, the symbol $\bot$ is treated as just another symbol, without its usual connotation.

$$\begin{aligned}
r(\lambda s.\sigma[s],\ '\bot', H) &=' \bot' \\
r(\lambda s.s, n, H) &= n, \\
r(\lambda s.\,\mathrm{tail}(\sigma[s]), n, H) &= r(\lambda s.\sigma[s], n+1, H), \\
r(\lambda s.\,\mathrm{head}(\sigma[s]) :: \sigma'[s], 0, H) &= r(\lambda s.\sigma[s], 0, H), \\
r(\lambda s.\,\mathrm{head}(\sigma[s]) :: \sigma'[s], n+1, H) &= r(\lambda s.\sigma'[s], n, H), \\
r(\lambda s.u(\sigma[s]), n, H) &= \text{if } H \preceq \{n\} \text{ then } '\bot' \text{ else} \\
&\quad \text{let } m = r(\lambda s.\rho[s], n, H \cup \{n\}) \\
&\quad \text{in } r(\lambda s.\sigma[s], m, H).
\end{aligned}$$

Correctness The algorithm only deviates from the canonical operational semantics in the branch $H \preceq \{n\}$ of the last equation, where we have $m \in H$ such that $m \preceq n$. But this means computing $u(s) \,!\, m$ depends on computing $u(\ldots) \,!\, n$, which, by construction of H, equals or in turn depends on computing $u(\ldots) \,!\, m$, i.e. formally $u(s) \,!\, m \to^+ u(\ldots)n \to^* u(\ldots) \,!\, m$, meaning the computation will loop. Thus, the result $\bot$ is justified.

Termination Since $H \not\preceq \{n\}$ implies $H \cup \{n\} \prec H$, recursive calls either involve a smaller history parameter $H' \in \mathbb{P}(\mathbb{N})$ or the same history parameter with a structurally smaller stream term argument. Since the corresponding lexicographical ordering is well-founded, termination is guaranteed. □

[5] Sketch of a direct proof: for an infinite chain $A_0 \succ A_1 \succ \ldots$, there must be elements $a_n \in A_n$ for $n \geq 1$ such that $a_m \succeq a_n$ implies $m \leq n$. By well-foundedness of $\prec$, there is $i_0 \in \mathbb{N}$ such that a_{i_0} is minimal in $\{a_n \mid n \in \mathbb{N}\}$. We inductively construct indices $i_j \in \mathbb{N}$ with $j \in \mathbb{N}$ such that $a_{i_{j+1}}$ is minimal in $\{a_n \mid n > i_j\}$. By construction of $(a_n)_{n\in\mathbb{N}}$ and choice of i_j, the elements a_{i_j} are mutually incomparable for $j \in \mathbb{N}$, contradicting finite width of $(\mathbb{N}, \preceq)$.

It was pointed out by the Internal Examiner that the argument above may be generalized from the unary singleton system context to an equation for a unary stream function u in any system subject to the following conditions:

- the indexing functions of all other stream functions in the system are already known to have decidable partiality,
- the outermost call the stream term defining u is to u itself.

We encourage the reader to in particular engage in proofreading the following theorem.

Theorem 1.4. *Given a unary system of size one, i.e. an equation for a single unary stream function u with no calls to other stream functions, we can completely determine the structure of $\overline{u}$. Aside from finitely many initial values, the indexing function of u essentially is a Collatz function except for non-productive equivalence classes.*

Concretely, there are $p > 0$ and $l \geq 0$ such that for each equivalence class of i modulo p with $0 \leq i < p$, either we can determine coefficients $a_i, b_i \in \mathbb{Q}$ such that $\overline{u}(k) = a_i k + b_i$, or $\overline{u}(k) = \bot$, both for all $k \equiv i \mod p$ with $k \geq l$.

Proof. We are given a single unary stream function equation, with no references to other stream functions. We can assume the definition has the form

$$\begin{aligned}
u_0 \quad (s) &= \mathrm{head}(v_{0,0} \quad (s)) :: \ldots :: \mathrm{head}(v_{0,f-1} \quad (s)) :: \mathrm{tail}^{b_0} \quad (u_0(u_1(s))) \\
u_1 \quad (s) &= \mathrm{head}(v_{1,0} \quad (s)) :: \ldots :: \mathrm{head}(v_{1,f-1} \quad (s)) :: \mathrm{tail}^{b_1} \quad (u_0(u_2(s))) \\
&\ldots \\
u_{e-1}(s) &= \mathrm{head}(v_{e-1,0}(s)) :: \ldots :: \mathrm{head}(v_{e-1,f-1}(s)) :: \mathrm{tail}^{b_{e-1}}(u_0(u_e(s))) \\
u_e \quad (s) &= \mathrm{head}(v_{e,0} \quad (s)) :: \ldots :: \mathrm{head}(v_{e,f-1} \quad (s)) :: \mathrm{tail}^{b_e} \quad (s)
\end{aligned}$$

where $v_{i,j}$ are arbitrary stream functions with only the head of $v_{i,j}(s)$ being of interest. We note that we used Theorem 1.3 to make equal the lengths of the initial head-cons parts of u_i.

In order to analyze the computational behaviour of $\overline{u_0}$, we need a formalism to explicitly represent the running computation. Given a functional relation $H \subseteq \mathbb{N} \times \mathbb{N}$ called *history* of already computed argument-value pairs of $\overline{u_0}$ and argument $k \in \mathbb{N}$, computation of $\overline{u_0}$ at argument k has *representation* a series of pairs $R_k := (x_k^{(t)}, y_k^{(t)})_{t=0,\ldots,q-1} \subseteq H$ with $q \leq e$ such that $x_k^{(t)} = y_k^{(t-1)} + (b_t - f)$ for $t < q$ with $y_k^{(-1)} := k$, and $y_k^{(q-1)} < f$ or $q = e$. In the former case, the resulting value is $v = \overline{v_{q,y_k^{q-1}}}(0) \neq \bot$, else it is $v = y_k^{(e-1)} + (b_e - f)$, yielding a new argument-value-pair of (k, v). If such a series of pairs does not exist in H, computation at argument k has *unresolved dependencies* in H. If it does exist, it is unique and independent of H. We have $\overline{u_0}(k) = \bot$ exactly if the empty history cannot be grown to include k in a finite series of such representation computation steps.

If $f \leq b_0$, computation of $\overline{u_0}(i)$ depends on computation of $\overline{u_0}(i+(b_0-f))$ for $i \geq f$, creating an infinite loop and hence non-productivity at all indices not smaller than f. Thus, we will assume period $:= f - b_0 > 0$ in all of the following. Similarly, if $b_e \leq f$, a proof by induction shows that $\overline{u_0}(i)$ is either $\bot$ or bounded by $c := \max(\overline{v_{i,j}}(0))$ where $j < f$ with $\overline{v_{i,j}}(0) \neq \bot$. Because $\overline{u_0}(i)$ is a function of $\overline{u_0}(i - f)$ for $i \geq f$, this means $\overline{u_0}$ eventually becomes periodic with the end of the first period bounded by $f + \mathrm{period}^c$. Thus, we will assume $f < b_e$ in all of the following.

Our algorithm is highly recursive, depending on hierarchically ordered parameters $a \in A$, $b \in B_a$, $c \in C_{a,b}$. Each of the parameters a, b, c is composed of certain verified data, termed *knowledge*. Furthermore, each parameter comes with an associated set of *assumptions* on the structure of $\overline{u_0}$ for larger values. These assumptions may or may not be valid, but the key is that they only have to hold for the finite part of $\overline{u_0}$ the algorithm ends up checking. If we notice a *violation* of the assumptions for a, b, or c during procession of $\overline{u_0}$, parameter sets $A, B_a, C_{a,b}$ come with instructions on how to deal with them by constructing new parameters (a', b_0, c_0), (a, b', c_0), or (a, b, c') on which the algorithm can be restarted. Here, we have $a' < a$, $b < b'$, or

$c < c'$, respectively, where each parameter sets comes with an associated well-ordering. Well-foundedness of the lexicographical ordering on $(a : A) \times (b : B_a) \times C_{a,b}$ means this restart happens only a finite number of times, guaranteeing termination in the end.

Parameter set A The first parameter takes care of which argument equivalence classes modulo period eventually become non-productive or repeating under $\overline{u}$ and which equivalence classes share the same value sequences.

Knowledge For $a \in A$,

- a set $\text{Nonprod}^{(a)} \subseteq \mathbb{Z}/(p)$ of *non-productive* equivalence classes eventually becoming non-productive under $\overline{u_0}$,
- a set $\text{Repeat}^{(a)} \subseteq \mathbb{Z}/(p)$ of *repeating* equivalence classes eventually becoming periodic under $\overline{u_0}$,
- for each repeating class C, elements $k_1 < k_2$ of C such that $\overline{u_0}(k_1) = \overline{u_0}(k_2)$,
- an equivalence relation $\sim^{(a)}$ on $\mathbb{Z}/(p) \setminus \text{Repeat}$,
- for all classes $C \sim^{(a)} C'$, an offset $\text{offset}^{(a)}_{C,C'} \in C' - C$,
- a bound $\text{bound}^{(a)} \in \mathbb{N}$ such that $\overline{u_0}(k) = \bot$ for all $k \in C \in \text{Nonprod}^{(a)}$ with $k \geq \text{bound}^{(a)}$, all repeating equivalence classes complete there first period before $\text{bound}^{(a)}$, and for non-repeating classes $C \sim^{(a)} C'$ and $k \in C$ with $k \geq \text{bound}^{(a)}$ we have $\overline{u_0}(k) = \overline{u_0}(k + \text{offset}^{(a)}_{C,C'})$.

Assumptions

- For all $k \geq \text{bound}^{(a)}$ with $\overline{k} \notin \text{Nonprod}^{(a)}$, we have $\overline{u_0}(k) \neq \bot$.
- For all $k, k' \geq f$ with $\overline{u_0}(k) = \overline{u_0}(k')$ and at least one of $[k], [k']$ non-repeating, we have $[k] \sim^{(a)} [k']$ and $k' = k + \text{offset}_{[k],[k']}$.

Ordering For $a, a' \in A$, we set $a < a'$ if $\text{Nonprod}^{(a)} \supsetneq \text{Nonprod}^{(a')}$, or equality holds and $\text{Repeat}^{(a)} \supsetneq \text{Repeat}^{(a')}$, or equality holds and $\sim^{(a)} \supsetneq \sim^{(a')}$. Given $a \in A$, there are only 2^p possibilities each for $\text{Nonprod}^{(a)}$ and $\text{Repeat}^{(a)}$ and $2^{p \cdot p}$ possibilities for $\sim^{(a)}$, so it is clear that $(A, <)$ is well-founded.

Dealing with Violation

- Given $k \geq \text{bound}^{(a)}$ with $\overline{k} \notin \text{Nonprod}^{(a)}$ but $\overline{u_0}(k) = \bot$, we can define $a' \in A$ with $a' < a$ fulfilling the knowledge part of a' by defining $\text{Nonprod}^{(a')} := \text{Nonprod}^{(a)} \cup \{\overline{k}\}$ since $\overline{u_0}(k + m \cdot \text{period}) = \bot$ by induction on $m \in \mathbb{N}$.
- Given $\overline{u_0}(k) = \overline{u_0}(k')$ with $k, k' \geq f$ and at least one of $[k], [k']$ non-repeating, but not $[k] \sim^{(a)} [k']$ and $k' = k + \text{offset}_{[k],[k']}$, it is clear we can define $a' \in A$ with $a' < a$ fulfilling the knowledge part of a'. If none of $[k], [k']$ are repeating in a and $[k] \not\sim^{(a)} [k']$, this is achieved by merging the classes of $[k]$ and $[k']$ in $\sim^{(a')}$. Otherwise, it is achieved by widening the set of repeating classes $\text{Repeat}^{(a')}$.

Start The starting parameter a_0 has non-productivity set $\mathrm{Nonprod}^{(a_0)}$, repetition set $\mathrm{Repeat}^{(a_0)}$, and equality relation $\sim^{(a')}$ empty. All defined offsets are zero, and $\mathrm{bound}^{(a_0)} = f$.

Lemma 1.7. *Given $a \in A$ and finite $S \subseteq \{f, f+1, \ldots\}$ such that no $s \in S$ is non-productive or repeating for a, if $|\overline{u_0}(S)|\,\mathrm{period} < |S|$, the assumption part of $a \in A$ is violated.*

Proof. A trivial application of the pigeonhole principle. □

Lemma 1.8. *Given $a \in A$, the set $\{\overline{u_0}(k) \mid k \in \mathbb{N}, [k] \in \mathrm{Repeat}^{(a)}\}$ is finite and we can list all its elements in finite time.*

Proof. The computability part utilizes the fact that we can restrict the set definition to $k < \mathrm{bound}^{(a)}$. □

Parameter set B_a, given $a \in A$ We call $k \in \mathbb{N}$ *referencing repeat* if it is productive and non-repeating for a and in $R_k = (x_k^{(t)}, y_k^{(t)})_{t=0,\ldots,q-1}$ there is t such that $x_k^{(t)}$ is repeating for a. The second parameter intends to establish a bound such that we will not have to deal with indices referencing repeat.

Knowledge For $b \in B_a$,

- an index $\mathrm{bound}^{(b)} \geq \mathrm{bound}^{(a)}$ and a lower bound $g^{(b)}$ for the number of repeat referencing indices k with $\mathrm{bound}^{(a)} \leq k < \mathrm{bound}^{(b)}$,
- the constraint $g^{(b)} \leq em\,\mathrm{period}$ where m is the size of the set in Lemma 1.8.

Assumption For $b \in B_a$, no $k \geq \mathrm{bound}^{(b)}$ is referencing repeat.

Ordering For $b, b' \in B_a$, we set $b < b'$ if $g^{(b)} > g^{(b')}$. Since $g^{(b)}$ is finitely bounded for fixed $a \in A$ and $b \in B_a$, this is trivially a well-ordering.

Dealing with Violation Given $k \geq \mathrm{bound}^{(b)}$ referencing repeat, we can define $b' \in A_b$ with $b' < b$ by setting $\mathrm{bound}^{(b')} := k+1$ and $g^{(b')} := g^{(b)} + 1$. If this is a violation of the size constraint, we have found $em\,\mathrm{period}+1$ repeat referencing $k \geq f$. By the pigeonhole principle, there must be $0 \leq t < t$ and $m\,\mathrm{period}+1$ representations $R_k = (x_k^{(t)}, y_k^{(t)})_t$ with $x_k^{(t)}$ defined and $[x_k^{(t)}]$ repeating for a. By Lemma 1.8 and Lemma 1.7, this constitutes a violation of $a \in A$.

Start The starting parameter $b_0 \in B_a$ has $\mathrm{bound}^{(b_0)} := \mathrm{bound}^{(a)}$ and $g^{(b_0)} := 0$.

Parameter set $C_{a,b}$, given $a \in A$ and $b \in B_a$ The third parameter set intends to establish an even higher bound of indices not even referencing values below the bound of the previous parameter.

Knowledge For $c \in C_{a,b}$,

- finite sets
$$W_0^{(c)}, \ldots, W_{e-1}^{(c)}, W_e^{(c)} \subseteq \{\mathrm{bound}^{(b)}, \mathrm{bound}^{(b)}+1, \ldots\}$$
with $W_e^{(c)} = \emptyset$ such that for $t = 0, \ldots, e-1$ and $w \in W_t^{(c)}$, we have $[w]$ productive and non-repeating for a and either $\overline{u_0}(w) < \mathrm{bound}^{(b)} + f$ or $\overline{u_0}(w) \geq \mathrm{bound}^{(b)} + f$ and $\overline{u_0}(w) + (b_{t+1} - f) \in W_{t+1}^{(c)}$,
- size constraints $|W_t^{(c)}| \leq \mathrm{period}(|W_{t+1}^{(c)}| + \mathrm{bound}^{(b)} + f)$,
- an abbreviation $\mathrm{bound}^{(c)} := \max(\mathrm{bound}^{(b)}, \max(W_0^{(c)}) + 1)$.

Assumption For $c \in C_{a,b}$, for all $k \geq \text{bound}^{(c)}$ productive and non-repeating for a, we have $R_k = (x_k^{(t)}, y_k^{(t)})_{t=0,\ldots,e-1}$ with $x_k^{(t)} \geq \text{bound}^{(b)}$ and $\overline{u_0}(k) = y_k^{(e-1)} + (b_e - f) \geq \text{bound}^{(b)}$.

Ordering For $c, c' \in C_{a,b}$, let $t = 0, \ldots, e-1$ be maximal such that $W_t^{(c)} \neq W_t^{(c')}$. We set $c' < c$ if such a t exists and $W_t^{(c)} \supsetneq W_t^{(c')}$. The proof that $(C_{a,b}, <)$ is a well-ordering goes as follows. Assume there is an infinite chain $c_0 > c_1 > \ldots$. For $n \in \mathbb{N}$, choose $v_n \in \{0, \ldots, e-1\}$ maximal such that $W_{v_n}^{(c_{n+1})} \neq W_{v_n}^{(c_n)}$, i.e. $W_{v_n}^{(c_{n+1})} \supseteq W_{v_n}^{(c_n)}$. Let $z \in \{0, \ldots, e-1\}$ be maximal such that $v_n = z$ for infinitely many $n \in \mathbb{N}$. Choose $m \in \mathbb{N}$ such that $v_n \leq z$ for $n \geq m$. It follows that $W_z^{(c_n)}$ grows arbitrarily large for $n \geq m$ while $W_{z+1}^{(c_n)}$ stays constant, a contradiction to the size constraint.

Dealing with Violation Given a violation, i.e. $k \geq \text{bound}^{(c)}$ productive and non-repeating for a with $R_k = (x_k^{(t)}, y_k^{(t)})_{t=0,\ldots,q-1}$ and $q \leq e$, we have $x_k^{(s)} < \text{bound}^{(b)}$ for some $s > 0$, or $q = e$ and $y_k^{(e-1)} + (b_e - f) < \text{bound}^{(b)}$. We define $c' \in C_{a,b}$ by adding $x_k^{(t)}$ to $W_t^{(c)}$, in the former case for $t < s$ and in the latter case for $t < e$. It is easy to see that $c' < c$ and c' is well-defined, except if some size constraint is violated. In this case, we can apply Lemma 1.7 to find a violation of $a \in A$.

Start The starting parameter $c_s \in C_{a,b}$ has $W_t^{(c_s)} = \emptyset$ for all t.

Core of the Algorithm, given $a \in A$, $b \in B_a$, and $c \in C_{a,b}$ For the remainder of the proof, we will implicitly assume that the assumptions from a, b, c are validated. The reader is invited to check that in each setting, a false assertion depending on erroneous assumptions from either a, b, or c yields constructive evidence for a violation of a, b, c, respectively, enabling a restart.

For each non-repeating $k \geq \text{bound}^{(b)}$, let $\hat{k}$ denote the set of non-repeating $k' \geq \text{bound}^{(b)}$ such that $[k] \sim^{(a)} [k']$ and $k' = k + \text{offset}^{(a)}_{[k],[k']}$. We call $\hat{k}$ an *argument class*. As far as our assumptions from a go, these are the only sets of arguments not smaller than $\text{bound}^{(b)}$ which $\overline{u_0}$ maps to the same value we know of. Let $K := \{\hat{k} \mid k \geq \text{bound}^{(b)}\}$ and $L := \{S \in K \mid S \geq \text{bound}^{(c)}\} \subseteq K$.

We use Theorem 1.3 to build an initial finite history

$$H_0 = \{(k, \overline{u_0}(k)) \mid \overline{u_0}(k) \neq \bot, \text{ and } k < f \text{ or } \hat{k} \notin L\},$$

extending it stepwise while traversing the argument classes in L, using $R_n \in \mathbb{N}$ to denote the current class.

Given H_n for some $n \in \mathbb{N}$, we choose $i = 0, \ldots, p-1$ minimal such that for $x \in [n+i]$ with $x \geq f$, $x \notin H_n$ minimal, x is productive and non-repeating, and computation at argument x has no unresolved dependencies in H_n. If no such i exists, $\overline{u_0}$ is non-productive after a certain index. Else, set $R_n := \hat{x}$, let $y = \overline{u_0}(R_n)$, and set $H_{n+1} = H_n \cup \{(x', y) \mid x' \in R_n\}$. We note that this way of extending the history is fair in the sense that every productive argument index will eventually be covered.

Given an argument class $S \in L$, we know $k \in S$ has representation $R_k = (x_k^{(t)}, y_k^{(t)})_{t=0,\ldots,e-1}$ with $x_k^{(t)} \geq \text{bound}^{(b)}$. Also, with $T := S - \text{period} \in K$, we have $x_k^{(t)} \in T$ and thus $y_0^{(k)} = y_0^{(k')}$ for $k, k' \in S$, meaning $(x_k^{(t)}, y_k^{(t)}) = (x_{k'}^{(t)}, y_{k'}^{(t)})$ for $t \geq 1$. Hence, we can uniquely associate with S a sequence of argument classes $U_S^{(0)}, U_S^{(1)}, \ldots, U_S^{(e-1)} \in K$ called *representation* of S such that $x_t^{(k)} \in U_S^{(k)}$ or $k \in S$ and all t.

Lemma 1.9. *For $t = 0, \ldots, e-1$, the function $f_t : L \to K$, $S \mapsto U_S^{(t)}$ is injective.*

Proof. For $S, S' \in K$ and $S, S' \geq \text{bound}^{(c)}$ with $S \neq S'$ and $U_S^{(t)} = U_{S'}^{(t)}$, we have $y_k^{(t)} = y_{k'}^{(t)}$ and hence $\overline{u_0}(k) = \overline{u_0}(k')$ for $k \in S, k' \in S'$, contradicting assumptions of a. □

At each point $n \in \mathbb{N}$ in the traversal, the yet to be consumed argument classes are given by

$$\text{Unused}_n := \{(t, S) \mid t = 0, \ldots, e-1, S \in K, S \subseteq H_n, S \neq f_t(R_m) \text{ for } m < n\}.$$

Lemma 1.10. *There is a constant $z \in \mathbb{N}$ such that $|\,\text{Unused}_n\,| = z$ for all $n \in \mathbb{N}$.*

Proof. By induction. For $n = 0$, we know $z := |\,\text{Unused}_0\,| = |K \setminus L| \in \mathbb{N}$. For general n, we have

$$\begin{aligned} |\,\text{Unused}_{n+1}\,| &= |\,\text{Unused}_n \setminus \{(t, f_t(R_n)) \mid t = 0, \ldots, e-1\} \\ &\qquad \cup \{(t, R_n) \mid t = 0, \ldots, e-1\}| \\ &= z - e + e = z \end{aligned}$$

by by the previous lemma and induction hypothesis. □

Given $a, b \in \mathbb{N}$ with $a \leq b$, we associate a linear system $L_{a,b}$ of equations with the traversal interval from a to b. Let $V := \text{Unused}_a \cup \{R_n \mid a \leq n < b\}$. For each argument class $S \in V$ and $k \in S$, we create variables X_k and Y_k. For each argument class $S = R_a, \ldots, R_{b-1}$ and $k \in S$, we create equations $Y_{x_k^{(t)}} - f + b_{t+1} = X_{x_k^{(t+1)}}$ for $t = -1, \ldots, e-1$, setting $x_k^{(e)} := \overline{u_0}(k)$ for convenience.

We know this system is solvable by setting $X_i = i$ and $Y_i = \overline{u_0}(i)$ for all variables X_i, Y_i. Our key interest is the dimension $d_{a,b}$ of the affine solution subspace.

Lemma 1.11. *For $a \leq b \leq c$, we have $d_{a,b} \geq d_{a,c}$ and $d_{a,b} \leq 2\,\text{period} \cdot z$.*

Proof. The first proposition follows inductively from the fact that even though $L_{a,b+1}$ contains additional variables, these variables appear in equations with the other side of the equation containing a variable from $L_{a,b}$, hence the additional variables being determined by a solution to $L_{a,b}$, i.e. $d_{a,b} \geq d_{a,b+1}$. The second proposition follows from $d_{a,b} \leq d_{a,a} \leq 2\,\text{period} \cdot z$ by Lemma 1.10. □

Definition 1.6. *For $a \leq b < a' \leq b'$ with $a \equiv a' \mod \text{period}$ and $b - a = b' - a'$, two systems $L_{a,b}$ and $L_{a',b'}$ are called* copies *(of each other) if $R_{a+i} = R_{a'+i} + q_i \cdot \text{period}$ for some $q_i \in \mathbb{N}$ and all $i < b - a$ and there is a variable-renaming isomorphism between $L_{a,b}$ and $L_{a',b'}$ such that for $i < b - a$, X_k and Y_k are renamed to X_{k+q_i} and Y_{k+q_i}, respectively, for $i < b - a$ and $k \in R_{a+i}$.*

For copies $L_{a,b}$ and $L_{a',b'}$, we necessarily have $d_{a,b} = d_{a',b'}$.

Lemma 1.12. *For $L_{a,b}$ and $L_{a',b'}$ copies with $d_{a,b} = d_{a,a'}$, we can determine the structure of $\overline{u}$.*

Proof. Since adding the equations for R_n with $b \leq n < a'$ does not decrease the solution space of $L_{a,b}$, they are valid in general for any solution of $L_{a,b}$ and hence, appropriately renamed, also for any solution of $L_{a',b'}$. But this means that the history becomes periodic after $H_{a'}$. Setting $p := \min(R_{a'}) - \min(R_a)$ and noting period $\mid p$, after finitely many values $\overline{u_0}$ is non-productive on all classes belonging to $\text{Nonprod}^{(a)}$ and those classes the history does not make any progress on between H_a and $H_{a'}$. On all other classes modulo p, we deduce $\overline{u_0}$ is linear. □

Lemma 1.13. *Given $l \geq 1$, there is $c \in \mathbb{N}$ such that given a sequence $a_0 \leq b_0 < \ldots < a_{c-1} \leq b_{c-1}$ with $a_{i+1} - a_i \leq l$ for $i = 0, \ldots, c-2$, there are $0 \leq r < s < c$ such that L_{a_r,b_r} and L_{a_s,b_s} are copies.*

Proof. A consequence of the pigeonhole principle. □

Definition 1.7. *The* sequence property *for dimension $d \in \mathbb{N}$ is the following proposition: there is $l \geq 1$ such that for all $c \in \mathbb{N}$ we can determine the structure of $\overline{u}$ or construct a sequence $a_0 \leq b_0 < \ldots < a_{c-1} \leq b_{c-1}$ with $a_{i+1} - a_i \leq l$ for $i = 0, \ldots, c-2$ and $d_{a_i,b_i} \leq d$ for $i = 0, \ldots, c-1$.*

Lemma 1.14. *The sequence property for dimension $d \in \mathbb{N}$ implies the sequence property for dimension $d-1$.*

Proof. This follows from Lemmata 1.12 and 1.13. □

By Lemma 1.11, the sequence property is fulfilled for dimension $2\,\mathrm{period}\cdot z$. Since the sequence property for dimension -1 is equivalent to determining the structure of $\overline{u}$, we can do the latter. □

1.6 Further Work

In the section on further work in our article [52], we claimed that is was possible to encode any recursive function as an indexing function occurring in a singleton system of four, and that productivity of unary systems was undecidable even for systems of size two. These claims were based on a complex encoding of a sufficiently large subset of Collatz-like functions as singleton unary equations. Unfortunately, the arithmetic involved turned out to be flawed, so we have to repudiate these claims.

Using the existing technical tools, we can at least show the following:

Proposition 1.2. *Given a unary system of size three defining a stream function f, it is undecidable whether f is productive. More specifically, the problem is Π^0_2-complete.*

Proof. By simple evaluation, the problem is seen to be in Π^0_2. Given a Collatz function g, we will construct a unary system of size three defining a stream function f such that f is productive iff g is reducing. By [40], we then deduce Π^0_2-hardness of productivity of f.

After having defined add as in the proof of Lemma 1.2, define

$$u(s) = \mathrm{head}(\mathrm{tail}^{e_0}(s)) :: \ldots :: \mathrm{head}(\mathrm{tail}^{e_{n\cdot n-1}}(s)) :: u(\mathrm{add}(s))$$

with $e_{n\cdot i+r} = n\cdot g(i) + i$ for $i, r = 0, \ldots, n-1$. By induction, we have

$$\overline{u}(n\cdot k + r) = n\cdot g(i) + i + \frac{k-i}{n}\cdot(n\cdot a_i) = n\cdot g(k) + i$$

for $r = 0, \ldots, n-1$ and $i < n$ such that $k \equiv i \mod n$. Now we can define

$$f(s) = \underbrace{\mathrm{head}(s) :: \mathrm{head}(s)}_{n \text{ times}} :: \mathrm{tail}^n(u(f(s))).$$

We note that

$$f(s)\,!\,n\cdot k + r \to^+ \begin{cases} s\,!\,0 & \text{if } k = 0, \\ n\cdot g(k) + r' & \text{else} \end{cases}$$

with some $r' \in \{0, \ldots, n-1\}$, meaning $\overline{f}(n\cdot k + r) = 0$ if g is reducing at k, and $\overline{f}(n\cdot k + r) = \bot$ else. □

Since productivity of unary singleton systems is decidable by Theorem 1.4, this leaves open the status of productivity for unary systems of size two.

Similarly, we suspect that our technical tools are capable of proving Theorem 1.2 while defining only five stream functions (instead of four as claimed originally). The minimum system size for unary definability thus still remains an open problem.

1.7 Related Work

Streams and corecursive functions on them are of course a well-studied model of coinductive behaviour, particularly in the setting of term rewriting [15, 59]. Note that what we call indexing function of a polymorphic stream function is a container morphism for streams in the terminology of Abott et al. [1].

Most proofs of undecidability and complexity results for stream equations, like the ones of Roşu [50] and Simonsen [53], use straightforward encodings of Turing machines: representing the infinite band of symbols as two streams, one each for the left and right side of the band relative to the head, with canonical rewrite rules examining the current symbol. This pattern matching dispatching mechanism is unavailable in our setting. Still, we have been able to recover all of these results even in the unary setting as direct corollaries of Theorem 1.2.

Endrullis et al. [14, 16, 17] strive to decompose rewriting into a stream layer and a data layer in such a way as to encapsulate just so much complexity into the data layer that the productivity of streams becomes decidable while still retaining usefulness of computation. Our work can be seen as a another extreme, eradicating the data layer and showing that polymorphic unary stream functions attain computational completeness. For example, our results imply that the lazy stream formats of Endrullis et al. [17] can actually be restricted to (general) unary stream functions with productivity still retaining Π^0_2-completeness (in the non-unary case, a hint of Proposition 1.1 can be found in their encoding of FRACTRAN-programs). We note that their notions of lazy stream specifications and data-oblivious analysis shares some points with our polymorphism restriction: choosing the unit type for the data type leaves no possibility of analyzing the input. We also note that the flat stream specifications, for which the authors develop an algorithm for semi-deciding productivity, present an exception: we allow general nested calls.

We stress that our encoding in Lemma 1.2 of generalized Collatz functions as indexing functions of unary polymorphic stream functions is fundamentally different from the encoding of Collatz functions or Fractran program iterations used by Endrullis et al. [17], the essential difference being the unavailability of zip in the unary setting. The dispatch mechanism of interleaving, enabling differing treatment of stream positions based on the residue of their indices, makes the implementation of Collatz-like constructs rather straightforward. Since we are lacking even such basic conditional control flow mechanisms, we have to resort to indirect constructions.

Note that the Collatz function $\widehat{\chi}_{\mathbf{A}}$ in Lemma 1.3 is special in that it is linear on each of its equivalence classes in the strict sense, i.e. with vanishing ordinate, corresponding to single multiplication with a fraction. Even though we do not make use of this fact in our developments, it shows the connection between IF-programs and the iteration steps of the FRACTRAN-programs of Conway [11], which are of equivalent expressive power. In fact, we could have used FRACTRAN-programs for the construction leading up to Theorem 1.2, but would then have lost the conceptual clarity of IF-program iteration syntax. In contrast to the proof [40] of undecidability of the generalized Collatz problem making use of FRACTRAN-programs [17], we view it as an unnecessary detour.

Chapter 2

Isomorphism of Finitary Inductive Types

2.1 Introduction

Given any two finitary inductive types, possibly parametric, a natural question to ask is whether and in what sense they are isomorphic. An intuitive example is provided by binary trees with two type parameters, one specifying the data at leafs and the other the data at nodes: [1]

```
data Tree (A B : Set) : Set where
  leaf : A → Tree A B
  node : B → Tree A B → Tree A B → Tree A B
```

An alternative but isomorphic presentation of such trees is as spine trees, unravelling the data structure through its left-most branch:

```
data Spine (A B : Set) : Set where
  nil  : Spine A B
  cons : B → SpineTreeA B → Spine A B → Spine A B

data SpineTree (A B : Set) : Set where
  spine : A → Spine A B → SpineTree A B
```

Note that this constitutes a *mutual* specification. We may equivalently transform into the following *nested* presentation:

```
data List (X : Set) : Set where
  nil  : List X
  cons : X → List X → List X

data ListTree (A B : Set) : Set where
  spine : A → List (B × ListTree A B) → ListTree A B
```

For brevity, when speaking about finitary inductive types, we will henceforth always mean parametric and possibly nested finitary inductive types. Note that it is folklore how to convert mutual finitary inductive types into nested ones.

[1] Because of already available typesetting infrastructure, we will use Agda syntax for presenting examples. This merely syntactic convention does not entail restricting ourselves to a particular object theory.

It is quite easy to write functions going back and forth between the different representations and show by induction that corresponding pairs of functions form inverses. This holds true in any number of settings, whether we are working in dependent type theory with a universe as our syntax misleadingly suggests, in complete partial orders like those employed by most papers reasoning about denotational semantics of Haskell programs, or in standard set theory.

At this point, we should clarify what exactly we mean by isomorphic. Certainly, the functions realizing the isomorphism should be independent of the specific model of finitary inductive types. But just as important, the proof that the functions constitute inverses should also be agnostic of the model. This last point will be the major source of contention in our development, for in the syntactic models like the term model we are not allowed to reason by semantic induction. Relatedly, colimits do not in general exist in such models, and hence finitary inductive types can not in general be constructed as the ω-colimit of a certain chain. We must look for ways to encode inductive arguments as universal properties of inductive types, in particular uniqueness of eliminators.

Our notion of model for finitary inductive types will be bicartesian-closed categories closed under formation of initial algebras for regular functors. Regular functors are, in essence, a synonym for nested parametric finitary inductive types. We consider the name a slight misnomer as it suggests a similarity with regular languages. In contrast, regular functors are closer to a proof-relevant version of context-free grammars with commuting terminal symbols. As observed by Altenkirch [4], isomorphism of parametric finitary inductive types in the set model corresponds to equivalence of such grammars.

This work makes progress on several open questions posed by Altenkirch [4]. It is shown that isomorphism of finitary inductive types in the set model is decidable for certain so-called *guarded* types, resulting in an algorithm computing a natural bijection if the answer is positive. In the bulk of the work, it is further shown that isomorphism of guarded finitary inductive types is decidable in the initial model, i.e. the syntactic category, and that the answer coincides with the one given for the set model. This makes the set model complete with respect to type isomorphism of guarded types. Again, since all our proofs are constructive, answering the question in the initial model automatically computes a pair of λ-terms witnessing an isomorphism (and the chain of term conversions proving they form inverses) for any pair of isomorphic guarded types.

We show how to decompose a finitary inductive type into so-called guarded and indefinite parts. Semantically, when viewing finitary inductive types as power series in their type variables, a guarded type has only finite coefficients while an indefinite type has only zero and infinite coefficients. Thus, we expect guarded types to live in a ring-like algebraic structure enjoying additive cancellability. However, the decomposition is purely syntactic. Decidability of indefinite regular types turns out to reduce to certain well-known facts in the theory of commutative grammars. We discuss the problems we encountered in trying to combine the arguments for the guarded and indefinite parts, with difficulties arising from only the indefinite part of the decomposition being uniquely determined. However, it can be argued that deciding isomorphism is of prime interest mostly for guarded types.

The outline for this chapter is as follows:

- First, we will briefly recall the notions of bicartesian-closed categories and initial algebras. We will review several useful tools from the literature, most prominently the μ-calculus [5] and derived results such as the abstraction theorem showing stability of parametric initial algebras under type variable substitution.

- We will then develop our tools for dealing with regular functors, in particular what we call the *biased derivative*. With this, we will establish a decomposition of regular functors into *guarded* and *indefinite* parts. This decomposition is only partly well-behaved in that the guarded part is semantically characterizable only modulo selective masking by the indefinite part. Up until this point, our development is agnostic of the specific model.

- Using the framework of containers [1], we will then solve the problem for guarded types in

the specific model of sets and functions. This step will make apparent the connection to commutative algebra, in particular algebraicity of power series. We remark on the decidability of isomorphism for indefinite types, and discuss problems enountered in combining arguments for both parts.

- As an interlude to the introduction of containers, we will analyse anti-derivates of certain quotient containers, negatively answering a question by Altenkirch about integrability of quotient containers. However, this work turns out to overlap with a corresponding previous result for combinatorial species.

- Next, we will review a series of technical tools needed for the term setting. This includes original work on traversable functors [20, 30], showing regular functors traversable in any model. We review the syntactic calculus useful for speaking about constructions internal to an arbitrary model and derive tools such as an internal approximation to structural induction over internal Boolean logic, a generalized internal equality predicate, internal injection and extraction of data and their connection to the concept of shapely functors [31].

- As an interlude, to become familiar with the tools reviewed and introduced in the preceding section, we will prove the following classification result: any regular constant is isomorphic in any model to either the internal natural numbers or a finite sum of unit types.

- Via the concept of sound and complete listings, we derive a description of guarded regular types in terms of internal power series. Stopping short of explicitely re-developing the basics of commutative algebra, we sketch how to use the internal framework of operations on polynomials and power series to handle the internal minimal polynomials for power series associated to guarded regular types.

Certain subsections allow to be read in a stand-alone fashion, most notably the interlude on anti-derivatives of quotient containers and the (we believe original) structural derivation of traversabilty for regular functors. These subsections were planned as parts of separate chapters, but are included here for completeness of presentation.

2.2 Preliminaries

2.2.1 The Setting

Bicartesian-closed categories A *bicartesian-closed category* $\mathcal{C}$ is a cartesian-closed category that in addition has finite coproducts. The existence of exponentials as right adjoints to binary products with a fixed factor makes products distribute over coproducts. The bicartesian structure of $\mathcal{C}$ is succinctly summarized by the adjoint triple

$$\sum_N \dashv \Delta \dashv \prod_N$$

for any finite set N where $\Delta : \mathcal{C} \to \mathcal{C}^N$ is the diagonal functor and $\sum_N, \prod_N : \mathcal{C}^N \to \mathcal{C}$ denote the N-ary coproduct and product, respectively. Closedness of $\mathcal{C}$ means that the product $\cdot \times B$ has a right adjoint for any object $B : \mathcal{C}$:

$$\cdot \times B \dashv (\cdot)^B.$$

Here, note that A^B denotes the internal hom-object for $A, B : \mathcal{C}$. It is covariantly functorial in A and contravariantly functorial in B.

Initial algebras Given an endofunctor $F : \mathcal{C} \to \mathcal{C}$ on a category $\mathcal{C}$, the category $\mathrm{Alg}_{\mathcal{C}}(F)$ of *algebras* over F has as objects pairs (A, f) where $A : \mathcal{C}$ and $f : F(A) \to A$. A morphism between (A, f) and (B, g) is given by $h : A \to B$ such that $h \circ f = g \circ F(h)$. An initial object in $\mathrm{Alg}_{\mathcal{C}}(F)$ is called *initial algebra* and usually denoted $\mu_{\mathcal{C}} F = (\mu_{\mathcal{C}} F, \textsc{init}_F)$ if existing (note the overloaded use of notation for the carrier of the initial algebra). It is common to write $\mu X.\, F[X]$ for $\mu(\lambda X.F[X])$. Given an algebra (A, f) over F there is a unique algebra morphism $\mathrm{elim}_A^F : \mu_{\mathcal{C}} F \to A$ from the initial algebra to (A, f). The operator $\mu_{\mathcal{C}}$ as denoting the carrier can be given the structure of a functor from the full subcategory of endofunctors on $\mathcal{C}$ having initial algebras to $\mathcal{C}$. As usual, subscripts and superscripts will be omitted if the extra information they provide is redundant.

Parametric initial algebras Given categories $\mathcal{C}, \mathcal{D}$, let $F : \mathcal{D} \times \mathcal{C} \to \mathcal{C}$ be a functor such that $F(X, \cdot)$ has an initial algebra for any object $X : \mathcal{D}$. There are two obvious ways of defining the *parametric initial algebra* with carrier $G : \mathcal{D} \to \mathcal{C}$ of F. Pointwise, we might say $G(X) = \mu Y.\, F(X, Y)$ for on object $X : \mathcal{D}$ and observe that this coherently extends to a functor. In fact, the individual initial algebra maps

$$\textsc{init}_{F(X,\cdot)} : F(X, G(X)) \to G(X)$$

form a natural transformation from $\overline{F}(G)$ to G where $\overline{F} : \mathcal{C}^{\mathcal{D}} \to \mathcal{C}^{\mathcal{D}}$ is defined by $\overline{F}(H) = F \circ \langle \mathrm{id}, H \rangle$. On the other hand, we might directly define $G = \mu_{\mathcal{C}^{\mathcal{D}}} \overline{F}$. It is the statement of the abstraction theorem [5] that both definitions are equivalent, and that in fact the uniform initial algebra $\mu_{\mathcal{C}^{\mathcal{D}}} \overline{F}$ exists whenever the initial algebras exist pointwise. From now on, we will say that a functor $F : \mathcal{D} \times \mathcal{C} \to \mathcal{C}$ has a parametric initial algebra whenever this condition is fulfilled. One particular implication is that parametric initial algebras are stable under change of context, i.e. substituting their type variables. That is, given $K : \mathcal{D}' \to \mathcal{D}$ the parametric initial algebra of $F \circ \langle K, \mathrm{id} \rangle$ is obtained by precomposing the parametric initial algebra of F with K. The whole story has a nice retelling in the form of the opfibration corresponding to $\mathrm{Alg}_{\mathcal{C}} : (\mathcal{C}^{\mathcal{C}})^{\mathrm{op}} \to \mathrm{Cat}$ via the Grothendieck construction, which seems to be folklore knowledge.

Regular functors Before we can say what constitutes a model for finitary inductive types, let us first recall the notion of regular functors [20, 45].

Definition 2.1. *The (syntactic codes for)* regular functors *of arity N are an inductive family defined over finite parameter index sets N, given by (constructors for)*

- *variable selectors π_n^N for $n : N$,*
- *the terminal functor 1 and products $G \times H$ of regular functors G, H of arity N,*
- *the initial functor 0 and coproducts $G + H$ of regular functors G, H of arity N,*
- *the parametric initial algebra $\mu Y.\, G(\cdot, Y)$ of regular functors G of arity $N + 1$ (up to isomorphism).*

Note that despite its name, the notion of regular functors is agnostic to any choice of category, being purely a definition of codes. However, we will be rather lax with the distinction between inductively defined syntactic codes for functors and their interpretation, calling a functor $\mathcal{C}^N \to \mathcal{C}$ regular if it is equivalent to the interpretation in the model $\mathcal{C}$ of an element of arity N of the above inductive type of codes for regular functors. [2] Similarly, we will leave isomorphisms of parameter index sets such as decompositions $M = N + 1$ as needed in the final clause implicit wherever reasonably possible, also making implicit use of isomorphisms such as $\mathcal{C}^{N+1} = \mathcal{C}^N \times \mathcal{C}$ when writing down equations for functors applied to parameters.

[2] The same convention will apply for further such definitions of syntactic codes with implicitly defined interpretations as functors.

Models A *model* now is a bicartesian-closed category $\mathcal{C}$ in which the above codes for regular functors can be interpreted, i.e. that inductively has all regular functors. For variable selection functors as well as finite products and coproducts of regular functors this is immediate from the bicartesian structure of $\mathcal{C}$. For parametric initial algebras, whenever we are given (an interpretation of) a regular functor $G : \mathcal{C}^{N+1} \to \mathcal{C}$, then $\mu Y.\, G(X, Y)$ must exist for every $X : \mathcal{C}$, with $\mu\overline{G} : \mathcal{C}^N \to \mathcal{C}$ being the interpretation of the code for the parametric initial algebra of G.

With the above conventions in place, we may observe that regular functors are closed under composition in any model. Note that this uses preservation of parametric initial algebras under type variable substitution. This closure is reflected by a composition operation on the level of codes.

2.2.2 The μ-calculus

The rules of the μ-calculus, including the diagonal, squaring, and rolling rules as well as the powerful μ-fusion rule as presented below, yield a way of manipulating least fixed-point equations purely on the type level. Not having to go down to the term level to state and prove isomorphisms provides for an elegant high-level presentation based on type-level rewriting. The calculus itself is rather flexible, with many possible choices for proceeding in any given situation.

The basics A comprehensive reference for the following five lemmata and the various orders in which they can be proved may be found in [5]. For the purpose of this subsection, let us work over arbitrary categories, not necessarily bicartesian-closed ones.

Lemma 2.1 (Fusion)**.** *Let categories $\mathcal{C}$ and $\mathcal{D}$ be connected via an adjunction $L \dashv R$ with functors $L : \mathcal{C} \to \mathcal{D}$ and $R : \mathcal{D} \to \mathcal{C}$. Consider endofunctors F and G on $\mathcal{C}$ and $\mathcal{D}$, respectively. If the diagram*

$$\begin{array}{ccc} \mathcal{C} & \xrightarrow{\;L\;} & \mathcal{D} \\ \downarrow{\scriptstyle F} & & \downarrow{\scriptstyle G} \\ \mathcal{C} & \xrightarrow{\;L\;} & \mathcal{D} \end{array}$$

commutes up to an equivalence $\alpha : L \circ F \simeq G \circ L$ of functors, then the adjunction $L \dashv R$ lifts to an adjunction $L' \dashv R'$ between $\mathrm{Alg}(F)$ and $\mathrm{Alg}(G)$ given by

$$\begin{aligned} L'(X, f) &= (LX, \alpha \circ L(f)), \\ R'(Y, g) &= \left(LY, \alpha^{-1} \circ R(g)\right). \end{aligned}$$

In particular, preservation of initial objects by left adjoints implies preservation $L(\mu F) = \mu G$ of initial algebras.

An independent account of this under-appreciated tool, generalized to the 2-categorical framework of inserters and with a more conceptual proof, can be found in [25].

Lemma 2.2 (Diagonal rule)**.** *Fix a category $\mathcal{C}$ and a functor $F : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$. Then,*

$$\mu Z.\, F(Z, Z) = \mu X.\, \mu Y.\, F(X, Y)$$

where the initial algebras on one side of the equation exist whenever the ones one the other side do. The witnessing isomorphism is natural in the functor F.

Lemma 2.3 (Binary mutual recursion)**.** *Fix categories $\mathcal{C}$ and $\mathcal{D}$ and functors $\langle U, V\rangle : \mathcal{C} \times \mathcal{D} \to \mathcal{C} \times \mathcal{D}$. Then,*

$$\begin{aligned} \mu\,\langle U, V\rangle = (&\mu X.\, U(X, \quad \mu Y.\, V(X, Y)), \\ &\mu Y.\, V(\mu X.\, U(X, Y), Y \quad)) \end{aligned}$$

whenever the initial algebras on the right-hand side exist. The witnessing isomorphism is natural in the functors U and V.

Lemma 2.4 (Rolling rule)**.** *Fix categories $\mathcal{C}$ and $\mathcal{D}$ and functors $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$. The functor from* $\mathrm{Alg}(G \circ F)$ *to* $\mathrm{Alg}(F \circ G)$ *induced by application of F creates initial objects. In other words, if either one of $\mu(F \circ G)$ and $\mu(G \circ F)$ exists, then so does the other, and*

$$F(\mu(G \circ F)) = \mu(F \circ G).$$

Lemma 2.5 (Squaring rule)**.** *Fix a category $\mathcal{C}$ and an endofunctor $F : \mathcal{C} \to \mathcal{C}$. The canonical functor from* $\mathrm{Alg}(F)$ *to* $\mathrm{Alg}(F \circ F)$ *reflects initial objects. In other words, if $\mu(F \circ F)$ exists, then so does μF, fulfilling*

$$\mu F = \mu(F \circ F)$$

and $\mathrm{INIT}_{F \circ F} = \mathrm{INIT}_F \circ F\,(\mathrm{INIT}_F)$ *via this isomorphism.*

The squaring rule has an obvious generalization to arbitrary powers, though we shall not need it here.

Easy consequences The simplest interesting parametric inductive type is the *list functor*

$$\begin{aligned} \mathrm{List} &: \mathcal{C} \to \mathcal{C}, \\ \mathrm{List}(X) &= \mu Y.\, 1 + X \times Y. \end{aligned}$$

As an instructing example for the power of fusion, let us demonstrate the following elementary lemma:

Lemma 2.6. *We have an isomorphism*

$$\mu Y.\, A + X \times Y = A \times \mathrm{List}(X)$$

natural in $A, X : \mathcal{C}$.

Proof. Commutativity of the diagram

$$\begin{array}{ccc} \mathcal{C} & \xrightarrow{A\times\cdot} & \mathcal{C} \\ {\scriptstyle Y\mapsto 1+X\times Y}\downarrow & & \downarrow{\scriptstyle Y\mapsto A+X\times Y} \\ \mathcal{C} & \xrightarrow{A\times\cdot} & \mathcal{C} \end{array}$$

is a triviality. However, applying fusion in this seemingly trivial context allows us to derive $A \times (\mu Y.\, 1 + X \times Y) = \mu Y.\, A + X \times Y$. For this, we only have to note that multiplication with A is left adjoint to exponentiation with A. □

The rules of the μ-calculus have mostly canonical parametric generalizations, in case of Lemma 2.2 and Lemma 2.3 a consequence of naturality of the postulated isomorphisms in the given functors. For the case of fusion, let us explicitly develop the parametric version.

Lemma 2.7 (Parametric Fusion)**.** *Fix categories Γ and Δ to serve as* contexts *and a functor $E : \Gamma \to \Delta$ to serve as* context morphism*. Let categories $\mathcal{C}$ and $\mathcal{D}$ be given that are connected via an adjunction $L \dashv R$ with functors $L : \mathcal{C} \to \mathcal{D}$ and $R : \mathcal{D} \to \mathcal{C}$. Consider functors $F : \Gamma \times \mathcal{C} \to \mathcal{C}$ and $G : \Delta \times \mathcal{D} \to \mathcal{D}$ to serve as* relative endofunctors *on $\mathcal{C}$ and $\mathcal{D}$ with respect to the contexts Γ and Δ, respectively. Assume that F and G have parametric initial algebras. Commutativity of the diagram*

$$\begin{array}{ccc} \Gamma \times \mathcal{C} & \xrightarrow{E\times L} & \Delta \times \mathcal{D} \\ {\scriptstyle F}\downarrow & & \downarrow{\scriptstyle G} \\ \mathcal{C} & \xrightarrow{L} & \mathcal{D} \end{array}$$

implies commutativity of the diagram

$$\begin{array}{ccc} \Gamma & \xrightarrow{E} & \Delta \\ {\scriptstyle \mu Y.\, F(\cdot,Y)}\downarrow & & \downarrow{\scriptstyle \mu Y.\, G(\cdot,Y)} \\ \mathcal{C} & \xrightarrow{D} & \mathcal{D} \end{array}$$

Proof. Let us rewrite the given diagram in higher-order style indexed over Γ:

$$\begin{array}{ccc} \mathcal{C}^\Gamma & \xrightarrow{L^\Gamma} & \mathcal{D}^\Gamma \\ {\scriptstyle \overline{F}}\downarrow & & \downarrow{\scriptstyle \overline{G \circ \langle E, \mathrm{id}\rangle}} \\ \mathcal{C}^\Gamma & \xrightarrow{L^\Gamma} & \mathcal{D}^\Gamma \end{array}$$

Certainly, the adjunction $L \dashv R$ lifts to an adjunction $L^\Gamma \dashv R^\Gamma$. We thus may apply Lemma 2.1 in this diagram and derive

$$L^\Gamma(\mu\overline{F}) = \mu\left(\overline{G \circ \langle E, \mathrm{id}\rangle}\right).$$

Recalling that

$$\mu\left(\overline{G \circ \langle E, \mathrm{id}\rangle}\right) = \mu\overline{G} \circ E,$$

we arrive at commutativity of the desired diagram. □

The mutual recursion theorem has a general n-ary statement following by iterated application of the binary case Lemma 2.3. It is, however, less convenient to state explicitly:

Corollary 2.1 (Mutual recursion)**.** *Fix a finite parameter index set N and categories $\mathcal{C}_n$ for $n : N$. Consider functors $U = \langle U_n \rangle_{n:N} : \prod_N \mathcal{C} \to \prod_N \mathcal{C}$. Then, we have an isomorphism $\mu U = \alpha^\emptyset$ where*

$$\alpha^I = \left\langle \alpha_n^I \right\rangle_{n:N} : \prod_{i:I} \mathcal{C}_i \to \prod_{n:N} \mathcal{C}_n,$$

$$\alpha_n^I(X) = \begin{cases} X_n & \text{if } n \in I, \\ \mu X_n.\, U_n\left(\alpha^{I \sqcup \{n\}}\left(X_i\right)_{i:I \sqcup \{n\}}\right) & \text{else} \end{cases}$$

is defined recursively over the partial order of subsets I of N, assuming that all initial algebras in this definition exist. The witnessing isomorphism is natural in the functor U.

Again, naturality of the mutual recursion isomorphism in the given functors in particular means that we may use it in contextual situations where we deal with parameterized endofunctors.

A general strategy for applying the basic rules of the μ-calculus seems to consist of first applying the diagonal rule any number of times to isolate different occurrences of fixed-point variables, then applying the squaring, rolling, or mutual recursion rules to manipulate the different occurrences separately. At the end, the diagonal rule is used in reverse to merge the artificially created nested fixed points back together. An instructive example is provided by the following exercise:

Corollary 2.2 (Partial squaring)**.** *Fix a category $\mathcal{C}$ and a functor $F : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$. We have an isomorphism*

$$\mu Z.\, F(Z, Z) = \mu Z.\, F(Z, F(Z, Z))$$

if either initial algebra exists, natural in the functor F.

Proof. Let us calculate

$$\begin{aligned}\mu Z.\, F(Z,Z) &= \mu X.\, \mu Y.\, F(X,Y) \\ &= \mu X.\, \mu Y.\, F(X, F(X,Y)) \\ &= \mu Z.\, F(Z, F(Z,Z)),\end{aligned}$$

where we have applied Lemma 2.2, Lemma 2.5, and finally Lemma 2.2 in reverse. □

2.3 Decomposition into Guarded and Unguarded Parts

2.3.1 Guarded and Shielded Functors

Fix a finite set I for the following two mutual definitions of subtypes of the regular functors. Our goal is to establish a syntactic criterion for finitary inductive types that have power series in variables from I with only finite coefficients. [3] The motivation is to make guarded types behave well with respect to additive inverses, later enabling powerful algebraic machinery. [4] For the initial algebra case, we require as a mutual definition the helper concept of shielded functors that in addition to being guarded are (i.e have power series) divisible by variables from a given subset of I.

To unlock the full algebraic prowess of power series, we would need the coefficients of $C(F)$ to live in some well behaved ring-like algebraic object like the integers. This is precisely the motivation for the definition of guarded functors.

Definition 2.2. *The (syntactic codes) for functors of arity N* guarded over I *are an inductive family defined over finite parameter index sets $N \supseteq I$, given by (constructors for)*

- *variable selectors π_n^N for $n : N$,*
- *the terminal functor 1 and products $G \times H$ of functors G, H of arity N guarded over I,*
- *the initial functor 0 and coproducts $G + H$ of functors G, H of arity N guarded over I,*
- *the parametric initial algebras $\mu Y.\, G(\cdot, Y)$ of a functor G of arity $N + 1$ (up to isomorphism) shielded over $I \subseteq N \hookrightarrow N + 1$.*

Definition 2.3. *Fix finite parameter index sets J, N such that $J \subseteq I \subseteq N$. Given a functor G_j of arity N guarded over I for every $i : J$, then the finite coproduct*

$$\left\langle \pi_j^N \right\rangle_{j:J} \bullet \left\langle G_j \right\rangle_{j:J} = \sum_{j:J} \pi_j^N \times G_j$$

of arity N is called J-shielded over I. [5] *If $J = I$, we simply say it is* shielded over I.

The parameter I in the above definitions is also known as the *(guardedness) variable cover*. Calling a functor of arity N (globally) guarded or shielded is shorthand for saying it is guarded or shielded over N, respectively.

Being shielded is stronger property than that of being guarded. Guardedness is a hereditary property of wellformedness concerning all appearances of the μ-operator inside a regular functor, while shieldedness adds a global shape requirement on top. Algebraically, a functor of arity N is shielded over $I \subseteq N$ if it lies in the semiring ideal of guarded functors generated by the variable selectors for I. With this intuition, it is easy to see that such functor are closed under finite coproducts and multiplication with functors of arity N guarded over I.

Clearly, the notions of guardedness and shieldedness of functors are monotonous over the variable cover: given $I_1 \subseteq I_2 \subseteq N$, any functor of arity N guarded or shielded over I_1 will also be guarded or shielded over I_2, respectively.

As a basic example, the list functor is a guarded functor, but not in the form $\mathrm{List}(X) = \mu Y.\, 1 + X \times Y$. Applying the rolling rule, we derive the alternative presentation

$$\mathrm{List}(X) = 1 + (\mu Z.\, X \times (Z + 1))\,.$$

Here, the functor $G : \mathcal{C}^2 \to \mathcal{C}$ defined by $G(X, Z) = X \times (Z + 1)$ is shielded over X. As can be seen here, we allow ourselves the use of basic syntactic sugar for talking about parameter indices. In this instance, being shielded over X is synonymous with being shielded over $\{0\} \subseteq [2]$.

[3] Here, finite coefficients means coefficients that are polynomials — instead of power series — with finite coefficients in the remaining type variables.

[4] This is quite direct in the case of the set model. Finding replacements emulating this reasoning in the initial model will be the primary concern of the later parts.

[5] The bullet operation denotes the scalar product.

Compositionality We warn that, in contrast to regular functors, guarded functors are not closed under composition. This can already be seen in the case of $\mathrm{List} \circ \mathrm{List} : \mathcal{C} \to \mathcal{C}$, as will later be proved rigorously.

However, if we also consider shieldedness, we get the desired compositionality. If $G : \mathcal{C}^N \to \mathcal{C}$ is guarded over $J \subseteq N$ and we have $F_n : \mathcal{C}^M \to \mathcal{C}$ guarded over $I \subseteq M$ for $n : N$ such that F_j is in addition shielded over I for $j : J$, then $G \circ \langle F_n \rangle_{n:N} : \mathcal{C}^M \to \mathcal{C}$ will be guarded over I as well. If in addition G was T-shielded with $T \subseteq J$ to begin with and F_t was shielded over $S \subseteq I$ for $t : T$, then $G \circ \langle F_n \rangle_{n:N}$ will be S-shielded as well. This compositionality can be made precise by defining a composition operation on the level of codes and showing that it corresponds to functor composition semantically.

It goes without saying that for any element n of a finite set N, the variable selector $\pi_n^N : \mathcal{C}^N \to \mathcal{C}$ is $\{n\}$-shielded.

Given a function $G : \mathcal{C}^{N+1} \to \mathcal{C}$ shielded over $I \subseteq N$, note that $F : \mathcal{C}^N \to \mathcal{C}$ defined by $F(X) = \mu Y.\, G(X, Y)$ is not just guarded over I, but also shielded over I itself. This can be seen by unfolding $F = G \circ \langle \mathrm{id}, F \rangle$ and the above remarks on compositionality.

Weakening the requirements The definition of guardedness we gave is in fact stricter than necessary. Before we can elaborate on that, let us see how guardedness interacts with mutual initial algebras.

Lemma 2.8. *Fix finite sets $I \subseteq M$ and N and a functor $K : \mathcal{C}^{M+N} \to \mathcal{C}^N$ such that the component functors $K_n : \mathcal{C}^{M+N} \to \mathcal{C}$ are shielded over $I \sqcup \emptyset \subseteq M + N$ for $n : N$. Then the component functors of $L = \mu Y.\, K(\cdot, Y) : \mathcal{C}^M \to \mathcal{C}^N$ are shielded over I.*

Proof. Let us first deal with the case $N = [2]$. Exploiting symmetry to focus on the first component functor $L_1 = \pi_1 \circ L : \mathcal{C}^M \to \mathcal{C}$, we may write

$$L_1(X) = \mu Y_1.\, K_1(X, (Y_1, \mu Y_2.\, K_2(X, Y_1, Y_2)))$$

by Lemma 2.3. Traversing the syntactical structure of the right-hand side, shieldedness of the invocation $K_i(X, Y_1, Y_2)$ for $i < 2$ in the variables $X|_I$ and previous remarks on the compositionality of guardedness validates this expression as a well-formed functor guarded and furthermore shielded over I.

The general case of a finite parameter index set N is proved by iterated application of the binary case, analogous to how the general finitary mutual recursion theorem follows from its binary version. □

The following lemma explains why the definition of shieldedness involved two separate parameter subsets. In essence, it claims that the requirement of G being shielded over I in the last clause of the definition of guardedness can be weakened to a variable cover that includes the variable introduced by the μ-operator.

Lemma 2.9. *Given finite sets $I \subseteq N$ and a functor $G : \mathcal{C}^{N+1} \to \mathcal{C}$ that is I-shielded over $I \sqcup \{\bullet\}$, then $F : \mathcal{C}^N \to \mathcal{C}$ defined by $F(X) = \mu Y.\, G(X, Y)$ is shielded over I.*

Proof. Traversing the syntax tree of G as a guarded functor top-down, we can find a collection of functors $H_a : \mathcal{C}^{N+1+1} \to \mathcal{C}$ shielded over $I \sqcup \{\bullet\} \sqcup \emptyset$ for $a : A$ such that

$$G(X, Y) = K_X\left[\mu Z_a.\, H_a(X, Y, Z_a)\right]_{a:A}$$

where K denotes a sequence of constructor applications for the inductive type of guarded functors that do not involve parametric initial algebras with holes indexed by A. Let us regard K as a polynomial functor of signature $\mathcal{C}^N \times \mathcal{C}^A \to \mathcal{C}$ that is I-shielded over $\emptyset$.

Fix $X : \mathcal{C}^N$ for the current paragraph. Consider the functor $U_X : \mathcal{C}^{1+A} \times C^{1+A} \to \mathcal{C}^{1+A}$ defined by

$$U_X\left(\left(Y^{(1)}, Z^{(1)}\right), \left(Y^{(2)}, Z^{(2)}\right)\right) = \left(K_X\left(Z^{(1)}\right), \left(H_a\left(X, Y^{(2)}, Z_a^{(1)}\right)\right)_{a:A}\right).$$

Noting that

$$(U_X \circ \Delta)(Y, Z) = (K_X(Z), (H_a\,(X, Y, Z_a))_{a:A})\,,$$

we identify $F(X)$ as the first component of $\mu S.\, U(S, S)$ by mutual recursion Corollary 2.1 in arity $1 + A$. Note that

$$\begin{aligned}\mu S.\, U_X(S, S) &= \mu S.\, U_X(S, U_X(S, S)) \\ &= \mu(Y, Z).\ \big(K_X(Z), (H_a(X, K_X(Z), Z))_{a:A}\big) \\ &= \mu(Y, Z).\ \big(K_X(Z), W_X(Z)\big)\end{aligned}$$

by Corollary 2.2, where we introduced the abbreviation $W_X : \mathcal{C}^A \to \mathcal{C}^A$ given by

$$W_X(Z) = (H_a(X, K_X(Z), Z_a))_{a:A}\,.$$

Applying binary recursion Lemma 2.3 in reverse, we derive $F(X) = K_X(\mu W_X)$.

Note that all constructions of the previous paragraph were natural in X. Recall that $H_a(X, Y, Z_a)$ is shielded over the formal parameters $X|_I$ and Y, and that K_X is shielded over I. By compositionality of shieldedness, it follows that the component functors of W are shielded over I as well. Lemma 2.8 then implies that the components of $\lambda X.\, \mu W_X : \mathcal{C}^N \to \mathcal{C}^A$ are shielded over I. Since the constructor sequence K did not involve any parametric initial algebra steps, the same holds true for F. □

2.3.2 The Biased Derivative

The tools developed in this section will serve as a kind of Swiss army knife to us. While the technical details of the lemmata here may make reading them not particularly pleasant, the result presented in the form of corollaries later means the trouble is well worth it.

Definition 2.4. *Fix a regular functor $F : \mathcal{C}^{M+N} \to \mathcal{C}$.* [6] *The* biased derivative $\underline{F} : \mathcal{C}^{M+3N} \to \mathcal{C}^3$ *of F with respect to its last N parameters, also written as*

$$\underline{F} = \big\langle F^L, F^M, F^R \big\rangle$$

with $F^L, F^M, F^R : \mathcal{C}^{M+3N} \to \mathcal{C}$, is defined inductively as follows.

- *If $F(X, Y) = X_m$ with $m : M$, then $F^L(X, Y) = F^R(X, Y) = X_m$ and $F^M = 0$.*
- *If $F(X, Y) = Y_n$ with $n : N$, then $\underline{F}(X, Y) = Y_n$.*
- *Taking the biased derivative distributes over finite sums. In detail:*
  - *If $F = 0$, then $\underline{F} = 0$.*
  - *If $F = G + H$ for regular functors $G, H : \mathcal{C}^{M+N} \to \mathcal{C}$, then $\underline{F} = \underline{G} + \underline{H}$.*
- *In the case of finite products, the left and right parts distribute. The middle part selects one factor, akin to the action of the derivative operator on products, and annotates all factors according to their relative position.*
  - *If $F = 1$, then $F^L = F^R = 1$ and $F^M = 0$.*

[6] Here, the first factor of the domain of F is treated as an opaque outer context. An alternative to try to improve the elegance of the presentation would be to entirely disregard the outer context and instead parameterize the definition of regular functors over some opaque base set of constants. In a situation where one would like to take e.g. the biased derivative of $F : \mathcal{E}^M \times \mathcal{E}^N \to \mathcal{E}$ with respect to its last N parameters, one may set $\mathcal{C} = \mathcal{E}^{\mathcal{E}^M}$ and treat the projections $\pi^M_m : \mathcal{E}^M \to \mathcal{E}$ with $m : M$ as the base set for regularity over $\mathcal{C}$.

– *If $F = G \times H$ with regular functors $G, H : \mathcal{C}^{M+N} \to \mathcal{C}$, then*

$$\begin{aligned} F^L &= G^L \times H^L, \\ F^M &= G^M \times H^R + G^L \times H^M, \\ F^R &= G^R \times H^R. \end{aligned}$$

Note that, in the general case of $F = G_0 \times \ldots \times G_{k-1}$ for regular functors $G_0, \ldots, G_{k-1} : \mathcal{C}^{M+N} \to \mathcal{C}$, the middle part may be written as

$$F^M = \sum_{i<k} G_0^L \times \ldots \times G_{i-1}^L \times G_i^M \times G_{i+1}^R \times \ldots \times G_{k-1}^R.$$

- *If $F(X, Y) = \mu Z.\, G(X, Y, Z)$ with a regular functor $G : \mathcal{C}^{M+N+1} \to \mathcal{C}$, then*

$$\underline{F}(X, Y) = \mu Z.\, \underline{G}(X, Y, Z).$$

Here, the biased derivative of G is understood to be taken with respect to its last $N + 1$ parameters.

It will always be clear from the surrounding context with respect to which variables we will be taking the biased derivative, most often in the form of an explicit splitting of the parameter index set into a binary coproduct. We will thus omit this intuitively accessible information.

Lemma 2.10. *The biased derivative is invariant under variable renamings.*

In detail, consider a morphism $u : M_1 \to M_2$ and an isomorphism $v : N_1 \to N_2$ between finite parameters index sets. We have induced variable renamings

$$\begin{aligned} \mathcal{C}^{u+\ v} &: \mathcal{C}^{M_2+\ N_2} \to \mathcal{C}^{M_1+\ N_1}, \\ \mathcal{C}^{u+3v} &: \mathcal{C}^{M_2+3N_2} \to \mathcal{C}^{M_1+3N_1}. \end{aligned}$$

Let $F : \mathcal{C}^{M_1+N_1} \to \mathcal{C}$ be a regular functor. Then,

$$\underline{F \circ \mathcal{C}^{u+v}} = \underline{F} \circ \mathcal{C}^{u+3v}.$$

Proof. A boring proof by structural induction on F, noting that the definition of the biased derivative $\underline{F}$ follows the general form of a definition invariant under variable namings. □

As is already evident from its definition, the interesting details of the biased derivative $\underline{F}$ of a regular functor $F : \mathcal{C}^{M+N} \to \mathcal{C}$ are contained in the middle action F^M. The following lemma makes this precise.

Lemma 2.11. *Let a regular functor $F : \mathcal{C}^{M+N} \to \mathcal{C}$ be given. We have*

$$\begin{aligned} F^L\left(X, \left\langle Y^L, Y^M, Y^R \right\rangle\right) &= F\left(X, Y^L\right), \\ F^R\left(X, \left\langle Y^L, Y^M, Y^R \right\rangle\right) &= F\left(X, Y^R\right) \end{aligned}$$

natural in $X : \mathcal{C}^M$ and $Y^L, Y^M, Y^R : \mathcal{C}^N$. In other words, we have commutativity of the following diagrams:

$$\begin{array}{ccc} \mathcal{C}^M \times \left(\mathcal{C}^3\right)^N & \xrightarrow{\mathrm{id} \times \left(\pi_1^3\right)^N} & \mathcal{C}^M \times \mathcal{C}^N \\ \downarrow{\scriptstyle \underline{F}} & & \downarrow{\scriptstyle F} \\ \mathcal{C}^3 & \xrightarrow{\pi_1^3} & \mathcal{C} \end{array} \qquad \begin{array}{ccc} \mathcal{C}^M \times \left(\mathcal{C}^3\right)^N & \xrightarrow{\mathrm{id} \times \left(\pi_3^3\right)^N} & \mathcal{C}^M \times \mathcal{C}^N \\ \downarrow{\scriptstyle \underline{F}} & & \downarrow{\scriptstyle F} \\ \mathcal{C}^3 & \xrightarrow{\pi_3^3} & \mathcal{C} \end{array}$$

Proof. By symmetry, it is sufficient to focus only on the left diagram. We induct on the structure of F, noting that all cases except for the one concerning parametric initial algebra formation are boring.

In the base cases where F is a variable selector, commutativity is direct from the definition of $\underline{F}$. Note that if F selects from its left domain factor, the outer context M, then the corresponding diagram for π_2^3 instead of π_1^3 would not commute.

If F is a finite coproduct or product of regular functors, then $F^L = \pi_1^3 \circ \underline{F}$ will by definition be the finite coproduct or product of their biased derivatives, respectively. Since π_1^3 preserves finite products and coproducts, commutativity of the diagram for F follows from that for its constituents.

Finally, the crucial case of F being a parametric initial algebra is taken care of by Lemma 2.7 since π_1^3 is left adjoint to the functor $\langle \mathrm{id}, 1, 1 \rangle : \mathcal{C} \to \mathcal{C}^3$. □

Given a parametric initial algebra

$$\begin{aligned} F &: \mathcal{C}^{M+N} \to \mathcal{C}, \\ F(X,Y) &= \mu Z.\, G(X,Y,Z) \end{aligned}$$

over a regular functor $G : \mathcal{C}^{M+N+1} \to \mathcal{C}$, then the preceding lemma enables us to simplify the mutually inductive definition of $\underline{F}$ as an initial algebra over $\mathcal{C}^3$. For two of its components, F^L and F^R, this is direct from the statement of the lemma. Now, Corollary 2.1 lets us write

$$\begin{aligned} F^M\left(X, \left\langle Y^L, Y^M, Y^R \right\rangle\right) &= \mu Z^M.\, G^M\left(X, \left\langle Y^L, Y^M, Y^R \right\rangle, \left(F\left(X, Y^L\right), Z^M, F\left(X, Y^R\right)\right)\right) \\ &= \mu Z^M.\, G^M\left(X, \left\langle \left(Y^L, F\left(X, Y^L\right)\right), \left(Y^M, Z^M\right), \left(Y^R, F\left(X, Y^R\right)\right)\right\rangle\right) \end{aligned} \tag{2.1}$$

natural in $X : \mathcal{C}^m$ and $Y^L, Y^M, Y^R : \mathcal{C}^n$. If we structurally build up $\underline{F}$ like this, then an important consequence is that F^M itself is a regular functor, albeit of asymptotically quadratic size (but of linear height).

The following two lemmata are the key results of this section. All primary use cases will in some form be instances of these results (together with later notes how the biased derivative interacts with guardedness and shieldedness).

Lemma 2.12. *Let a regular functor $F : \mathcal{C}^{M+N} \to \mathcal{C}$ be given. We have*

$$F^L\left(X, \left\langle Y^L, Y^M, Y^L + Y^M \right\rangle\right) + F^M\left(X, \left\langle Y^L, Y^M, Y^L + Y^M \right\rangle\right) = F\left(X, Y^L + Y^M\right)$$

natural in $X : \mathcal{C}^m$ and $Y^L, Y^M, Y^R : \mathcal{C}^n$. In other words, we have commutativity of the diagram

$$\begin{array}{ccc} \mathcal{C}^M \times \left(\mathcal{C}^2\right)^N & \xrightarrow{\mathrm{id} \times (\cdot + \cdot)^N} & \mathcal{C}^M \times \mathcal{C}^N \\ \downarrow{\scriptstyle \mathrm{id} \times u^M} & & \\ \mathcal{C}^M \times \left(\mathcal{C}^3\right)^N & & \\ \downarrow{\scriptstyle \underline{F}} & & \downarrow{\scriptstyle F} \\ \mathcal{C}^3 & & \\ \downarrow{\scriptstyle v} & & \\ \mathcal{C}^2 & \xrightarrow{\cdot + \cdot} & \mathcal{C} \end{array}$$

where $u : \mathcal{C}^2 \to \mathcal{C}^3$ and $v : \mathcal{C}^3 \to \mathcal{C}^2$ denote the operations

$$\begin{aligned} u(L, M) &= (L, M, L + M), \\ v(L, M, R) &= (L, M). \end{aligned}$$

Proof. We induct on the structure of F.

In the base cases where F is a variable selector, commutativity is rather direct from the definition of $\underline{F}$ and the fact that applications of u and v do not interfere with the first two components of each triplet.

If F is a finite coproduct of regular functors, then $\underline{F}$ will be the finite coproduct of their biased derivatives. As sums distribute over sums, all of u, v, and $\cdot + \cdot$ preserve finite coproducts. Commutativity of the diagram for F thus follows from that for its constituents.

For the case of finite products, introduce variables $X : \mathcal{C}^M$ and $Y^L, Y^M, Y^R : \mathcal{C}^N$. Let us abbreviate $\alpha = \big(X, \langle Y^L, Y^M, Y^L + Y^M\rangle\big)$ and $\beta = \big(X, Y^L + Y^M\big)$. With this notational convenience, the goal is to show that $F^L(\alpha) + F^M(\alpha) = F(\beta)$, natural in the variables X and Y^L, Y^M, Y^R.

If $F = 1$, then

$$F^L(\alpha) + F^M(\alpha) = 1 + 0 = 1 = F(\beta)$$

by definition of $\underline{F}$.

The most interesting case, if $F = G \times H$ with regular functors $G, H : \mathcal{C}^{M+N} \to \mathcal{C}$, then

$$F^L(\alpha) + F^M(\alpha) = G^L(\alpha) \times H^L(\alpha) + \big(G^M(\alpha) \times H^R(\alpha) + G^L(\alpha) \times H^M(\alpha)\big)$$

(by Lemma 2.11 to derive $H^R(\alpha) = H(\beta)$)

$$= G^L(\alpha) \times \big(H^L(\alpha) + H^M(\alpha)\big) + G^M(\alpha) \times H(\beta)$$

(by induction hypothesis for H)

$$\begin{aligned} &= G^L(\alpha) \times H(\beta) + G^M(\alpha) \times H(\beta) \\ &= \big(G^L(\alpha) \times G^M(\alpha)\big) \times H(\beta) \end{aligned}$$

(by induction hypothesis for G)

$$\begin{aligned} &= G(\beta) \times H(\beta) \\ &= F(\beta). \end{aligned}$$

Again, the crucial case of F being a parametric initial algebra is taken care of by Lemma 2.7 since $\cdot + \cdot$ is left adjoint to the diagonal functor $\Delta = \langle \mathrm{id}, \mathrm{id}\rangle : \mathcal{C} \to \mathcal{C}^2$. □

Lemma 2.13. *Let a regular functor $F : \mathcal{C}^{M+N} \to \mathcal{C}$ be given. Then F^M is linear in the middle parts of its argument triplets. In formulae, this expresses as*

$$F^M = \Big(\cdot|_{\mathrm{inr} \circ (1,\cdot)}\Big) \bullet \Big\langle F^M_{(n)} \Big\rangle_{n:N}$$

where for $n : N$ we abbreviated $F^M_{(n)} : \mathcal{C}^{M+2N} \to \mathcal{C}$ given by

$$F^M_{(n)}(X, \langle Y^L, Y^R\rangle) = F^M\left(X, \langle Y^L, E^N_n, Y^R\rangle\right)$$

where $E^N_n : \mathcal{C}^N$ denotes the category-level unit vector defined by

$$E^N_{i,j} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{else} \end{cases}$$

for $i, j : N$.

In perhaps more readable pointed style, the same equation reads as

$$\begin{aligned} F^M\left(X, \langle Y^L, Y^M, Y^R\rangle\right) &= Y^M \bullet \big\langle F^M\left(X, \langle Y^L, E^N_n, Y^R\rangle\right)\big\rangle_{n:N} \\ &= \sum_{n:N} Y^M_j \times F^M\left(X, \langle Y^L, E^N_n, Y^R\rangle\right) \end{aligned}$$

natural in $X : \mathcal{C}^M$ and $Y^L, Y^M, Y^R : \mathcal{C}^N$.

Proof. Again, we induct on the structure of F.

In case F is a variable selector for an argument from its left domain factor, the outer context M, the left-hand side and the second argument of the scalar product on the right-hand side of the equation for F will vanish.

In case $F(X,Y) = Y_n$ for $X : \mathcal{C}^M$ and $Y : \mathcal{C}^N$ with $n : N$, we have

$$\begin{aligned} Y^M \bullet \left\langle F^M\left(X, \left\langle Y^L, E_n^N, Y^R\right\rangle\right)\right\rangle_{n:N} &= Y^M \bullet E_n^N \\ &= Y_n^M \\ &= F^M\left(X, \left\langle Y^L, Y^M, Y^R\right\rangle\right) \end{aligned}$$

natural in $X : \mathcal{C}^M$ and $Y^L, Y^M, Y^R : \mathcal{C}^N$.

If F is a finite coproduct of regular functors, the equation for F will be the result of summing the individual equations for the constituting summands of F as the scalar product is linear in its second component.

If $F = 1$, then — similar to the first case — the left-hand side and the second argument of the scalar product on the right-hand side of the equation for F will vanish.

Let us now examine the more interesting case of $F = G \times H$ with regular functors $G, H : \mathcal{C}^{M+N} \to \mathcal{C}$. Abbreviating $\alpha = \left(X, \left\langle Y^L, Y^M, Y^R\right\rangle\right)$ and $\beta_n = \left(X, \left\langle Y^L, E_n^N, Y^R\right\rangle\right)$ for $n : N$, we have

$$\begin{aligned} & Y^M \bullet \left\langle F^M(\alpha_n)\right\rangle_{n:N} \\ = {} & Y^M \bullet \left\langle G^M(\alpha_n) \times H^R \alpha_n + G^L(\alpha_n) \times H^M(\alpha_n)\right\rangle_{n:N} \end{aligned}$$

(by Lemma 2.11 to derive $H^R(\alpha_n) = H\left(X, Y^R\right)$ and $G^L(\alpha_n) = G\left(X, Y^L\right)$ and thereby remove their dependency on $n : N$)

$$= Y^M \bullet \left\langle G^M(\alpha_n) \times H\left(X, Y^R\right) + G\left(X, Y^L\right) \times H^M(\alpha_n)\right\rangle_{n:N}$$

(by linearity of the scalar product in its second argument)

$$= \left(Y^M \bullet \left\langle G^M(\alpha_n)\right\rangle_{n:N}\right) \times H\left(X, Y^R\right) + G\left(X, R^L\right) \times \left(Y^M \bullet \left\langle H^M(\alpha_n)\right\rangle_{n:N}\right)$$

(by induction hypothesis)

$$= G^M(\beta) \times H\left(X, Y^R\right) + G\left(X, Y^L\right) \times H^M(\beta)$$

(by Lemma 2.11 in reverse)

$$\begin{aligned} &= G^M(\beta) \times H^R(\beta) + G^L(\beta) \times H^M(\beta) \\ &= F^M(\beta) \end{aligned}$$

natural in $X : \mathcal{C}^M$ and $Y^L, Y^M, Y^R : \mathcal{C}^N$.

Finally, let us consider the case where F is a parametric initial algebra of a regular functor $G : \mathcal{C}^{M+N+1} \to \mathcal{C}$, i.e. $F(X,Y) = \mu Z.\, G(X,Y,Z)$ with $X : \mathcal{C}^M$ and $Y : \mathcal{C}^N$. Let us introduce variables $X : \mathcal{C}^M$ and $Y^L, Y^M, Y^R : \mathcal{C}^N$. Abbreviating

$$\alpha = \left(X, \left(\left\langle \left(Y^L, F\left(X, Y^L\right)\right), \left(Y^R, F\left(X, Y^R\right)\right)\right\rangle\right)\right)$$

and starting with observation (2.1), note that

$$F^M\left(X, \left\langle Y^L, Y^M, Y^R\right\rangle\right) = \mu Z^M.\, G^M\left(X, \left(\left\langle \left(Y^L, F\left(X, Y^L\right)\right), \left(Y^M, Z^M\right), \left(Y^R, F\left(X, Y^R\right)\right)\right\rangle\right)\right)$$

(by induction hypothesis)

$$= \mu Z^M.\, Y^M \bullet \left\langle G^M_{(\mathrm{inl}(n))}(\alpha)\right\rangle_{n:N} + Z^M \times G^M_{(\mathrm{inr}(\bullet))}(\alpha)$$

(by Lemma 2.6)

$$= \left(Y^M \bullet \left\langle G^M_{(\mathrm{inl}(n))}(\alpha)\right\rangle\right) \times \mathrm{List}\left(G^M_{(\mathrm{inr}(\bullet))}(\alpha)\right)_{n:N}$$

(by linearity of the scalar product in its second component)

$$= Y^M \bullet \left\langle G^M_{(\mathrm{inl}(n))}(\alpha) \times \mathrm{List}\left(G^M_{(\mathrm{inr}(\bullet))}(\alpha)\right)\right\rangle_{n:N}.$$

This has already brought $F^M\left(X, \left\langle Y^L, Y^M, Y^R\right\rangle\right)$ into a form linear in Y^M. By evaluation, we find that

$$F^M_{(n)}\left(X, \left\langle Y^L, Y^R\right\rangle\right) = G^M_{(\mathrm{inl}(n))}(\alpha) \times \mathrm{List}\left(G^M_{(\mathrm{inr}(\bullet))}(\alpha)\right)$$

for $n : N$, thus validating the target equation. □

Combining the previous two lemmata, we forge the blade of our Swiss army knife:

Corollary 2.3. *Let a regular functor $F : \mathcal{C}^{M+N} \to \mathcal{C}$ be given. Then,*

$$F\left(X, Y^L + Y^M\right) = F\left(X, Y^L\right) + \sum_{n:N} Y^M_j \times F^M_{(n)}\left(X, \left\langle Y^L, Y^L + Y^M\right\rangle\right)$$

Proof. One half is given by Lemma 2.12 to establish the basic additive splitting. The other half is given by refining the right summand according to Lemma 2.13. Note that we made use of Lemma 2.11 in rewriting the left summand. □

Here is a specific instantiation used quite often later on:

Corollary 2.4. *Let a regular functor $F : \mathcal{C}^{M+1} \to \mathcal{C}$ be given. Then,*

$$F(X, Y) = F(X, 0) + Y \times F^M_{(\bullet)}\left(X, \langle 0, Y\rangle\right)$$

natural in $X : \mathcal{C}^M$ and $Y : \mathcal{C}$.

Proof. In Corollary 2.3, choose $N = 1$ as well as $Y^L = Y$ and $Y^M = 0$. □

Note that $F^M_{(\bullet)}\left(X, \langle Y, Y\rangle\right)$ yields the usual notion of derivative of a regular functor. This explains our choice of name of biased derivative: the plain derivative results from choosing symmetric (unbiased) arguments for $F^M_{(\bullet)}$.

Let us close this subsection by analyzing how forming the coefficients of the right summand in Corollary 2.3 interacts with guardedness and shieldedness.

Lemma 2.14. *Consider a regular functor $F : \mathcal{C}^{M+N} \to \mathcal{C}$ and fix $I \subseteq M$ and $J \subseteq N$.*

- *If F is guarded over $I \sqcup J$, then $F^M_{(n)}$ is guarded over $I \sqcup 2J \subseteq M + 2N$ for $n : N$.*
- *If F is shielded over $I \sqcup J$, then $F^M_{(n)}$ is shielded over $I \sqcup 2J \subseteq M + 2N$ for all $n : N \setminus J$.*

Proof. We prove both claims simultaneously by induction on guardedness and shieldedness as a mutual family.

Guardedness In case F is either a variable selector or the terminal object functor 1, then $F^M_{(n)}$ will be a regular functor with no occurrence of initial algebra formation, making it trivially guarded over $I \sqcup 2J$.

As all involved constructions distribute over finite coproducts, the case of finite coproducts is not particularly interesting. For completeness, if F is a finite coproduct of functors $G_k : \mathcal{C}^{M+N} \to \mathcal{C}$ guarded over $I \sqcup J$ for $k : K$, then $F^M_{(n)}$ will be the finite coproduct of $(G_k)^M_{(n)}$ for $k : K$. The latter summands are guarded over $I \sqcup 2J$ by induction hypothesis, thus verifying the same assertion for $F^M_{(n)}$.

In case $F = G \times H$ with functors $G, H : \mathcal{C}^{M+N} \to \mathcal{C}$ guarded over $I \sqcup J$, recall with Lemma 2.11 that

$$F^M_{(n)}\left(X, \langle Y^L, Y^R\rangle\right) = G^M_{(n)}\left(X, \langle Y^L, Y^R\rangle\right) \times H(X, Y^R) + G(X, Y^L) + H^M_{(n)}\left(X, \langle Y^L, Y^R\rangle\right)$$

natural $X : \mathcal{C}^M$ and $Y^L, Y^R : \mathcal{C}^N$. By induction hypothesis and compositionality of guardedness, all factors in the above expression of $F^M_{(n)}$ as a sum of products of functors $\mathcal{C}^{M+2N} \to \mathcal{C}$ are guarded over $I \sqcup 2J$, showing the same for $F^M_{(n)}$.

Finally, consider the case that F is the parametric initial algebra of $G : \mathcal{C}^{M+(N+1)} \to \mathcal{C}$ shielded over $I \sqcup (J \sqcup \emptyset)$. Recall from the proof of the Lemma 2.13 that

$$F^M_{(n)} = \left(G^M_{(\mathrm{inl}(n))} \times \left(\mathrm{List} \circ G^M_{(\mathrm{inr}(\bullet))}\right)\right) \circ K$$

where $K : \mathcal{C}^{M+2N} \to \mathcal{C}^{M+2(N+1)}$ is defined by

$$K\left(X, \langle Y^L, Y^R\rangle\right) = \left(X, \left(\left\langle \left(Y^L, F\left(X, Y^L\right)\right), \left(Y^R, F\left(X, Y^R\right)\right)\right\rangle\right)\right).$$

Utilizing both claims of the induction hypothesis, we see that $G^M_{(\mathrm{inl}(n))}$ is guarded and $G^M_{(\mathrm{inr}(\bullet))}$ is shielded over $I \sqcup 2(J \sqcup \emptyset)$, respectively. Here, the latter consequence crucially depended on $\mathrm{inr}(\bullet) \notin J \sqcup \emptyset$. Since List is a guarded endofunctor, postcomposing with it transforms shieldedness into guardedness. We hence see that

$$G^M_{(\mathrm{inl}(n))} \times \left(\mathrm{List} \circ G^M_{(\mathrm{inr}(\bullet))}\right)$$

is guarded over $I \sqcup 2(J \sqcup \emptyset)$.

Note that the component functors of K inherit guardedness over $I \sqcup 2J$ from guardedness of F over $I \sqcup J$. Since those component functors of K that are indexed by $I \sqcup 2(J \sqcup \emptyset)$ are trivially shielded over $I \sqcup 2J$, it finally follows that $F^M_{(n)}$ is guarded over $I \sqcup 2J$.

Shieldedness Let $F : \mathcal{C}^{M+N} \to \mathcal{C}$ be a functor shielded over $I \sqcup J \subseteq M + N$. This means we have a decomposition

$$F(X, Y) = X|_I \bullet \langle U_i(X, Y)\rangle_{i:I} + Y|_J \bullet \langle V_j(X, Y)\rangle_{j:J}$$

where $X : \mathcal{C}^M$ and $Y : \mathcal{C}^N$ with functors $U_i, V_j : \mathcal{C}^{M+N}$ guarded over $I \sqcup J$ for $i : I$ and $j : J$, respectively.

Unfolding the definition of $\underline{F}$ over products of sums, we have

$$F^M(\beta) = X|_I \bullet \left\langle U^M_i(\beta)\right\rangle_{i:I} + Y^L\big|_J \bullet \left\langle V^M_j(\beta)\right\rangle_{j:J} + Y^M\big|_J \bullet \left\langle V^R_j(\beta)\right\rangle_{j:J}$$

where $\beta = \left(X, \left\langle Y^L, Y^M, Y^R\right\rangle\right) : \mathcal{C}^{M+3N}$.

In the context of $n : N \setminus J$, first note that $E^N_n\big|_J = 0$. This explains the last summand having vanished in the equation

$$F^M_{(n)}(\alpha) = X|_I \bullet \left\langle (U_i)^M_{(n)}(\alpha)\right\rangle_{i:I} + Y^L\big|_J \bullet \left\langle (V_j)^M_{(n)}(\alpha)\right\rangle_{j:J}$$

natural in $\alpha = \left(X, \left\langle Y^L, Y^R\right\rangle\right) : \mathcal{C}^{M+2N}$. Since $(U_i)^M_{(n)}$ and $(V_j)^M_{(n)}$ are all guarded over $I \sqcup 2J \subseteq M + 2N$ for $i : I$ and $j : J$, respectively, we see that $F^M_{(n)}$ is shielded over $I \sqcup 2J$. [7] □

[7] Specifically, $F^M_{(n)}$ is $(I \sqcup \{0\} \times J)$-shielded over $I \sqcup 2J$. We shall not need this level of detail here.

2.3.3 The Biased Derivative: Applications

Lemma 2.15. *Let regular functors $F : \mathcal{C}^{N+1} \to \mathcal{C}$ and $C : \mathcal{C}^N \to \mathcal{C}$ be given. We have*

$$\mu Y.\, F(X,Y) + C(X) = (\mu Y.\, F(X,Y)) + C(X) \times U(X)$$

natural in $X : \mathcal{C}^N$. Here, $U : \mathcal{C}^N \to \mathcal{C}$ abbreviates the regular functor given by

$$U(X) = \mathrm{List}\left(F^M_{(\bullet)}\left(X, (\mu Y.\, F(X,Y), \mu Y.\, F(X,Y) + C(X))\right)\right).$$

Proof. Consider the functor $H : \mathcal{C}^{N+1} \to \mathcal{C}$ defined by $H(X,T) = \mu Y.\, F(X,Y) + T$. Instantiating Corollary 2.4 with $F = H$ and $Y = C(X)$, we read off

$$H(X, C(X)) = H(X,0) + C(X) \times H^M_{(\bullet)}(X, (0, C(X)))$$

natural in $X : \mathcal{C}^N$, which is precisely the equation we were looking for.

For the shape of U, note that by observation (2.1) together with Lemma 2.10, we have

$$\begin{aligned} H^M_{(\bullet)}(X, (0, C(X))) &= H^M(X, (0, 1, C(X))) \\ &= \mu Y^M.\, F^M\left(X, (H(X,0), Y^M, H(X, C(X)))\right) + 1 \end{aligned}$$

(by linearity of F^M as per Lemma 2.13)

$$\begin{aligned} &= \mu Y^M.\, Y^M \times F^M\left(X, (H(X,0), 1, H(X, C(X)))\right) + 1 \\ &= \mathrm{List}\left(F^M\left(X, (H(X,0), 1, H(X, C(X)))\right)\right) \\ &= \mathrm{List}\left(F^M_{(\bullet)}\left(X, (H(X,0), H(X, C(X)))\right)\right) \end{aligned}$$

□

Surprisingly, the above result can be generalized to the case when C also depends on Y with the variable conventions of the above equation. This is perhaps striking as the previous proof crucially depended on $C(X)$ being constant so that it could be abstracted over as a parameter of H outside of the binding context of μ over Y.

Lemma 2.16. *Let regular functors $F, G : \mathcal{C}^{N+1} \to \mathcal{C}$ be given. Introduce a regular functors $R : \mathcal{C}^N \to \mathcal{C}$ abbreviating $R(X) = \mu Y.\, F(X,Y) + G(X,Y)$. We have $R = (\mu Y.\, F(\cdot, Y)) + (G \circ \langle \mathrm{id}, R\rangle) \times U$, i.e.*

$$\mu Y.\, F(X,Y) + G(X,Y) = (\mu Y.\, F(X,Y)) + G(X, R(X)) \times U(X)$$

natural in $X : \mathcal{C}^N$. Here, $U : \mathcal{C}^N \to \mathcal{C}$ is the regular functor given by

$$U(X) = \mathrm{List}\left(F^M_{(\bullet)}\left(X, (\mu Y.\, F(X,Y), \mu Y.\, F(X,Y) + G(X,Y))\right)\right).$$

Proof. Convince yourself of the following chain of equations natural in $X : \mathcal{C}^N$:

$$R(X) = \mu Y.\, F(X,Y) + G(X,Y)$$

(by the diagonal rule)

$$= \mu Y_1.\, \mu Y_2.\, F(X, Y_2) + G(X, Y_1)$$

(unfolding the outer initial algebra once)

$$= \mu Y_2.\, F(X, Y_2) + G(X, \mu Y_1.\, \mu Y_2.\, F(X, Y_2) + G(X, Y_1))$$

(by the diagonal rule in reverse)

$$\begin{aligned} &= \mu Y_2.\, F(X, Y_2) + G(X, \mu Y.\, F(X,Y) + G(X,Y)) \\ &= \mu Y.\, F(X,Y) + G(X, R(X)). \end{aligned}$$

With this, the desired equation is directly given by Lemma 2.15 for $C = G \circ \langle \mathrm{id}, R\rangle$. □

Lemma 2.17. *Let a regular functor $F : \mathcal{C}^n \to \mathcal{C}$ be given where $n : \mathbb{N}$. There exist regular functors $U_i : \mathcal{C}^{[n]\setminus[i]} \to \mathcal{C}$ for $i < n$ such that*

$$F(X_0, \ldots, X_{n-1}) = F(0) + \sum_{i<n} X_i \times U_i(X_i, \ldots, X_{n-1})$$

natural in $X_0, \ldots, X_{n-1} : \mathcal{C}$. If F is furthermore guarded over $I \subseteq [n]$, then U_i will be guarded over $I \setminus [i]$ for $i < n$.

Proof. By induction on n, with the base $n = 0$ being trivial.

In the induction step, apply Corollary 2.4 with the derivative of F being taken with respect to its first variable, i.e. mapping its parameter index set under the isomorphism $[n] = 1 + [n-1]$ witnessed in the reverse direction by $\langle 0, 1 + \cdot \rangle$. With the latter isomorphism being treated implicitly, we read off

$$F(X_0, X_1, \ldots, X_{n-1}) = F(0, X_1, \ldots, X_{n-1}) + X_0 \times F^M_{(\bullet)}\left((X_1, \ldots, X_{n-1}), \langle 0, X_0 \rangle\right).$$

Using the induction hypothesis on the regular functor of arity $n-1$ mapping $(X_1, \ldots, X_n)$ to $F(0, X_1, \ldots, X_n)$, we derive a decomposition

$$F(0, X_1, \ldots, X_{n-1}) = F(0) + \sum_{1 \le i < n} X_i \times U_i(X_i, \ldots, X_{n-1})$$

with regular functors $U_i : \mathcal{C}^{n-i} \to \mathcal{C}$ for $1 \le i < n$. Substituting the right-hand side expression for $F(0, X_1, \ldots, X_{n-1})$ into the previous equation while defining the regular functor $U_0 : \mathcal{C}^n \to \mathcal{C}$ as

$$U_0(X_0, \ldots, X_{n-1}) = F^M_{(\bullet)}\left((X_1, \ldots, X_{n-1}), \langle 0, X_0 \rangle\right),$$

we establish the desired decomposition

$$F(X_0, \ldots, X_{n-1}) = F(0) + \sum_{i<n} X_i \times U_i(X_i, \ldots, X_{n-1}).$$

Now assume F is guarded over $I \subseteq [n]$. Guardedness of U_0 in I follows from Lemma 2.14. By compositionality of guardedness, we see that $F(0, X_1, \ldots, X_{n-1})$ is guarded over $I \setminus [1]$. Guardedness of U_i in $I \setminus [i]$ for $i \ge 1$ then follows by induction. □

Given a finite parameter index set M, a functor $\mathcal{C}^M \to \mathcal{C}$ is called *polynomial* if it is built out of variable selectors as well as finite products and coproducts. Given finite parameter index sets M and N, a functor $K : \mathcal{C}^M \to \mathcal{C}^N$ is called *generalized polynomial* if all its component functors are polynomial. Polynomial functors and generalized polynomial functors are closed under composition.

Lemma 2.18. *Let $F : \mathcal{C}^M \to \mathcal{C}$ be a regular functor. Then there exists a finite parameter index set N and a generalized polynomial functors $K : \mathcal{C}^{M+N} \to \mathcal{C}^N$ such that*

$$F(X) = \pi^N_k\left(\mu Y.\, K(X, Y)\right)$$

natural in $X : \mathcal{C}^M$ for some $k : N$.

Proof. For technical reasons, we will instead first prove a weaker but more efficient claim by induction on the structure of regular functors.

A weaker invariant For any regular functor $F : \mathcal{C}^M \to \mathcal{C}$ there exists a generalized polynomial functors $K : \mathcal{C}^{M+N} \to \mathcal{C}^N$ and a polynomial functor $R : \mathcal{C}^{M+N} \to \mathcal{C}$ such that

$$F(X) = R\left(X, \mu Y.\, K(X, Y)\right).$$

The induction If $F(X) = X_m$ for some $m : M$ or $F = 0$ or $F = 1$, then we choose $N = \emptyset$ and $R = F$.

Given regular functors $F_1, F_2 : \mathcal{C}^M \to \mathcal{C}$, assume that we already have generalized polynomial functors $K_1 : \mathcal{C}^{M+N_1} \to \mathcal{C}^{N_1}$ and $K_2 : \mathcal{C}^{M+N_2} \to \mathcal{C}^{N_2}$ as well as polynomial functors $R_1 : \mathcal{C}^{M+N_1} \to \mathcal{C}$ and $R_2 : \mathcal{C}^{M+N_2} \to \mathcal{C}$ such that

$$\begin{aligned} F_1(X) &= R_1\left(X, \mu Z_1.\, K_1(X, Z_1)\right), \\ F_2(X) &= R_2\left(X, \mu Z_2.\, K_2(X, Z_2)\right). \end{aligned}$$

Define a combined generalized polynomial functor $K : \mathcal{C}^{M+N_1+N_2} \to \mathcal{C}^{N_1+N_2}$ by setting

$$\begin{aligned} K_{\mathrm{inl}(n_1)}(X, Z_1, Z_2) &= K_1(X, Z_1), \\ K_{\mathrm{inr}(n_2)}(X, Z_1, Z_2) &= K_2(X, Z_2) \end{aligned}$$

for $n_1 : N_1$ and $n_2 : N_2$, respectively. Given an arbitrary binary polynomial functor $Q : \mathcal{C}^2 \to \mathcal{C}$, define $R : \mathcal{C}^{X+N_1+N_2} \to \mathcal{C}$ by $R(X, Z_1, Z_2) = Q(R_1(X, Z_1), R_2(X, Z_2))$. By the binary mutual recursion theorem, it follows that

$$\begin{aligned} R\left(X, \mu(Z_1, Z_2).\, K(X, Z_1, Z_2)\right) &= R\left(\mu Z_1.\, K_1(X, Z_1), \mu Z_2.\, K_2(X, Z_2)\right) \\ &= Q(R_1(X, \mu Z_1.\, K_1(X, Z_1)), R_2(X, \mu Z_2.\, K_1(X, Z_2))) \\ &= Q(F_1(X), F_2(X)). \end{aligned}$$

Now the claim for $F_1 + F_2$ and $F_1 \times F_2$ follows by choosing Q accordingly.

Finally, let F be the parametric initial algebra of a regular functor $G : \mathcal{C}^{M+1} \to \mathcal{C}$. Assume that we have a generalized polynomial functor $L : \mathcal{C}^{M+1+O} \to \mathcal{C}^O$ as well as a polynomial functor $S : \mathcal{C}^{M+1+O} \to \mathcal{C}$ such that

$$G(X, Y) = S\left(X, Y, \mu Z.\, L(X, Y, Z)\right).$$

Let us define an extended generalized polynomial functor $K : \mathcal{C}^{M+1+O} \to \mathcal{C}^{1+O}$ by setting $K_{\mathrm{inl}(\bullet)}(X, Y, Z) = S(X, Y, Z)$ and $K_{\mathrm{inr}(o)}(X, Y, Z) = L_o(X, Y, Z)$ for $o : O$. By the mutual recursion theorem, we have

$$\begin{aligned} \pi_{\mathrm{inl}(\bullet)}\left(\mu(Y, Z).\, K(X, Y, Z)\right) &= \mu Y.\, S\left(X, Y, \mu Z.\, L(X, Y, Z)\right) \\ &= \mu Y.\, G(X, Y) \\ &= F(X). \end{aligned}$$

Choosing the variable selection polynomial functor $R : \mathcal{C}^{M+1+O} \to \mathcal{C}$ given by $R(X, Y, Z) = Y$, we thus have verified the claim for F.

Deriving the stronger claim Note that the case of F being a parametric initial algebra returned a result selection functor R of variable selector shape as desired in the stronger claim of the lemma. Given an arbitrary regular functor $F : \mathcal{C}^M \to \mathcal{C}$, we may thus simply apply the above algorithm to the degenerate parametric initial algebra mapping $X : \mathcal{C}^M$ to $\mu Y.\, F(X)$, which of course still equals F. □

Let us now prove the main theorem of this section, establishing the decomposition of any regular functor into a guarded and an indefinite part. We apologize to the reader in advance, for during write-up, we noticed a mistake in our original proof, which we were not able to fix in the brief time available. The original proof was based on the tools developed above and in the preceding subsection, aiming for a more local nested approach. In contrast, the replacement proof uses a global mutual approach, explicitly requiring the component functors of the mutual specification to be polynomial. It is thus less modular with respect to future possible extensions of the language of functors under consideration. Our hope is that the original mistake is fixable, so that a future presentation may be more streamlined.

Note however, some of the results derived above, in particular Lemma 2.16, are still used.

Theorem 2.1 (Decomposition theorem)**.** *For any regular functor $F : \mathcal{C}^M \to \mathcal{C}$, there are regular functors $U^F, V^F : \mathcal{C}^M \to \mathcal{C}$ with U^F guarded such that*

$$F(X) = U^F(X) + \mathbf{N} \times V^F(X)$$

natural in $X : \mathcal{C}^M$, or $F = U^F + \mathbf{N} \times V^F$ for short.

Proof. By Lemma 2.18, there is a generalized polynomial functor $K : \mathcal{C}^{M+N} \to \mathcal{C}^N$ with $F(X) = R(X, \mu Y.\,(K(X,Y)))$ for some polynomial functor $R : \mathcal{C}^{N+M} \to \mathcal{C}$. Our goal will be to find a suitable decomposition for K and then reflect it back through R.

Given a finite parameter index set A, let $\underline{\mathrm{Id}} : \mathcal{C}^{M+A} \to \mathcal{C}^M$ denote the local identity functor $\underline{\mathrm{Id}}(X,Y) = Y$ in contexts of type $\mathcal{C}^M$, and given $S : \mathcal{C}^{M+A} \to \mathcal{C}^B$ and $T : \mathcal{C}^{M+B} \to \mathcal{C}^C$, let

$$\begin{aligned} T \mathbin{\underline{\circ}} S \quad &: \mathcal{C}^{M+A} \to \mathcal{C}^C, \\ (T \mathbin{\underline{\circ}} S)(X,Y) &= T(X, S(X,Y)) \end{aligned}$$

denote the local composition of S and T in contexts of type $\mathcal{C}^M$. Given $S : \mathcal{C}^{M+A} \to \mathcal{C}^A$, introduce abbreviations $S^{\underline{n}} = S \mathbin{\underline{\circ}} \ldots \mathbin{\underline{\circ}} S$ for $n : \mathbf{N}$ and $\underline{\mu} S : \mathcal{C}^M \to \mathcal{C}^A$ given by $(\underline{\mu} S)(X) = \mu Y.\, S(X,Y)$. These definitions can be made elegant formally by working in the Kleisli category over the reader monad over $\mathcal{C}^M$.

Stability of initial coefficients For any generalized polynomial $S : \mathcal{C}^{M+A} \to \mathcal{C}^B$, let $C(S)$ denote the set of those $a : A$ such that the $\mathbf{N}$-coefficient of the monomial of degree zero in S_a is non-zero, i.e. $S_a(0) \geq 1$. Fixing a generalized polynomial $T : \mathcal{C}^{M+B} \to \mathcal{C}^C$, observe that $C(T \mathbin{\underline{\circ}} S)$ is a monotone function of $C(S)$ for generalized polynomial $S : \mathcal{C}^{M+A} \to \mathcal{C}^B$. Since $C\left(\underline{\mathrm{Id}}\right) = \emptyset \subseteq C(K)$, we in particular have an ascending chain

$$\emptyset = C\left(K^{\underline{0}}\right) \subseteq C\left(K^{\underline{1}}\right) \subseteq \ldots$$

that stabilizes after attaining its first equality. But $C\left(K^{\underline{n}}\right) \subseteq N$, so there must be a minimal $k \geq 1$ with $k \leq |N|$ with such that $C\left(K^{\underline{k}}\right) = C\left(K^{\underline{k+1}}\right)$, and hence also $C\left(K^{\underline{k}}\right) = C\left(K^{\underline{2k}}\right)$. Since $\mu Y.\, K(X,Y) = \mu Y.\, K^{\underline{k}}(X,Y)$ natural in $X : \mathcal{C}^M$ by the n-powered version of the squaring rule Lemma 2.5, [8], we may thus substitute $K^{\underline{k}}$ for K and without loss of generality assume that $C\left(K \mathbin{\underline{\circ}} K\right) = C(K)$.

Removal of initial coefficients Decompose $K(X,Y) = S + T(X,Y)$ with $S : \mathcal{C}^N$ and $T : \mathcal{C}^{M+N} \to \mathcal{C}^N$ where S denotes the monomials of K having degree zero. Writing $Q : \mathcal{C}^{M+N} \to \mathbb{N}$ for $Q(X,Y) = S + Y$, the rolling rule Lemma 2.4 tell us that

$$\begin{aligned} F &= R \mathbin{\underline{\circ}} \underline{\mu} K \\ &= R \mathbin{\underline{\circ}} \underline{\mu}(Q \mathbin{\underline{\circ}} T) \\ &= (R \mathbin{\underline{\circ}} Q) \mathbin{\underline{\circ}} \underline{\mu}(T \mathbin{\underline{\circ}} Q) \end{aligned}$$

where we have identified $\mathcal{C}^N$ with $\mathcal{C}^{0+N}$.

From $K \mathbin{\underline{\circ}} K = K$, we derive $S \mathbin{\underline{\circ}} K \mathbin{\underline{\circ}} K = S \mathbin{\underline{\circ}} K$. Recall functionality of $C(U \mathbin{\underline{\circ}} V)$ in $C(V)$ for fixed U. Since $C(T) = \emptyset = C(\underline{\mathrm{Id}})$, precomposition with T does not change the value of C for any given argument. We continue deriving $S \mathbin{\underline{\circ}} K \mathbin{\underline{\circ}} T = S \mathbin{\underline{\circ}} T$, i.e.

$$(S \mathbin{\underline{\circ}} T)^{\underline{2}} = S \mathbin{\underline{\circ}} T.$$

Substituting $S \mathbin{\underline{\circ}} T$ for K and $R \mathbin{\underline{\circ}} Q$ for R, we may thus without loss of generality assume that $C(K) = \emptyset$, i.e. $K(0,0) = 0$.

[8] Alternatively, we may enlarge k to the nearest power of 2 and avoid the need for generalizing the squaring rule.

Delooping For any generalized polynomial $S : \mathcal{C}^{M+A} \to \mathcal{C}^B$, let $U^S : \mathcal{C}^A \to \mathcal{C}^B$ denote the linear part of $S(0, \cdot)$ and $V^S : \mathcal{C}^{M+A} \to \mathcal{C}^B$ the remainder, i.e.

$$S(X, Y) = U^S(X) + V^S(X, Y).$$

Given generalized polynomials $S : \mathcal{C}^{M+A} \to \mathcal{C}^B$ and $T : \mathcal{C}^{M+B} \to \mathcal{C}^C$, note that

$$\begin{aligned}(T \underline{\circ} S)(X, Y) &= U^T(S(X, Y)) + V^T(X, S(X, Y)) \\ &= U^T(U^S(X)) + U^T(V^T(X, Y)) + V^T(X, S(X, Y)),\end{aligned}$$

i.e. $U^{T \underline{\circ} S} = U^T \circ U^S$ and $V^{T \underline{\circ} S} = U^T \circ V^T + V^T \underline{\circ} S$, implying that the operator U preserves composition. Given generalized polynomial $S : \mathcal{C}^{M+A} \to \mathcal{C}^B$, let further $U'(S) \subseteq A \times B$ denote the relation relating $a : A$ and $b : B$ if U^S_a has non-zero coefficient in front of the monomial π^B_b. Again, note that U' translates composition of functors into composition of relations.

Since $U'(K) \subseteq N \times N$ is an endorelation with finite domain, there must be $k : \mathbb{N}$ such that $U'(K)^k$ is transitive. [9]. By substituting $K^{\underline{k}}$ for K, assume without loss of generality that $U'(K)$ is transitive.

Let J consist of those $n : N$ such that $(n, n) \in U'(K)$. For ease of reasoning, identify $N = I + J$ with finite parameter index set I. Fixing $X : \mathcal{C}^M$ and $Y : \mathcal{C}^I$, we may thus write

$$K_2(X, (Y, Z)) = Z + Q(X, (Y, Z))$$

for generalized polynomial $Q : \mathcal{C}^{M+(I+J)}$. Abbreviating $T_{X,Y} = \mu Z.\, K_2(X, (Y, Z))$, we have

$$\begin{aligned}T_{X,Y} &= \mu Z.\, Z + Q(X, (Y, Z)) \\ &= (\mu Z.\, Z) + Q(X, (Y, T_{X,Y})) \times \mathrm{List}^J\left((\mathrm{Id}_{\mathcal{C}^J})^M_{(\bullet)}(\ldots)\right) \\ &= 0 + Q(X, (Y, T_{X,Y})) \times \mathrm{List}^J(1) \\ &= (\mathbf{N})_{j:J} \times Q(X, (Y, T_{X,Y}))\end{aligned}$$

by the higher dimensional analogue of the theory of biased derivatives of regular functors and in particular Lemma 2.16, and a trivial application of mutual recursion Corollary 2.1 to derive $\mathrm{List}^J(1) = (\mathbf{N})_{j:J}$.

By an application of binary mutual recursion Lemma 2.3, we then see that

$$\mu(Y, Z).\, \langle K_1, K_2 \rangle (X, (Y, Z)) = \left(\mu Y.\, K_1(X, (Y, T_{X,Y})), T_{X, \mu Y.\, K_1(X,(Y,T_{X,Y}))}\right)$$

We may rewrite the first component as follows:

$$\begin{aligned}\mu Y.\, K_1(X, (Y, T_{X,Y})) &= \mu Y.\, K_1(X, (Y, 0)) + T_{X,Y} \bullet \left\{ \begin{array}{c} \text{componentwise regular matrix} \\ \text{of dimension } J \times I \end{array} \right\} \\ &= \mu Y.\, K_1(X, (Y, 0)) + (\mathbf{N})_{i:I} \times \ldots \\ &= (\mu Y.\, K_1(X, (Y, 0))) + (\mathbf{N})_{i:I} \times \ldots.\end{aligned}$$

Here, the first step uses the higher dimensional analogue of Corollary 2.4, and the last step uses the higher dimensional analogue of Lemma 2.15. The omitted matrix and I-dimensional vectors have as entries regular functors applied to arguments X and Y.

In total, we have derived a decomposition

$$\underline{\mu} K = (\mu Y.\, K_1(X, (Y, 0)), 0) + (\mathbf{N})_{n:I+J} \times S$$

for some generalized regular functor $S : \mathcal{C}^{M+(I+J)} \to \mathcal{C}^{I+J}$. Since R is polynomial, it is easy to see that this translates into

$$F(X) = R\left(X, \underline{\mu} K\right) = R(X, (\mu Y.\, K_1(X, (Y, 0)), 0)) + \mathbf{N} \times \ldots$$

[9] Note that this is distinct from the transitive closure of $U'(K)$

where the omitted term is again regular. It remains is to solve the original problem for the generalized polynomial functor $K' : \mathcal{C}^{M+I} \to \mathcal{C}^I$ given by $K'(X,Y)$ to $K_1(X,(Y,0))$ and the result collecting functor $R' : \mathcal{C}^{M\times I} \to \mathcal{C}$ given by $R'(X,Y) = R(X,(Y,0))$.

Recall that $(i,i) \notin U'(K)$ for $i : I$ by construction of I. Since $U'(K)$ is transitive, this implies $U'(K') = U'(K) \cap (I \times I) = \emptyset$. Substituting I for N, and K' for K, and R' for R, we may hence without loss of generality assume that $U(K) = 0$.

The guarded part Recapitulating, all monomials with non-zero coefficient in the component functors of K that have the form $(X,Y) \mapsto Y^\alpha$ must have degree $|\alpha| \geq 2$. This assumption enables us to write

$$K_n(X,Y) = \sum_{i:M} X_i \times P_{n,i}(X,Y) + \sum_{j_1,j_2:N} Y_{j_1} \times Y_{j_2} \times Q_{n,j_1,j_2}(X,Y)$$

with generalized polynomials $P : \mathcal{C}^{M+N} \to \mathcal{C}^{N\times M}$ and $Q : \mathcal{C}^{M+N} \to \mathcal{C}^{N\times N\times N}$. Note that this is not a unique decomposition, but requires some particular, though arbitrary choice.

Consider the following generalized polynomial:

$$\begin{aligned}
H &: \mathcal{C}^{M+(N+M\times N)} \to \mathcal{C}^{N+N\times M},\\
H_{\mathrm{inl}(n)}(X,(Y,Z)) &= K_n\left(X, \sum_{i:M} X_i \times (P_{n,i}(X,Y) + Z_{n,i})\right),\\
H_{\mathrm{inr}(n,i)}(X,(Y,Z)) &= \sum_{j_1,j_2:N} (P_{j_1,i}(X,Y) + Z_{j_1,i})\\
&\qquad \times \left(\sum_{i':M} X_{i'} \times (P_{j_2,i'}(X,Y) + Z_{j_2,i'})\right) \times Q_{n,j_1,j_2}(X,Y).
\end{aligned}$$

Since $K_n(0,0) = 0$, note that K_n is shielded for all $n : N$. By composition of shieldedness, we therefore see that $H_{\mathrm{inl}(n)}(X,(Y,Z))$ is shielded over X. Even more direct, by distributivity, we see that $H_{\mathrm{inl}(n,i)}(X,(Y,Z))$ is shielded over X for all $n : N$ and $i : I$. By Lemma 2.8, it follows that each component functor of $\underline{\mu}H$ is guarded over its first argument.

By the rolling rule Lemma 2.4, note that H is related to the generalized polynomial

$$\begin{aligned}
H' &: \mathcal{C}^{M+(N+N\times M)} \to \mathcal{C}^{N+N\times M},\\
H'_{\mathrm{inl}(n)}(X,(Y,Z)) &= K_n\left(X, \sum_{i:M} X_i \times Z_{n,i}\right),\\
H'_{\mathrm{inr}(n,i)}(X,(Y,Z)) &= P_{n,i}(X,Y) + \sum_{j_1,j_2:N} Z_{j_1,i} \times \left(\sum_{i':M} X_{i'} \times Z_{j_2,i'}\right) \times Q_{n,j_1,j_2}(X,Y)
\end{aligned}$$

via the functor

$$\begin{aligned}
G &: \mathcal{C}^{M+(N+M\times N)} \to \mathcal{C}^{N+M\times N},\\
G(X,(Y,Z)) &= (X,(Y,P(X,Y)+Z)),
\end{aligned}$$

i.e. $\underline{\mu}(H') = G \circ \underline{\mu}(H)$, and thus in particular $\underline{\mu}(H')_1 = \underline{\mu}(H)$.

Next, consider the functor

$$\begin{aligned}
T &: \mathcal{C}^{M+(N+N)} \to \mathcal{C}^{N+N},\\
T_{\mathrm{inl}(n)}(X,(Y,Z)) &= K_n(X,Z_n),\\
T_{\mathrm{inr}(n)}(X,(Y,Z)) &= \sum_{i:M} X_i \times P_{n,i}(X,Y) + \sum_{j_1,j_2:N} Z_{j_1} \times Z_{j_2} \times Q_{n,j_1,j_2}(X,Y).
\end{aligned}$$

Fix $X : \mathcal{C}^M$ and introduce the functor

$$\begin{aligned} L &: \mathcal{C}^M \to \mathcal{C}, \\ L(Z) &= \sum_{i:M} X_i \times Z_i \end{aligned}$$

right adjoint to $U : \mathcal{C} \to \mathcal{C}^M$ defined by $U(Z) = \left(Z^{X_i}\right)_{i:M}$. Elementary algebra verifies commutativity of the following fusion diagram:

$$\begin{array}{ccc} \mathcal{C}^{N+N\times M} & \xrightarrow{\mathcal{C}^N \times L^N} & \mathcal{C}^{N+N} \\ \downarrow{\scriptstyle H'(X,\cdot)} & & \downarrow{\scriptstyle T(X,\cdot)} \\ \mathcal{C}^{N+N\times M} & \xrightarrow{\mathcal{C}^N \times L^N} & \mathcal{C}^{N+N} \end{array}$$

With the functor $\mathcal{C}^N \times L^N$ being left adjoint to $\mathcal{C}^N \times U^N$, we apply fusion Lemma 2.1 and derive

$$\underline{\mu}(T) = \left(\mathcal{C}^N \times L^N\right) \circ \underline{\mu}(H'),$$

and thus in particular $\underline{\mu}(T)_1 = \underline{\mu}(H')_1$.

Using partial squaring Corollary 2.2, we finally calculate

$$\begin{aligned} \mu(Y,Z).\, T(X,(Y,Z)) &= \mu(Y,Z).\, T(X,(T(X,(Y,Z)),Z)) \\ &= \mu(Y,Z).\, T(X,(K(X,Z),Z)) \\ &= \mu(Y,Z).\, (K(X,Z),K(X,Z)). \end{aligned}$$

By mutual recursion Lemma 2.3, we thus have $\underline{\mu}(T)_2 = \underline{\mu}K$ and $\underline{\mu}(T)_1 = K(X, \underline{\mu}(T)_2) = \underline{\mu}K$.

Putting the above chain of transformations together, the reader is invited to realize that $\underline{\mu}K = \underline{\mu}H_1$. Since each component functor of $\underline{\mu}H$ was guarded over its first argument, the same holds true for $\underline{\mu}K$. It follows that $F = R \circ \underline{\mu}K$ is a guarded functor.

□

2.4 The Set Model

2.4.1 Containers

An N-*ary container* $S \lhd P$ on the category Set of sets consists of a *shape* $S : \mathrm{Set}$ and a family $P : S \to \mathrm{Set}^N$ of *positions* over S [1]. It induces a *container extension* functor $[\![S \lhd P]\!] : \mathrm{Set}^N \to \mathrm{Set}$ by associating to families of sets $X : \mathrm{Set}^N$ the sum over S of products of powers of X_n for $n : N$ with exponents given by P:

$$\begin{aligned}[\![S \lhd P]\!]\,(X) &= \sum_{s:S} \prod X^{P(s)} \\ &= \sum_{s:S} \prod_{n:N} X_n^{P(s)_n}\end{aligned}$$

The category Cont_N of N-ary containers has as morphisms between containers $S_1 \lhd P_1$ and $S_2 \lhd P_2$ pairs (f, u) where $f : S_1 \to T_2$ is a function between the shapes and u is a family over $s : S_1$ of contravariant functions $u_s : P_2(f(s)) \to P_1(s)$ between the positions, with identity and composition defined in the obvious way. Recalling that the shape projection functor $(S \lhd P) \mapsto S$ is a fibration, a morphism (f, u) is cartesian (always implicitly with respect to this fibration) if u is a family of isomorphisms. Extension of n-ary containers is a fully faithful functor $[\![\cdot]\!] : \mathrm{Cont}_N \to (\mathrm{Set}^N \to \mathrm{Set})$. Let us recall some basic closure properties of containers.

Colimits Container extension creates colimits. Given a diagram $\mathcal{I} \to \mathrm{Cont}_N$ of containers written $(S_i \lhd P_i)_{i:\mathcal{I}}$, note that the family $P : \prod_{i:\mathcal{I}} S_i \to \mathrm{Set}$ forms a cocone over the diagram $S : \mathcal{I} \to \mathrm{Set}$. Its colimit is given by

$$\mathrm{colim}_{i:\mathcal{I}}(S_i \lhd P_i) = \mathrm{colim}\, S \lhd [P].$$

Coproducts The previous paragraph includes the special case of coproducts. Given a (discrete) family $(S_i \lhd P_i)_{i:I}$ of N-ary containers, its coproduct is given

$$\sum_{i:I} (S_i \lhd P_i) = \sum_{i:I} S_i \lhd [P_i]_{i:I}.$$

Products Container extension creates products. Given a (discrete) family $(S_i \lhd P_i)_{i:I}$ of N-ary containers, its product is given by

$$\prod_{i:I} (S_i \lhd P_i) = \prod_{i:I} S_i \lhd \lambda s. \sum_{i:I} P_i(s_i).$$

Variable selection The variable selection functors $\pi_n^N : \mathrm{Set}^N \to \mathrm{Set}$ with $n : N$ are representable by N-ary containers. In detail, we have that $[\![1 \lhd \lambda\_.\ e_n]\!] = \pi_n^N$ where $e_n : \mathrm{Set}^N$ is defined by

$$e_{i,j} = \begin{cases} 1 & \text{if } j = i, \\ 0 & \text{else.} \end{cases}$$

We will silently reuse the identifier π_n^N for $1 \lhd \lambda\_.\ e_n$.

Composition Given an n-ary container $S \lhd P$ and a family of K-ary containers $(T_n \lhd Q_n)_{n:N}$, the composition

$$[\![S \lhd P]\!] \circ \langle [\![T_n \lhd Q_n]\!] \rangle_{n:N}$$

is represented by the container

$$(S \lhd P) \circ \langle T_n \lhd Q_n \rangle_{n:N} = [\![S \lhd P]\!]\,(T) \lhd \sum_{n:N} \sum_{p:(Ps)_n} Q_n(t_{n,p}).$$

Note that this can be seen as the application to arguments $(T_n \lhd Q_n)_{n:I}$ of a generalization of the extension functor from target Set to target Cont_K. Composition is functorial in both $S \lhd P$ and the family $(T_n \lhd Q_n)_{n:I}$. The functorial action preserves cartesian morphisms.

Parametric initial algebras Container extension creates parametric initial algebras. In contrast to the previous properties, this is specific to our current setting Set. Let us further elaborate the details.

Fix an $(N{+}1)$-ary container $S \lhd P$ with positions written as $P = \langle P_1, P_2 \rangle$ with $P_1 : S \to \mathrm{Set}^N$ and $P_2 : S \to \mathrm{Set}$. Given any N-ary container $T \lhd Q$, note the equivalence

$$\begin{aligned}\overline{[\![S \lhd P]\!]}\,[\![T \lhd Q]\!] &= [\![S \lhd P]\!] \circ \left\langle \left\langle \pi_n^N \right\rangle_{n:N}, [\![T \lhd Q]\!] \right\rangle \\ &= \left[\!\left[(S \lhd P) \circ \left\langle \left\langle \pi_n^N \right\rangle_{n:N}, T \lhd Q \right\rangle \right]\!\right] \\ &= [\![E_{S\lhd P}(T \lhd Q)]\!]\,,\end{aligned}$$

with an endofunctor $E_{S\lhd P} : \mathrm{Cont}_N \to \mathrm{Cont}_N$ defined by

$$\begin{aligned}E_{S\lhd P}(T \lhd Q) &= (S \lhd P) \circ \left\langle \left\langle \pi_n^N \right\rangle_{n:N}, T \lhd Q \right\rangle \\ &= \underbrace{\sum_{s:S} T^{P_2(s)}}_{U(T)} \lhd \underbrace{\lambda(s,t).\ P_1(s) + \sum_{j:P_2(s)} Q(t_j)}_{V(T,Q)}\end{aligned}$$

where we have introduced additional notation $U : \mathrm{Set} \to \mathrm{Set}$ and

$$V : \prod_{T:\mathrm{Set}} \prod_{Q:T\to\mathrm{Set}^N} U(T) \to \mathrm{Set}^N\,.$$

Abstractly speaking, we have just verified that the following diagram of functors commutes:

$$\begin{array}{ccc} \mathrm{Cont}_N & \xrightarrow{E_{S\lhd P}} & \mathrm{Cont}_N \\ \downarrow{\scriptstyle [\![\cdot]\!]} & & \downarrow{\scriptstyle [\![\cdot]\!]} \\ \mathrm{Set}^N \to \mathrm{Set} & \xrightarrow{\overline{[\![S\lhd P]\!]}} & \mathrm{Set}^N \to \mathrm{Set} \end{array} \tag{2.2}$$

Hence, container extension lifts to a fully faithful functor embedding algebras over $E_{S\lhd P}$ into algebras over $\overline{[\![S \lhd P]\!]}$. In particular, container extension reflects initial algebras.

Recall that, in the specific categorical setting Set, the initial algebra of the strictly positive functor $\overline{S \lhd P}$ can be constructed using a transfinite chain

$$0 \xrightarrow{\ !\ } \overline{S \lhd P}(0) \xrightarrow{\overline{S\lhd P}(!)} \overline{S \lhd P}^2(0) \xrightarrow{\overline{S\lhd P}^2(!)} \dots$$

by taking colimits at limit ordinal steps. [10] By using commutativity of (2.2) at successor ordinal steps and colimit reflectivity of the extension functor at limit ordinal steps, it follows that the above chain is (equivalent to) the image under container extension of the corresponding chain over $E_{S\lhd P}$:

$$0 \xrightarrow{\ !\ } E_{S\lhd P}(0) \xrightarrow{E_{S\lhd P}(!)} E^2_{S\lhd P}(0) \xrightarrow{E^2_{S\lhd P}(!)} \dots\ .$$

By full faithfulness of container extension, this chain stabilized at the same ordinal as the first one. It follows that the extension functor reflects initial algebra formation as well. Purely in the category Cont_N, and in fact a basic instance of the semantics of induction-recursion, it can be further computed that

$$\mu E_{S\lhd P} = \mu U \lhd \mathrm{fold}_U\,(V_{\mathrm{Set}^N}\,\mathrm{id})\,,$$

where we have silently ignored the size issue $\mathrm{Set}^N : \mathrm{Set}$ in favour of elegant treatment. [11]

[10] This process stabilizes at the first regular ordinal larger than S.

[11] This is easily remediable by introducing V in a universe polymorphic fashion.

2.4.2 Interlude: Classification of Integrals of Certain Quotient Containers

Recall the notion of the derivative [2] of a container. For regular functors, this notion is a special case of our notion of biased derivative as was explained in the previous section. A natural question to ask is whether every container has an anti-derivative.

Gylterud [22] examined more closely the quotient containers of [3], themselves based on the work by Joyal [32] on combinatorial species. Recall that *quotient containers* generalize shapes from sets (in the sense of homotopy type theory) to groupoids, with the position family generalized to a functor from the shape groupoid to the category of sets. They have use in modelling unordered datatypes [3].

Gylterud showed that every analytic container — restricting the position functor to the target category of finite sets — has an anti-derivative. Concretely, an anti-derivative of the container $1 \lhd \lambda \bullet . [n]$ representing the functor $X \mapsto X^n$ is given by the *cyclic* quotient container $\mathbb{Z}/(n+1) \lhd C_{n+1}$ where the functor $C_{n+1} : \mathbb{Z}/(n+1) \to [n+1]$ denotes the cyclic action of the single-object groupoid $\mathbb{Z}/(n+1)$ on $[n+1]$. Observe here that quotient containers with groups for shapes are essentially the same as actions from group theory.

A natural question to ask is whether the cyclic quotient container itself has an anti-derivative in quotient containers. As we will see, the answer is negative. The core of the problems boils down to the following purely group theoretic exercise:

Theorem 2.2. *The transitive actions S on a finite set X of cardinality $n > 1$ such that, for each $x : X$, the action of S_x on $S \setminus \{x\}$ is isomorphic to the cyclic action of $\mathbb{Z}/(n-1)$ on itself are, up to isomorphism, given by the actions of the group* $\mathrm{Affine}(K)$ *of bijective affine transformations on a finite field K.* [12]

Proof. First of all, let K be a finite field. We have

$$\mathrm{Affine}(K) = \{z \mapsto a \cdot z + b \mid a, b \in K, a \neq 0\}.$$

It is clear that $\mathrm{Affine}(K)$ acts transitively on K since, for any $x, y \in K$, the map $z \mapsto z + (y - x)$ is bijective, affine, and maps x to y. By transitivity, $\mathrm{Affine}(K)_x$ is isomorphic to $\mathrm{Affine}(K)_0$ for any $x \in F$. Since

$$\mathrm{Affine}(K)_0 = \{z \mapsto a \cdot z \mid a \in X, a \neq 0\},$$

the isomorphism condition on $\mathrm{Affine}(K)_x$ is now equivalent to the well-known fact that $K^* := (K \setminus \{0\}, \cdot)$ is isomorphic to $\mathbb{Z}/(|K| - 1)$.

The rest of the proof is concerned with the much more involved reverse direction. Given a transitive subgroup S of the group of permutations on a set X of cardinality n, assume S_x is cyclic and acting transitively on $X \setminus \{x\}$, i.e. isomorphic to $\mathbb{Z}/(n-1)$, for $x \in X$.

Transitivity of S implies that there is $s \in S$ sending $x \in X$ to arbitrarily chosen $y \in X$. Since S is a group, the number $c_{x,y}$ of such elements of S does not depend on y. In particular, we have $c_{x,y} = c_x = |S_x| = n - 1$, and hence $|S| = \sum_{x \in X} c_x = n(n-1)$. Furthermore, since S_x is acting freely on $X \setminus \{x\}$, every element $s \in S$ with $s \neq \mathrm{id}$ has at most one fixpoint. By cardinality, it follows that there are exactly $n - 1$ fixpoint-free elements F of S.

Lemma 2.19. *If $s \in S$ commutes with an element $f \in F$, then $s = \mathrm{id}$ or $s \in F$.*

Proof. Assume that $s \in S_x$ for some $x \in X$. It follows that $s(f(x)) = (sf)(x) = (fs)(x) = f(x)$. Since $x \neq f(x)$, we deduce s has at least two fixpoints, showing $s = \mathrm{id}$. □

Let $f \in F$ be a fixpoint-free permutation of order $\mathrm{ord}(f) > 1$. If f had two cycles of different lengths $m, n > 1$, then $s^m \neq \mathrm{id}$ would have m fixpoints. Hence, s is the product of $\frac{n}{\mathrm{ord}(f)}$ disjoint cycles of length $\mathrm{ord}(f)$.

[12] An action S on a set X is called *transitive* if all $x, y \in X$ have $s \in S$ such that $s(x) = y$. It is called *free* if all $x, y \in X$ have at most one such s. For $x \in X$, the *stabilizer* S_x is the subgroup of those $s \in S_x$ such that $s(x) = x$.

Now, let $x \in X$ and consider the action of S_x on F by conjugation. Lemma 2.19 shows that this action is free. Since $|S_x| = |F|$, it is also transitive, meaning all $f, g \in F$ are conjugated, i.e. there is a unique $s \in S_x$ such that $f_s = g$. This implies $\mathrm{ord}(f) = \mathrm{ord}(g)$, i.e. all elements of F have the same order $p > 1$. Recall that $f \in F$ consists of $\frac{n}{p}$ disjoint cycles of length p. If p had a non-trivial factor q, then f^q would be a fixpoint-free permutation of order $\frac{p}{q} \neq p$, yielding a contradiction. Hence, p is prime. Note that, by freeness and transitivity of the applicative action of S_x on $X \setminus \{x\}$ and the conjugative action of S_x on F, there exists a unique $f_{x\to y} \in F$ such that $f_{x\to y}(x) = y$ for any different $x, y \in X$.

For $f \in F$, let $\langle f \rangle$ be the subgroup of S generated by f. By the above, $\langle f \rangle$ is isomorphic to $\mathbb{Z}/(p)$. Since it has $p - 1$ generators and $|F| = n - 1$, we deduce $|D| = \frac{n-1}{p-1}$ where $D = \{\langle f \rangle \mid f \in F\}$. For different $x, y \in X$, we denote $X_{x\to y}$ the orbit of x under the applicative action of $\langle f_{x\to y} \rangle$.

Given $x \in X$, let $T_x \subseteq S_x$ be the subgroup of those t that act invariantly by conjugation on $\langle f \rangle$ for some $f \in F$. Since conjugatively, S_x acts transitively on F and is abelian, this is equivalent to t being invariant on $\langle f \rangle$ for all $f \in F$. Since $\langle f \rangle$ has $p - 1$ generators and the conjugative action of S_x is free and transitive, we have $|T_x| = p - 1$. Since $T_x \subseteq S_x$ is cyclic, it must hence be isomorphic to $\mathbb{Z}/(p-1)$, or equivalently $(\mathbb{Z}/(p))^*$, the multiplicative group of the field $\mathbb{Z}/(p)$.

Lemma 2.20. *Fix $x \in X$ and $s \in S_x$ with $s \neq \mathrm{id}$. Consider $f : F$ and let Y be a cycle of f. Then s is invariant on Y precisely if $s \in T_x$.*

In other words, the subgroup of S of elements invariant on Y is the disjoint union of $\langle f \rangle$ and $T'_x := T_x \setminus \{\mathrm{id}\}$ for $x \in Y$.

Proof. Assume that $s \in T_x$. Since Y is an orbit of $\langle f \rangle$, we know $s(Y)$ is an orbit of $\langle f \rangle_s = \langle f \rangle$. Since $x \in Y$ and $s(x) = x$, note that Y and $t(Y)$ have a common element. Being cycles of $\langle f \rangle$, they must hence be equal.

Conversely, recall that s consists of $\frac{n-1}{\mathrm{ord}(s)}$ cycles of length $\mathrm{ord}(s)$ together with the trivial cycle $\{x\}$. Note that s is invariant on Y precisely if Y decomposes into a disjoint union of cycles of s. Since p and $n - 1$ are coprime, we know that $\mathrm{ord}(s) \geq 2$ is not a divisor of $|Y| = p$. Thus, we have $x \in Y$ and $\mathrm{ord}(u) \mid p - 1$, i.e. $z \in T_z$. □

Lemma 2.21. *The set $M := F \cup \{id\}$ is closed under inversion and multiplication, i.e. M is a subgroup of S.*

Proof. First of all, observe that M is closed under taking powers.

In the case $p = 2$, observe that F consists only of involutions. Let $f, g \in F$. If gf has a fixpoint $x \in X$, then $y := f(x) \neq x$ fulfills $x = g(y)$. Involuting, we get $x = f(y)$ and $y = g(x)$, so $g(f(y)) = y$, i.e. y is another fixpoint of gf, proving that $gf = \mathrm{id}$. Otherwise, gf is fixpoint-free, i.e. $gf \in F$.

The remainder of the proof of this lemma is devoted to the case $p > 2$.

Given $x \in X$ and $u \in T_x$, recall that conjugation with u induces an outer automorphism on $H \in D$. Since H is isomorphic to $\mathbb{Z}/(p)$, this action must correspond to exponentiation with some $q_x(u) \in (\mathbb{Z}/(p))^*$, i.e. mapping $f \in H$ to $f^{q_x(u)}$, where well-definedness follows from $f^p = \mathrm{id}$. Given any other $g \in F$, there exists $s \in S_x$ such that $g_{s^{-1}} \in H$ as S_x acts transitively on F. Commutativity of $u, s \in S_x$ then implies $g_u = g_{u_s} = ((g_{s^{-1}})_u)_s = ((g_{s^{-1}})^{q_x(u)})_s = g^{q_x(u)}$, justifying the independence of $q_x(u)$ from the choice of H and proving that $q_x : T_x \to (\mathbb{Z}/(p))^*$ is a group homomorphism.

Let $x, y \in X$ be given. For $u \in T_x$ and $v \in T_y$, we have $vu^{-1} \in M$ if and only if $q_x(u) = q_y(v)$. For the first direction, observe that $q_x(u) = q_y(v)$ implies the conjugative action of vu^{-1} on F is trivial and apply Lemma 2.19. For the other direction, assume $vu^{-1} \in F$ and observe that vu^{-1} has trivial conjugative action on $\langle vu^{-1} \rangle$, i.e. $q_x(u)q_y(v^{-1}) = 1$, i.e. $q_x(u) = q_y(y)$.

Now, let $f, g \in F$ be given. Choose $y \in X$ and an element v of T'_y. Note that this makes use of the assumption $p > 2$. Observe that $u := fv$ is invariant on $X_{y\to f(y)}$. By Lemma 2.20 and since $fv \notin \langle y \to f(y) \rangle$ (since the right-hand side is generated by f and hence a subset

of M), we must have $u \in T'_x$ for some $x \in X_{y \to f(y)}$. Similarly, define $w := gv \notin \langle y \to g(y) \rangle$ and $z \in X_{y \to g(y)}$ such that $w \in T'_z$. By transitivity, if $q_x(u) = q_y(v)$ and $q_z(w) = q_y(v)$, then $q_x(u) = q_z(w)$. Under the equivalence from two paragraphs above, this translates into $uv^{-1} \in M$ and $wv^{-1} \in M$ implying $wu^{-1} \in M$. Since $uv^{-1} = f \in M$ and $wv^{-1} = g \in M$, we thus have $gf^{-1} = wu^{-1} \in M$. □

Since all elements of M have order 1 or p, we know M is a p-group. In particular, there is a center element $f \in Z(M)$ with $f \neq \mathrm{id}$, i.e. $f \in F$. Choose any $x \in X$. Since the conjugative action of S_x is transitive on F, this implies $F \subseteq Z(M)$, i.e. M is abelian. Now, fix arbitrary but different elements $0, 1 \in X$. Given $x, y \in M$, define $x + y := (fg)(0) = f(y) = g(x)$ with unique $f, g \in M$ such that $x = f(0)$ and $y = g(0)$. Since M is an abelian group, so is $(X, +)$. Given $x, y \in M$, define $x \cdot y := 0$ if $x = 0$ or $y = 0$, and $x \cdot y = (uv)(1) = u(y) = v(x)$ with unique $u, v \in S_0$ such that $x = u(1)$ and $y = v(1)$. Since S_0 is an abelian group, so is $(X \setminus \{0\}, \cdot)$. Finally, given $x, y, z \in X$ with $x \neq 0$, choose $u \in S_0$ such that $x = u(1)$ and $f \in M$ such that $y = f(0)$. We derive $x \cdot y + x \cdot z = u(1) \cdot f(0) + u(1) \cdot z = u(f(0)) + u(z) = f_u(0) + u(z) = f_u(u(z)) = u(f(z)) = u(f(0) + z) = u(1) \cdot (f(0) + z) = x \cdot (y + z)$, proving distributivity and showing that $(X, +, \cdot)$ is a field.

Let $x \in X$ and $u \in S_x$ be given. These is a unique $f \in M$ such that $x = f(0)$. We have $u_{f^{-1}} \in S_0$, hence $u_{f^{-1}}$ acts on X by multiplication with $y := u_{f^{-1}}(1) \neq 0$. For any $z \in X$, we now have $u(z) = f(u_{f^{-1}}(f^{-1}(z))) = y \cdot (z - x) + x = y \cdot z + (x - y \cdot x)$, proving that u acts on X as an affine transformation. Similarly, given $f \in M$, we have $f(z) = z + f(0)$ for any $z \in X$, again an affine transformation. Conversely, given any affine transformation $z \mapsto x \cdot z + y$ with $x \neq 0$, choose $u \in S_0$ such that $u(1) = x$ and $f \in M$ such that $f(0) = y$. For any $z \in X$, we have $(fu)(z) = f(u(z)) = f(x \cdot z) = x \cdot z + y$. Because $u(z) = z$ for all $z \in X$ implies $u = \mathrm{id}$ for $u \in S$, this proves the action S on X is isomorphic to the action of the group of bijective affine transformations on a finite field. □

Since finite fields exist only for cardinalities p^k with p prime and $k > 0$, with a unique instance $\mathbb{F}_{p^k}$ for each combination of p and k, we easily deduce the following corollary.

Corollary 2.5. *Given $n > 0$, the cyclic quotient container $\mathbb{Z}/(n) \lhd \lambda \bullet . [n]$, i.e. the additive action of $\mathbb{Z}/(n)$ on itself, has an anti-derivative only for $n = p^k - 1$ with p prime and $k > 0$. This anti-derivative is unique up to isomorphism and consists of the action of* $\mathrm{Affine}(F_{p^k})$ *on* $\mathbb{F}_{p^k}$.

Proof. Given a quotient container $S \lhd P$, recall that its derivative $S' \lhd P'$ can be constructed by letting $S' = \int^{s:S} P(s)$ and $P'(s, p) = P(s) - \{p\}$. Assume that $S' \lhd P'$ is isomorphic to $\mathbb{Z}/(n) \lhd \lambda \bullet . [n]$. Without changing the derivative of $S \lhd P$, we may remove those objects $s : S$ such that $P(s)$ is empty.

Since $\int^{s:S} P(s)$ was isomorphic to a group, it follows that all objects of S must be isomorphic. Switching to its skeleton, we may hence assume it is a group, i.e. a groupoid with a single object $s : S$. Furthermore, for any two positions $p, q : P(s)$, there must be an isomorphism $f : s \to s$ such that $P(f)(p) = q$. The group action P is thus transitive.

Let us further fix $p : P(s)$. We have seen that the derivative of $S \lhd P$ is isomorphic to $G \lhd \lambda \bullet . P(s) - \{p\}$ where G consists of those isomorphisms $f : s \to s$ such that $f(p) = p$. By assumption, this action is isomorphic to the cyclic action of length n.

The prerequisites of Theorem 2.2 are thus fulfilled, yielding the expected conclusion. □

Late during the writing-up phase, we stumbled upon work by Rajan [49] that consists of a complete classification of all higher anti-derivatives, i.e. not just the first two levels, of the combinatorial species corresponding to the functor mapping X to X^n. We suspect that the differences between combinatorial species and (possibly symmetric) quotient containers are inconsequential enough for these arguments to transfer to our case.

2.4.3 Containers Fiberwise

Using the closure properties established above, it is easily seen that every regular functor is represented by a container. What is more, the family of positions will always be valued in finite sets. In this setting, the fibred presentation of containers becomes practical.

An *N-ary power series* $C : \mathrm{PowSeries}_N$ is a family $C : \mathbb{N}^N \to \mathrm{Set}$ of *coefficients*. Such a power series C may be interpreted as an N-ary container $\sum_{k:\mathbb{N}^N} C(k) \lhd \lambda(k, c).[k]$ where we abbreviate $[k] = \{0, \ldots, k-1\} : \mathrm{Set}$ for the embedding $\mathbb{N} \to \mathrm{Set}$. [13] Reversely, given a *finitary* container $S \lhd P$, i.e. where $P : S \to \mathrm{FinSet}^N$, we may refactor

$$\begin{aligned}
[\![S \lhd P]\!](X) &= \sum_{s:S} \prod X^{Ps} \\
&= \sum_{k:\mathbb{N}^N} \sum_{\substack{s:S,\\ |Ps|=k}} \prod X^k \\
&= \sum_{k:\mathbb{N}^N} \{s : S \mid |Ps| = k\} \times \prod X^k
\end{aligned}$$

to derive a power series representation of $S \lhd P$ with coefficients $C : \mathbb{N}^N \to \mathrm{Set}$ given by the fibers of S over P:

$$C(k) = \{s : S \mid |Ps| = k\}.$$

This equivalence, with in fact is just a discrete version of the Grothendieck construction, enables us to identity N-ary finitary containers with N-ary power series. The corresponding categorical structure of power series has morphisms between $C_1, C_2 : \mathrm{PowSeries}_N$ given by

$$\mathrm{PowSeries}_N(C_1, C_2) = \prod_{k:\mathbb{N}^N} C_1(k) \to \sum_{h:\mathbb{N}^N} C_2(h) \times ([h] \to [k]).$$

Cartesian morphisms between C_1 and C_2 correspond to families

$$\prod_{k:\mathbb{N}^N} C_1(k) \to C_2(k) \times ([h] \simeq [h]).$$

We call such a cartesian morphism *order-preserving* if the bijection component above is always the identity. [14] Under this change of representation, the container extension corresponds to the *power series extension* functor $[\![\cdot]\!] : \mathrm{PowSeries}_N \to (\mathrm{Set}^N \to \mathrm{Set})$ defined by

$$[\![C]\!](X) = \sum_{k:\mathbb{N}^N} C(k) \times \prod X^k,$$

justifying the naming.

Let us rephrase the already developed properties for containers as restricted to finitary containers in terms of power series.

Colimits of order-preserving diagrams Given a diagram $D : \mathcal{I} \to \mathrm{PowSeries}_N$ of power series such that for a morphism $f : i \to j$ in $\mathcal{I}$, the morphism $D(f) : D_i \to D_j$ is cartesian and order-preserving, note that f simply corresponds to a family $f_k : D_i(k) \to D_j(k)$ for $k : \mathbb{N}^N$. The colimit over D is thus computed coefficientwise: given $k : \mathbb{N}^N$, we have

$$(\mathrm{colim}\, D)(k) = \mathrm{colim}_{i:\mathcal{I}}\, D_i(k).$$

[13] In other words, we identify $\mathbb{N}$ with a skeleton of FinSet.

[14] Note that, in certain contexts, this may depend on the precise way finite sets are identified with natural numbers, albeit we will only use it in the coherent way where the same chosen identification isomorphism appears on both sides of the bijection $[h] \simeq [h]$, making the notion invariant under its choice.

Coproducts The previous paragraph includes the special case of coproducts. Given a family of power series C_i : $\mathrm{PowSeries}_N$ with $i : I$, its coproduct is computed coefficientwise: given $k : \mathbb{N}^N$, we have

$$\left(\sum_{i:I} C_i\right)(k) = \sum_{i:I} C_i(k).$$

Finite products Given power series $C_1, \ldots, C_m$, we have

$$\left(\prod_{j<m} C_j\right)(k) = \sum_{\substack{k_j:\mathbb{N}^N \text{ for } j<m,\, j<m \\ \sum_j k_j = k}} \prod_{j<m} C_j(k_j).$$

where $k : \mathbb{N}^N$. Note that this construction stops working for infinite products.

Variable selection The variable selection functor $\pi_n^N : \mathrm{Set}^N \to \mathrm{Set}$ with $n : N$ is represented by the N-ary power series C where, for $k : \mathbb{N}^N$, we have

$$C(k) = \begin{cases} 1 & \text{if } [k] = e_i \\ 0 & \text{else.} \end{cases}$$

Again, we will reuse the identifier π_n^N for C.

Composition Given an N-ary power series C with N finite and M-ary power series D_n for $n : N$, let us represent the composition

$$[\![C]\!] \circ \langle [\![D_n]\!] \rangle_{n:N}$$

as a power series. Now that we already have the above ingredients of sums and finite products of power series, we may simply state it is represented by

$$C \circ \langle D_n \rangle_{n:N} = \sum_{k:\mathbb{N}^N} C(k) \times \prod_{n:N} D_n^{k_n}.$$

This recasts composition in terms of a version of the extension functor generalized from target Set to target M-ary containers. Composition is functorial in both C and the family D. The functorial action preserves cartesian and order-preserving morphisms.

Parametric initial algebras Fix an $(N+1)$-ary power series C with N finite. Let E_C denote the endofunctor on $\mathrm{PowSeries}_N$ that acts by postcomposition

$$E_C(D) = C \circ \left\langle \left\langle \pi_n^N \right\rangle_{n:N}, D \right\rangle.$$

Again, we have the following commutative diagram:

$$\begin{array}{ccc} \mathrm{PowSeries}_N & \xrightarrow{E_C} & \mathrm{PowSeries}_N \\ \downarrow{\scriptstyle [\![\cdot]\!]} & & \downarrow{\scriptstyle [\![\cdot]\!]} \\ \mathrm{Set}^N \to \mathrm{Set} & \xrightarrow{\overline{[\![C]\!]}} & \mathrm{Set}^N \to \mathrm{Set} \end{array}$$

and the parametric initial algebra of C with respect to its last argument is computed via the chain

$$0 \xrightarrow{!} E_C(0) \xrightarrow{E_C(!)} E_C^2(0) \xrightarrow{E_C^2(!)} \cdots .$$

Since $\overline{[\![C]\!]}$ is an ω-cocontinuous functor (being composed only of variable selection, products, and sums, all of which are left adjoints), the colimit of this ω-chain will already be the initial algebra of E_C. Furthermore, since $! : 0 \to E_C 0$ is order-preserving and the functorial action of E_C preserves order-preserving morphisms, note that the colimit will be over a diagram of order-preserving morphisms. This implies that we may compute the initial algebra of E_C coefficientwise: given $k : \mathbb{N}^N$, we have

$$(\mu E_C)(k) = \mathrm{colim}_{i:\omega}(E_C^i 0)(k).$$

2.4.4 Power Series of Guarded Functors

By the above properties, it follows that regular functors are represented not only by containers, but more definitely by power series. For a regular functor F of arity N, let $C(F) : \mathbb{N}^N \to \mathrm{Set}$ denote the power series associated to its interpretation over Set. From now on, we will use the more familiar algebraic notation, denoting $C(F) : \mathrm{Set}\,[\![X^N]\!]$ instead. Furthermore, let the categorical structure on Set $[\![X^N]\!]$ be given by only the order-preserving cartesian-morphisms. Note that the variables X are purely synthetic, a notational convenience for speaking about the variable selector power series.

To unlock the full algebraic prowess of power series, we would need the coefficients of $C(F)$ to live in some well behaved ring-like algebraic object like the integers. This is precisely the motivation for the definition of guarded functors.

But first, let us introduce some notation. Given $\alpha : \mathbb{N}^N$, we have a *degree* function

$$\begin{aligned} \deg_\alpha \quad &: \mathrm{Set}\,[\![X^N]\!] \to \{-\infty\} \cup \mathbb{N} \cup \{\infty\}, \\ \deg_\alpha(C) &= \sup\left\{\alpha \bullet k \mid k : \mathbb{N}^N, C(k) \neq \emptyset\right\}. \end{aligned}$$

Let $\deg_I = \deg_{\theta_I}$ where $\theta_I : N \to \mathbb{N}$ is the characteristic function of $I \subseteq N$. Further, let $\deg_n = \deg_{\{n\}}$ for $n : N$. With these conventions, the *total degree* function is defined as $\deg = \deg_N$. Note that degree functions are multiplicative, and that

$$\deg_\alpha(C + D) = \max\left(\deg_\alpha(C), \deg_{\alpha(D)}\right).$$

Note further that the degree functions are linear in the parameter α.

Of course, we also have the standard degree functions on algebraic structures $R[X^N]$ or $R\,[\![X^N]\!]$ with a ring R. We will use the same notation for this case. Note, however, that for addition we only have the inequality

$$\deg_\alpha(P + Q) \leq \max\left(\deg_\alpha(P), \deg_{\alpha(Q)}\right)$$

since leading coefficients might annihilate each other.

Given $\alpha : \mathbb{N}^N$, we have a *truncation* functor

$$\begin{aligned} |\cdot|_\alpha^{<\cdot} \quad &: \mathbb{N} \to \mathrm{Set}\,[\![X^N]\!] \to \mathrm{Set}\,[\![X^N]\!]_\alpha^{<\cdot}, \\ |C|_t^{<k} &= \begin{cases} C(K) & \text{if } \alpha \bullet k < t, \\ 0 & \text{else.} \end{cases} \end{aligned}$$

Here, note that Set $[\![X^N]\!]_\alpha^{<k}$ denotes the full subcategory of those power series $C : \mathrm{Set}\,[\![X^N]\!]$ such that $\deg_\alpha(C) < k$. We use similar conventions for variants of the truncation functor as we did for the degree function. This truncation functor preserves colimits of diagrams with order-preserving cartesian morphisms since they are computed coefficientwise, and thus in particular arbitrary coproducts.

Consider a ring R, a collection of elements $X : R^I$, and $d : \mathbb{N}$. Let us introduce the abbreviations

$$X^d = \{X^\alpha\}_{|\alpha|=d} = \left\{\prod_{i:I} X_i^{\alpha_i}\right\}_{\alpha:\mathbb{N},|\alpha|=d}$$

for ease of executing modulo calculations. With respect to this convention, modulo X is to be understood as standing for modulo $(X_i)_{i:I}$.

Recall that the rings $\mathbb{Z}[X^I]$ and $\mathbb{Q}[X^I]$ form unique factorization domains for any (finite) variable index set I.

Given a field extension $K \subseteq L$, an element x of L is called *algebraic* if there is an irreducible monic polynomial $P : K[X]$ such that $P(x) = 0$. With respect to the stated normalization constraint of being monic and irreducible, the polynomial P is in fact unique. The set L_{alg} of algebraic elements of L forms a subfield of L, yielding the series $K \subseteq L_{\mathrm{alg}} \subseteq L$ of field extensions.

Given a field K and a finite set I, the field of fractions of $K[\![X^I]\!]$ is known as the *formal Laurent series* L_K over K. Just like we had inclusions $K \subseteq K[X^I] \subseteq K[\![X^I]\!]$, we have inclusions $K \subseteq K(X^I) \subseteq L_K$. We will keep the inclusion $K[\![X^I]\!] \subseteq L_K$ implicit, calling an element of $K[\![X^I]\!]$ algebraic if it is algebraic as an element of L_K over the subfield $K(X^I)$. Since $K[\![X^I]\!]$ is a ring, the subset of algebraic elements of $K[\![X^I]\!]$ also forms a ring $K[\![X^I]\!]_{\mathrm{alg}}$. The below development should be read under the tagline of the algebraic power series $K[\![X^I]\!]$ being the Henselization of $K[X^I]$ localized at the origin, i.e. adjoining formal inverses of elements non-zero modulo X.

Theorem 2.3. *Consider a regular functor $F : \mathrm{Set}^{I+J} \to \mathrm{Set}$ guarded over $I \hookrightarrow I + J$. There is a power series $q : \mathbb{N}[Y^J][\![X^I]\!] \subseteq \mathbb{Q}(Y^J)[\![X^I]\!]$ algebraic over $\mathbb{Q}(Y^J, X^I)$ such that the power series representation of F is given by the image of q under the embedding $j : \mathbb{N}[Y^J][\![X^I]\!] \hookrightarrow \mathrm{Set}[X^I, Y^J]$.*

If in addition the functor F is shielded over I, then $q = 0 \mod (X_i)_{i:I}$.

Proof. By simultaneous induction on guardedness and shieldedness as a mutual family.

The guarded case If $F = \pi^{I+J}_{\mathrm{inl}(i)}$ with $i : I$, we simply choose $q = X_i$, which is the unique zero of the linear polynomial $Q = Z - X_i \in \mathbb{Q}(Y^J, X^I)[Z]$. Note that $\deg(q) = 1$.

Similarly, if $F = \pi^{I+J}_{\mathrm{inr}(j)}$ with $j : J$, we choose $q = Y_i$, which is the unique zero of the linear polynomial $Q = Z - Y_i \in \mathbb{Q}(Y^J, X^I)[Z]$. Note that $\deg(q) = 0$.

The cases of finite coproducts and products are rather straightforward. Consider algebraic power series $q_t : \mathbb{N}[Y^J][\![X^I]\!]$ for $t < k$. First note that the embedding j forms a ring homomorphism. It thus suffices to verify that $\sum_{t<k} q_t$ and $\prod_{t<k} q_t$ are algebraic. But as was explained previously, we know that $\mathbb{N}[Y^J][\![X^I]\!]_{\mathrm{alg}}$ forms a ring.

Finally, consider the initial algebra case $F(X) = \mu Y.\, G(X, Y)$ with $G : \mathrm{Set}^{I+J+1} \to \mathrm{Set}$ shielded over $I \hookrightarrow I + J + 1$. By induction hypothesis, there is $r : \mathbb{N}[Y^J, Z][\![X^I]\!]$ algebraic over $Q(Y^J, Z)(X^I)$ representing G under the embedding $\mathbb{N}[Y^J, Z][\![X^I]\!] \hookrightarrow \mathbb{N}[X^I, Y^{J+1}]$ sending Y_j to $Y_{\mathrm{inl}(j)}$ for $j : J$ and Z to $Y_{\mathrm{inr}(\bullet)}$. By shieldedness, we furthermore have $r = 0 \mod (X_i)_{i:I}$. By multiplying with the least common multiple of denominators of occurring fraction, we may construct an irreducible polynomial $R : \mathbb{Z}[Y^J, Z][X^I][W]$ which has r as a zero.

Let F be represented by a power series $C : \mathrm{Set}[X^I, Y^J]$. Since F is the parametric initial algebra of G, recall that C may be constructed as the colimit of the ω-chain

$$0 \xrightarrow{\quad ! \quad} E_r(0) \xrightarrow{\quad E_r(!) \quad} E_r^2(0) \xrightarrow{\quad E_r^2(!) \quad} \cdots \tag{2.3}$$

of order-preserving cartesian morphisms where

$$\begin{aligned} E_r(D) &: \ \mathrm{Set}[\![X^I, Y^J]\!] \to \mathrm{Set}[\![X^I, Y^J]\!], \\ E_r(D) &= r \circ \left\langle \left\langle \pi_J^{I+J}, D \right\rangle, \pi_I^{I+J} \right\rangle. \end{aligned}$$

For fixed $d : \mathbb{N}$, note that

$$|E_r(D)|_I^{<d+1} = \left| r \circ \left\langle \left\langle \pi_J^{I+J}, |D|_I^{<d} \right\rangle, \pi_I^{I+J} \right\rangle \right|_I^{<d+1}$$

since $\deg_I(r) \geq 1$ by shieldedness assumption. We hence have a lift in the following strictly commutative square of functors:

$$\begin{array}{ccc} \mathrm{Set}\left[\!\left[X^I, Y^J\right]\!\right]_I^{<d} & \xrightarrow{|E_r|_I^{<d}} & \mathrm{Set}\left[\!\left[X^I, Y^J\right]\!\right]_I^{<d} \\ {\scriptstyle |\cdot|_I^{<d}}\uparrow & \overset{F_d}{\searrow} & \uparrow{\scriptstyle |\cdot|_I^{<d}} \\ \mathrm{Set}\left[\!\left[X^I, Y^J\right]\!\right]_I^{<d+1} & \xrightarrow{|E_r|_I^{<d+1}} & \mathrm{Set}\left[\!\left[X^I, Y^J\right]\!\right]_I^{<d+1} \end{array}$$

Note that the upper triangle commutes since the lower triangle and the outer square commute and the vertical projections are full and essentially surjective.

What follows is just a standard application of any of a variety of fixpoint theorems for contractive functions. With our lift, we may write

$$\left|E_r^{k+1}(0)\right|_I^{<d+1} = |E_r|_I^{<d+1}\left(\left|E_r^k(0)\right|_I^{<d+1}\right) = F_d\left(\left|E_r^k(0)\right|_I^{<d}\right)$$

and

$$\left|E_r^{k+1}(!)\right|_I^{<d+1} = |E_r|_I^{<d+1}\left(\left|E_r^k(!)\right|_I^{<d+1}\right) = F_d\left(\left|E_r^k(!)\right|_I^{<d}\right)$$

for $k, d : \mathbb{N}$.

Since $\mathrm{Set}\left[\!\left[X^I, Y^J\right]\!\right] \mod (X^\alpha)_{|\alpha|=0}$ is the terminal category, observe that $\left|E_r^k(0)\right|_I^{<0} = 0$ and $\left|E_r^k(!)\right|_I^{<0} = \mathrm{id}$ for all $k : \mathbb{N}$. By induction on $d : \mathbb{N}$, we see that $\left|E_r^{k+d}(0)\right|_I^{<d} = \left|E_r^d(!)\right|_I^{<d}$ and $\left|E_r^{k+d}(!)\right|_I^{<d} = \mathrm{id}$ for all $k : \mathbb{N}$. For a given $d : \mathbb{N}$, the chain (2.3) thus stabilizes after d iterations modulo $(X^\alpha)_{|\alpha|=d}$. Since colimits of order-preserving diagrams are computed coefficientwise, we thus have $C(k) = E_r^{|k|+1}(0)(k)$ for $k : \mathbb{N}^I$.

Note that E_r preserves the property of being polynomial in Y^J:

$$\begin{array}{ccc} \mathbb{N}[Y^J]\left[\!\left[X^I\right]\!\right] & \xrightarrow{r[Z=\cdot]} & \mathbb{N}[Y^J]\left[\!\left[X^I\right]\!\right] \\ \downarrow{\scriptstyle j} & & \downarrow{\scriptstyle j} \\ \mathrm{Set}\left[\!\left[X^I, Y^J\right]\!\right] & \xrightarrow{E_r} & \mathrm{Set}\left[\!\left[X^I, Y^J\right]\!\right] \end{array}$$

where $r[Z = \cdot] : \mathbb{N}[Y^J]\left[\!\left[X^I\right]\!\right][Z]$. Defining $q : \mathbb{N}[Y^J]\left[\!\left[X^I\right]\!\right]$ by setting

$$q(k) = \left(r[Z = \cdot]^{|k|+1}(0)\right)(k)$$

for $k : \mathbb{N}^I$, we hence have $j(q) = C$ by coefficientwise comparison.

Recalling Lambek's Lemma, we have $E_r(C) = C$ via an order-preserving cartesian isomorphism. By faithfulness of the embedding j, it follows that $r[Z = q] = q$ strictly since isomorphic objects in $\mathbb{N}[Y^J]\left[\!\left[X^I\right]\!\right]$ are strictly equal.

Applying R to the value Z from its underlying coefficient ring, note that the result may be regarded as a polynomial $R(Z) : \mathbb{Z}[Y^J][X^I][Z]$ in Z. If $R(Z) = 0$, then $R = W - Z$ since R was assumed irreducible. But then $r = Z$, violating $r = 0 \mod (X_i)_{i:I}$. So we have $R(Z) \neq 0$.

By a series of structural transformations, we see that

$$\begin{aligned} R(Z)(q) &= R(Z)[Z = q] \\ &= R[Z = q](q) \\ &= R[Z = q](r[Z = q]) \\ &= R(r)[Z = q] \\ &= 0[Z = q] \\ &= 0, \end{aligned}$$

crucially exploiting that $r[Z = q] = q$. Therefore, we can be certain that q is algebraic.

However, the minimal polynomial of q should be effectively computable. [15] Factorize [35] $R(Z) = cQ_0^{\epsilon_0} \dots Q_{k-1}^{\epsilon_{k-1}}$ with $Q_t : \mathbb{Q}(Y^J)(X^I)[Z]$ and $\epsilon_t \geq 1$ for $t < k$ and $c : \mathbb{Q}(Y^J)$ such that the bases Q_t are monic, irreducible, and pairwise distinct.

Consider distinct $u, v < k$. Since $Q(Y^J)(X^I)[Z]$ is a Euclidean domain, there are $P_u, P_v : \mathbb{Q}(Y^J)(X^I)[Z]$ such that $P_u Q_u + P_v Q_v = 1$. Multiplying with the denominators, we compute polynomials $U, V : \mathbb{Q}(Y^J)[X^I][Z]$ and $W : \mathbb{Q}(Y^J)[X^I]$ such that $UP_u + VP_v = W$ and $W \neq 0$. Let $d = \deg_X(W) + 1$. We can compute $(UP_u)(q)$ and $(VP_v)(q)$ modulo X^d. Note that this computation depends only on the first d coefficients of q. Since $W(q) \neq 0 \mod X^d$, one of them must be non-zero.

For each pair $u, v < k$ with $u \neq v$, we can determine $w \in \{u, v\}$ such that $Q_w(q) \neq 0$. By straightforward combinatorial reasoning, this means we can find $s < k$ such that $Q_t(q) \neq 0$ for $t < k$ with $t \neq s$. Since $R(q) = 0$, we then must have $Q_s(q) = 0$. We have thus computed the minimal polynomial of q.

The shielded case Given algebraic power series $q_i : \mathbb{N}[Y^J] \llbracket X^I \rrbracket$ for $i : I$, we of course have $\sum_{i:I} X^i q_i = 0 \mod (X_i)_{i:I}$. □

Let $Q_F : \mathbb{Q}(X^N)[Z]$ denote the minimal polynomial associated with a guarded functor F of arity N as calculated by executing the algorithm detailed in the above lemma.

A silly example As a simple and somewhat silly example, let us decide whether

$$\mathrm{List}(X \times Y)(\mathrm{List}(X) + \mathrm{List}(Y)) = \mathrm{List}(X \times Y) + \mathrm{List}(X) \times \mathrm{List}(Y)$$

natural in $X, Y : \mathrm{Set}$. Let us abuse notation and use functor identifiers themselves to stand for their associated power series in $\mathbb{Z} \llbracket X, Y \rrbracket$. Executing the construction detailed in the proof of the previous lemma, we see for any $P : \mathbb{N}[X, Y]$ that $(1 - P)Z - 1 \in \mathbb{Z}[X, Y][Z]$ is the minimal polynomial for the power series $\mathrm{List}(P)$: the equation

$$\mathrm{List}(P) = \frac{1}{1 - P}$$

is well-known from the theory of generating functions. This explains why this example is silly: since all minimal polynomials are linear, the involved power series already live in the base field $\mathbb{Q}(X, Y)$. Hence, the algebraic machinery developed above is not really being used. We calculate

$$\begin{aligned}
&\mathrm{List}(X \times Y)(\mathrm{List}(X) + \mathrm{List}(Y)) - \mathrm{List}(X \times Y) \\
&= \frac{1}{(1 - XY)(1 - X)} + \frac{1}{(1 - XY)(1 - Y)} - \frac{1}{1 - XY} \\
&= \frac{1}{1 - XY} \cdot \frac{(1 - Y) + (1 - X) - (1 - X)(1 - Y)}{(1 - X)(1 - Y)} \\
&= \frac{1}{1 - XY} \cdot \frac{1 - XY}{(1 - X)(1 - Y)} \\
&= \frac{1}{(1 - X)(1 - Y)}
\end{aligned}$$

[15] Finding a minimal polynomial for some zero of a given polynomial tends to be a computationally inefficient process. What we can do very easily is bring it into a square-free and hence separable form.

For the purpose of comparing (the polynomials associated to) given finitiary inductive types so as to decide isomorphism, it should not actually be necessary to start with minimal polynomials: just compute the greatest common divisor D of the two given separable polynomials PD, QD using the Euclidean domain structure, follow the below process to check whether both types are still zeroes of D, and then compare sufficiently many initial coefficients so as to distinguish the roots of D using Lemma 2.22 and separability of D.

In other words, minimality assumptions may be replaced by separability assumptions, a computationally more efficiently tractable concept in characteristic zero.

$$= \mathrm{List}(X) \times \mathrm{List}(Y).$$

Arriving at a finitary description The advantage of calculating a minimal polynomial for the power series representation associated to each guarded regular functor is that it yields a finitary description of a formally infinite object. One may now ask whether we are already done, with the minimal polynomial being a complete description. However, an irreducible polynomial with a zero in a given extension over the base field has in general multiple and distinct zeroes in that extension.

A conrete example, showing that multiple distinct zeroes may in fact all have non-negative coefficients and thus live in $\mathbb{N}[X]$, is as follows. Consider the polynomial

$$\begin{aligned} Q &= \prod_{i,j\in\{0,1\}} Z - \left(1 - \frac{(-1)^i\sqrt{1-8X} + (-1)^j\sqrt{1-4X}}{2}\right) \\ &= \left(Z^2(Z^2-1) + 6XZ^2 + X^2\right)[Z = Z-1] \\ &= Z^4 - 4Z^3 + (5+6X)Z^2 - 2(1+6X)Z + 6X(X+1) \end{aligned}$$

which we conveniently stated in factorized form (which in fact is how it was constructed in the first place). Let us develop two of its zeroes as power series in X using e.g. Henselization:

$$\begin{aligned} 1 - \frac{\sqrt{1-8X} + \sqrt{1-4X}}{2} &= 3X + 5X^2 + 18X^3 + 85X^4 + 462X^5 + \ldots, \\ 1 - \frac{\sqrt{1-8X} - \sqrt{1-4X}}{2} &= 1 + X + 3X^2 + 14X^3 + 75X^4 + 434X^5 + \ldots \end{aligned}$$

An standard argument about the coefficients in the Taylor series developments of $\sqrt{1-4X}$ and $\sqrt{1-8X}$ makes rigorous the intuition that all coefficients in the above series are non-negative. Since 1, $\sqrt{1-4X}$, and $\sqrt{1-8X}$ are linearly independent over $\mathbb{Q}(X)$, we see by a degree argument that Q is in fact irreducible. This makes Q the minimal polynomial associated to the two shown power series in X, lending palpable credence to the claim the minimal polynomial does not uniquely determine a power series with non-negative coefficients.

For a power series $q : \mathbb{Q}[\![X^I]\!]$ with minimal[16] polynomial $Q : \mathbb{Q}(X^I)[Z]$ such that $Q'(q) \neq 0 \mod (X)$, the typical Henselization argument shows that q is determined as a zero of Q by its first coefficient. Note that the polynomial from the example above still did not fulfill this assumption. In general, we have to deal with degenerate cases where $Q'(q) \in (X^d)$ for large $d : \mathbb{N}$. However, since $\mathbb{Q}(X^I)$ has characteristic zero, we can be certain that $Q'(q) \neq 0$, i.e. there must be a $d : \mathbb{N}$ such that $Q'(q) = 0 \mod (X^d)$, but $Q'(q) \neq 0 \mod (X^{d+1})$. In this scenario, we can apply the following lemma, justifying that supplying a certain prefix of coefficients of computable length will always determine a power series from its minimal polynomial.

Lemma 2.22. *Let K be a field and I a finite variable index set. Consider a monic polynomial $P : K[X^I][Z]$ and fix $d : \mathbb{N}$. For any starting value $S : K[X^I]/(X^{d+1})$ such that $P(S) = 0 \mod (X^{d+1})$ and $P'(S) = 0 \mod (X^d)$, but $P'(S) \neq 0 \mod (X^{d+1})$, there is at most a single power series $q : K[\![X^I]\!]$ extending S, i.e. $q = S \mod (X^d)$, such that $P(q) = 0$. This power series, if it exists, is effectively computable.*

Proof. Analogous to the proof of Hensel's Lemma [13]. We will calculate the coefficients of q step by step.

For $\beta : \mathbb{N}^I$, let $A_\beta \subseteq K[\![X^I]\!]$ denote the ideal generated by $X^{|\beta|+1}$ together with X^α for $|\alpha| = |\beta|$ and $\alpha > \beta$ with respect to the lexicographical comparison with induced by an arbitrary total order on I. With this, we may write

$$P'(S) = X^\beta U + V$$

[16] As per the previous footnote, minimality may be weakened to separability.

with $U : K[X^I]$ and $V \in A_\beta$ such that $U \neq 0 \mod X$ and $|\beta| = d$.

Let $n \geq d+1$ be given and assume that we have already found a unique $Q_n : K[\![X^I]\!]$ with $\deg(Q_n) < n$ such that $Q_n = S \mod (X^{d+1})$ and $P(Q_n) = 0 \mod (X^{n+d})$. Introduce $Q_{n+1} : K[\![X^I]\!]$ with $\deg_X(Q_{n+1}) < n+1$ where $Q_{n+1} = Q_n + \sum_{|\alpha|=n} X^\alpha T_\alpha$ with $T_\alpha : K$ for $\alpha : \mathbb{N}^I$ with $|\alpha| = n$. Let us show that Q_{n+1} is determined uniquely under the constraint $P(Q_{n+1}) = 0 \mod (X^{n+d+1})$, i.e. there is at most a single such choice for the family T. We have

$$\begin{aligned}
0 &= P(Q_{n+1}) \\
&= P\left(Q_n + \sum_{|\alpha|=n} X^\alpha T_\alpha\right) \\
&= \sum_{i<d} P^{(i)}(Q_n) \sum_{\substack{|\alpha_j|=n \\ \text{for } j<i}} X^{\sum_{j<i}\alpha_j} \prod_{j<i} T_{\alpha_j} \\
&= P(Q_n) + P'(Q_n) \sum_{|\alpha|=n} X^\alpha T_\alpha \qquad \mod (X^{n+d+1})
\end{aligned}$$

since for $i \geq 2$ we have

$$\deg\left(X^{\sum_{j<i}\alpha_j}\right) = \sum_{j<i} \alpha_j = i \cdot n \geq d+n+1$$

because of $n \geq d+1$.

Note that

$$P'(Q_n) = P'(S) = X^\beta U + V \mod X^{d+1}.$$

Fix $\gamma : \mathbb{N}^I$ with $|\gamma| = n$. Since $(X^{n+d+1}) \subseteq A_{\beta+\gamma}$, we have

$$\begin{aligned}
0 &= P(Q_n) + X^\beta U \sum_{\substack{|\alpha|=n, \\ \alpha \leq \gamma}} X^\alpha T_\alpha + V \sum_{\substack{|\alpha|=n, \\ \alpha < \gamma}} X^\alpha T_\alpha \\
&= P(Q_n) + X^{\beta+\gamma} U T_\gamma + P'(Q_n) \sum_{\substack{|\alpha|=n, \\ \alpha < \gamma}} X^\alpha T_\alpha \mod A_{\beta+\gamma}
\end{aligned}$$

where the comparisons are again with respect to the chosen lexicographical ordering. For a given choice of T_α with $\alpha < \gamma$, there is at most a single choice of T_γ fulfilling the above equation, noting that U is invertible. By induction on the lexicographical ordering on $\gamma : \mathbb{N}^I$ with $|\gamma| = n$, we have thus constrainted the family T to at most a single choice. □

Theorem 2.4. *Given guarded regular functors F and G of arity N, we can decide whether $F = G$ in the set model.*

Proof. A combination of Theorem 2.3 and Lemma 2.22 as detailed above, recalling that $F = G$ is equivalent in the set model to the coefficients of F and G in their respective power series representation $\mathrm{Set}[\![N]\!]$ being isomorphic. □

The Unguarded Part[17]

Since (parametric) initial algebra formation over regular functors is computed by an ω-colimit. The coefficients in the power series representation of a regular functor will always be either

[17] Originally, we had a custom development to deal with the indefinite part, effectively rediscovering Parikh's theorem and parts of the theory of semilinear sets. During the writing-up phase, we discovered that most of this had already been widely known. However, our development may still have some use in internalizing the handling of the indefinite part in the term model. Since we have not yet fully worked out this part in sufficient detail, we decided to omit it. With this, our custom development became redundant and we decided to omit it entirely. This may serve as a lesson in more exhaustively researching background literature before thinking too much about any problem.

finite or countably infinite. Since an indefinite regular functor is invariant under multiplication with $\mathbf{N}$, which equals $\mathbb{N}$ in the set model, all coefficients in the power series representation of an indefinite regular functor will be either zero or isomorphic to $\mathbb{N}$.

In the set model, we may thus identify an indefinite functor F of arity N with the subset $I_F \subseteq \mathbb{N}^N$ of those indices k such that $C_k \neq \emptyset$ in the power series representation $C : \mathrm{Set}\left[\!\left[X^N\right]\!\right]$ of F. Via Lemma 2.18, we can then regard I_F as the language defined by a context-free grammar with commuting terminals. Parikh's theorem [46] then allows us to write I_F as a *semilinear set*, i.e. a finite union of *linear sets* of the form

$$L(c,d) = \left\{ c + \sum_{i<n} k_i d_i \mid k_i \in \mathbb{N} \text{ for } i < n \right\}$$

with $c, d_i : \mathbb{N}$ for $i < n$. By combinatorial topological reasoning [29], one may prove that every semilinear set is in fact the finite union of pairwise disjoint fundamental linear sets. Here, a linear set $L(c,d)$ is called *fundamental* if the vector elements of d are linearly independent over $\mathbb{Q}$. Equality of semilinear sets is already known to be decidable [28] (the exact complexity being Π_2^P in the polynomial hierarchy).

2.4.5 Combining It?

Given a regular functors $F : \mathrm{Set}^N \to \mathrm{Set}$, let us use Theorem 2.1 to decompose $F = U_F + V_F$ with a guarded regular functors $U_F : \mathrm{Set}^N \to \mathrm{Set}$ and an indefinite regular functor $V_F : \mathrm{Set}^N \to \mathrm{Set}$. Switching to the power series representation of regular functors, recall that the coefficients of U_F are all finite, while the coefficients of V_F are respectively either zero or countably infinite. Thus, note that V_F is uniquely determined: its non-zero coefficients occur precisely at the indices where the corresponding coefficients of F are infinite.

We cannot say the same about U_F. Returning to a previous example, note that

$$\begin{aligned}
&\mathrm{List}(X) \times \mathrm{List}(Y) + \mathbf{N} \times \mathrm{List}(X \times Y) \\
&= \mathrm{List}(X \times Y)(\mathrm{List}(X) + \mathrm{List}(Y)) + \mathbf{N} \times \mathrm{List}(X \times Y) \\
&= \mathrm{List}(X \times Y)(X \times \mathrm{List}(X) + Y \times \mathrm{List}(Y)) + \mathbf{N} \times \mathrm{List}(X \times Y)
\end{aligned}$$

as power series over X and Y. Here, all three power series are already presented in guarded-indefinite decomposed form. There does not seem to be a canonical criterion for deciding which is the most fundamental decomposition.

We are thus faced with the problem of deciding equality of algebraic power series *modulo an indefinite part*: given two algebraic power series $A, B : \mathbb{N}\left[\!\left[X^N\right]\!\right]$ and an indefinite power series represented as a semilinear set $C \subseteq \mathbb{N}^N$, do we have $A_k = B_k$ for all $k \in \mathbb{N}^N$ such that $k \notin C$? Noting that the difference $A - B$ will still be algebraic, we may rephrase this as follows:

Problem 2.1. *Given an algebraic power series $A : \mathbb{Z}\left[\!\left[X^N\right]\!\right]$ and an indefinite power series represented as a semilinear set $C \subseteq \mathbb{N}^N$, do we have $A_k = 0$ for all $k \in \mathbb{N}^N$ such that $k \notin C$?*

We thus ask whether it is possible to decide if a given algebraic power series is entirely consumed by a given indefinite power series.

Let us introduce some tools to deal with the periodicity induced by (semi-)linear sets.

Lemma 2.23. *Consider an algebraic power series $q : \mathbb{Q}\left[X^I, U\right]$ with I finite and choose $n \geq 1$. Let $r : \mathbb{Q}\left[X^I, V\right]$ denote the power series comprising the coefficients of q sampled in direction U at periodic intervals of lengths n, i.e. $r_{a,b} = q_{a,nb}$. Then r is algebraic.*

Proof. Let K denote the result of adjoining a primitive n-th root of unity ζ_n to $\mathbb{Q}$ for all $i : I$. Let $Q : \mathbb{Q}(X^I, U)[Z]$ denote the minimal polynomial of q. We have

$$\begin{aligned}
Q[U = \zeta_n^k U](q(U = \zeta_n^k U)) &= Q(q)[U = \zeta_n^k U] \\
&= 0[U = \zeta_n^k U] \\
&= 0
\end{aligned}$$

for $k < n$, making $q(U = \zeta_n^k U)$ algebraic over $\mathbb{Q}(\zeta_n, X^I, U)$. By closure properties of algebraicity, this makes

$$r = \frac{1}{n} \sum_{k<n} q[U = \zeta_n^k U]$$

algebraic over $\mathbb{Q}(\zeta_n, X^I, U)$. However, note that r is invariant under automorphisms of $\mathbb{Q}(\zeta_n)$ over $\mathbb{Q}$ sending ζ_n to ζ_n^k for $k < n$ coprime to n. The same must hold for its minimal polynomial. Thus, we deduce r is in fact algebraic over $\mathbb{Q}(X^I, U)$. □

Corollary 2.6. *Consider an algebraic power series $q : \mathbb{Q}\left[X^I\right]$ with I finite and choose $u, v : \mathbb{N}^I$ with $v_i \geq 1$ for $i : I$. Let $r : \mathbb{Q}\left[Y^I\right]$ denote the power series comprising the coefficients of q sampled at periodic intervals of lengths v_i starting at u_i for $i : I$, i.e. $r_k = q_{(u_i + k_i v_i)_{i:I}}$. Then r is algebraic.*

Proof. A combination of Lemma 2.23 and primitive algebraic operations. □

With Corollary 2.6, it is possible to attack Problem 2.1 in case we can guarantee that the indefinite power series is represented by a union of fundamental linear sets of full dimension where negative integral coefficients are also allowed (subject to the resulting vector having non-negative components), for then the coefficients of the indefinite power series would be periodic in the direction of each formal variable by standard linear algebra. However, we do not currently know how to attack the full problem, which remains open. A previous flawed argument depended on the known falsehood that a diagonal of an algebraic power series is not necessarily algebraic, a fact related to non-closure of algebraic power series under the Hadamard product. For example, the power series in two variables

$$\begin{aligned}
[(1-4X)(1-4Y)]^{-\frac{1}{2}} &= (1-4X)^{-\frac{1}{2}}(1-4Y)^{-\frac{1}{2}} \\
&= \left(\sum_i \binom{2i}{i} X^i\right)\left(\sum_j \binom{2j}{j} Y^j\right) \\
&= \sum_{i,j} \binom{2i}{i}\binom{2j}{j} X^i Y^j
\end{aligned}$$

is clearly algebraic of degree 2, but its diagonal

$$\sum_k \binom{2k}{k}^2 Z^k$$

is not algebraic [57].

2.5 Tools for Working in the Initial Model

2.5.1 Regular Functors are Traversable

Before starting the introduction, let us briefly recall the notions of applicative and traversable functors. Jaskelioff and Rypacek [30] gave rigorous categorical meaning to these notions, being the first to require the correct laws in the definition of traverable functors. A more detailed exposition can be found in the above reference.

For this subsection, fix a cartesian-closed category $\mathcal{C}$, viewed as a monoidal category.

Applicative functors An *applicative functor* on $\mathcal{C}$ is a lax monoidal functor $F : \mathcal{C} \to \mathcal{C}$ that in addition is pointed in a coherent way. Recall that a lax monoidal structure on $F : \mathcal{C} \to \mathcal{C}$ consists of mediating natural maps $\nu^F : 1 \to F1$ and $\mu^F_{X,Y} : FX \times FY \to X \times Y$ fulfilling the usual unitarity and associativity axioms. The functor F being pointed requires a natural transformation $\eta^F : 1 \to F$, and coherence with the monoidal structure means $\nu^F = \eta^F_1$ and $\mu^F_{X,Y} \circ (\eta^F_X \times \eta^F_Y) = \eta^F_{X \times Y}$.

An *applicative morphism* between applicative functors $F, G : \mathcal{C} \to \mathcal{C}$ is a monoidal transformation $u : F \to G$, i.e. a natural transformation respecting the lax monoidal structure on F and G, that in addition respects the pointedness of F and G.

Applicative functors and morphisms form a strict monoidal category $\mathscr{A}$ where the monoidal structure is given by functor composition.

Traversable functors A functor $T : \mathcal{C} \to \mathcal{C}$ is called *traversable* if it is endowed with a family of *traversals* $\delta^T_F : TF \to FT$ natural in the applicative functor argument $F : \mathscr{A}$. This family is required to respect the monoidal structure of $\mathscr{A}$, meaning it has to fulfill the *unitarity law*

$$\begin{array}{ccc} T \circ \mathrm{Id} & \xrightarrow{\mathrm{id}_T} & \mathrm{Id} \circ T \\ & \searrow^{\delta^T_{\mathrm{Id}}} \nearrow & \end{array}$$

and the *linearity law*

$$\begin{array}{ccccc} T \circ G \circ F & & \xrightarrow{\quad\delta^T_{GF}\quad} & & (G \circ F) \circ T \\ & \searrow^{\delta^T_G \circ RF} & & \nearrow_{G \circ \delta^T_F} & \\ & & G \circ T \circ RF & & \end{array}$$

for applicative functors F and G.

For a slick 2-categorical presentation, also found in [30], recall that a monoidal category $\mathscr{C}$ can be viewed as a 2-category $\underline{\mathscr{C}}$ with a single object. Objects and morphisms of the original monoidal category correspond to 1-cells and 2-cells of the 2-category, with the monoidal structure in particular corresponding to identity and composition of 1-cells. Under this equivalence, strict monoidal categories correspond to strict 2-categories with a single object.

Given any category $\mathcal{C}$, the functor category $\mathcal{C} \to \mathcal{C}$ of endofunctors on $\mathcal{C}$ has a strict monoidal structure given by functor composition. There is a canonical inclusion 2-functor from $\underline{\mathcal{C} \to \mathcal{C}}$ to Cat mapping the unique 0-cell to $\mathcal{C}$ and leaving everything else untouched. Noting that $\mathscr{A}$ is a monoidal sub-category of $\mathcal{C} \to \mathcal{C}$, let $I : \underline{\mathscr{A}} \to \mathrm{Cat}$ denote this inclusion 2-functor. Traversable functors are then just oplax natural transformations $I \to I$.

Introduction The development in [30] is done for the special case where $\mathcal{C}$ is the category of sets Set. As shown above, the basic definitions easily transfer to the general case, and so does their proof that traversable functors are closed under products. Unfortunately, the rest of the

development is specific to Set: they go on to show that every finitary container is traversable. In Set, containers are closed under parametric formation of initial algebras, so this implies traversability of all regular functors over Set. This approach does not carry over to our setting, where $\mathcal{C}$ might for example be a term model. However, preservation of traversability under colimits and parametric formation of initial algebras still holds, and this insight will enable us to conclude that regular functors over any bicartesian-closed category are traversable in any of their parameters, and in fact with respect to simultaneous traversal of any subset of parameters, as will be elaborated upon later.

Let us give some intuition on why previous attempts in the literature, notably [20], fail to properly establish closure of traversabilty under formation of initial algebras. Suppose we are given a functor $S : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ separately traversable in each argument. Assume further that the parametric initial algebra of S in its second argument exists, and denote $T(X) = \mu Y.S(X, Y)$. Let us naively try to construct a family of traversals $\delta^T_F : TF \to FT$ for F applicative. Fixing $X : \mathcal{C}$, the typing

$$\delta^T_{F,X} : \mu Y.S(F(X), Y) \to F(\mu Y.S(X, Y))$$

suggests defining $\delta^T_{F,X}$ as a fold. We are left with defining an algebra

$$S(F(X), F(\mu(Y).S(X, Y))) \to F(\mu Y.S(X, Y)).$$

Expanding the μ-expression on the right in preparation of traversing on F, our goal is

$$S(F(X), F(\mu(Y).S(X, Y))) \to F(S(X, \mu Y.S(X, Y))).$$

This typing does indeed look awkwardly appropriate for a traversal, but in each variable at once! We can try to traverse separately on each variable, e.g. giving us

$$S(F(X), F(\mu(Y).S(X, Y))) \longrightarrow FS(X, F(\mu(Y).S(X, Y))) \longrightarrow F^2S(X, \mu(Y).S(X, Y)),$$

but since F is not a monad, there seems to be no way to get rid of the duplication of F. In fact, this is precisely the reason why traversals (under the name of distributivity and monadic maps) in [20] are constructed only for the case of monads F. However, non-composability of monads and the requirement for the structure of the monad F to align with its structure as a lax monoidal functor [20, Paragraph 5.1] make this treatment not only less general, but unnecessarily convoluted, mixing together two orthogonal concerns.

Let us remark on an aspect that for some reason does not appear mentioned in the literature. Of all the below constructions involved in showing that regular functors are traversable, only closure under finite products makes use of the specific structure of the domain 2-category $\underline{\mathscr{A}}$ and the functor I. The remaining constructions all take place in a vastly more general setting of arbitrary 2-categories and 2-functors. Furthermore, even the proof of closure under finite products only ever uses the lax monoidal structure of applicative functors, not the pointedness. Thus, Theorem 2.5 can immediately be made more abstract by defining traversability with respect to any lax monoidal functor, not just applicative ones. Of course, this makes no difference for a cartesian-closed category $\mathcal{C}$ such that every lax monoidal endofunctor has a strength, most notably Set, for in that case, pointedness can be recovered from the unit of the lax monoidal structure.

R-traversable functors The solution to the above dilemma is, of course, to generalize the notion of traversability to functors of arbitrary arity. Given a cartesian-closed category $\mathcal{D}$ and a cartesian functor $R : \mathcal{C} \to \mathcal{D}$, a functor $T_0 : \mathcal{D} \to \mathcal{C}$ is called *R-traversable* if it is the 0-component of an oplax monoidal transformation $T : R \circ I \to I$. In the standard case where $\mathcal{D} = \mathcal{C}^n$ and R is the diagonal functor $\Delta_n : \mathcal{C} \to \mathcal{C}^n$, we will simply say that T_0 is *$n-$traversable*. Explicitly, this amounts to a family of *R-traversals* $\delta^{T_0}_F : T_0 \circ RF \to F \circ T_0$ natural in the

applicative functor argument $F : \mathscr{A}$, subject to the laws of *unitarity*

$$T_0 \circ R\,\mathrm{Id} \xrightarrow{\mathrm{id}_{T_0}} \mathrm{Id} \circ T_0 \qquad \delta^{T_0}_{\mathrm{Id}}$$

and *linearity*

$$T_0 \circ R(G \circ F) \xrightarrow{\delta^{T_0}_{GF}} (G \circ F) \circ T_0$$
$$\delta^{T_0}_G \circ RF \qquad G \circ \delta^{T_0}_F$$
$$G \circ T_0 \circ RF$$

for applicative functors F and G.

Lemma 2.24 (Closure under Composition)**.** *Fix a cartesian-closed category $\mathcal{D}$ and a natural number n. For any $i < n$, the variable selection functor $\pi^n_i : \mathcal{C}^n \to \mathcal{C}$ is n-traversable. For any functor $T : \mathcal{C}^n \to \mathcal{C}$ and any sequence of argument functors $S_1, \ldots, S_n : \mathcal{D} \to \mathcal{C}$, if T is n-traversable and if $S_1, \ldots, S_n$ are R-traversable with respect to a cartesian functor $R : \mathcal{C} \to \mathcal{D}$, then $T \circ (S_1 \times \ldots \times S_n)$ is R-traversable.*

Proof. This is simply due to the cartesian-closed structure of the category $[I, \mathrm{Cat}]_{l^{\mathrm{op}}}$ with oplax natural transformations as morphisms. □

A side effect of our generalization is that closure of traversability under finite products actually becomes more direct to prove.

Lemma 2.25 (Cartesian Closure)**.** *The terminal object functor $1 : 1 \to \mathcal{C}$ is 0-traversable. The product functor $\cdot \times \cdot : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ is 2-traversable.*

Proof. This is an unfolding of $\mathscr{A}$ as a monoidal category of lax monoidal functors, making I into a 2-monoid in the cartesian-closed 2-category $[\mathscr{A}, \mathrm{Cat}]_{l^{\mathrm{op}}}$ that has oplax natural transformations for morphisms. Nevertheless, let us check the details:

For the terminal object, the corresponding oplax natural transformation $1 \to I$ is given as follows:

- The 0-component is the terminal object functor 1.
- Given an applicative functor F, the 1-component corresponding to F is given by the unit mediating map of the lax monoidal endofunctor F.
- The 2-cells for identity and composition hold by definition of identity and composition of lax monoidal functors.

For binary products, the corresponding oplax natural transformation $I \times I \to I$ is given as follows:

- The 0-component is the product functor $\cdot \times \cdot$.
- Given an applicative functor F, the 1-component corresponding to F is given by the product mediating map of the lax monoidal endofunctor F.
- The 2-cells for identity and composition hold by definition of identity and composition of lax monoidal functors.

□

For our primary goal of endowing all regular functors with traversals, we are missing two more lemmata: closure of traversability under finite coproducts and parametric formation of initial algebras. It turns out that both assertions hold in a much more abstract context.

The below result may be summarized as stating that under suitable conditions, limits in categories of lax natural transformations are computed pointwise. We suspect this result to be known in the enriched category theory community and proved elegantly as well as shortly using standard methods, but our knowledge of the literature is too slim to say for sure. We apologize to the reader in advance.

Lemma 2.26. *Fix a 2-category $\mathcal{B}$ and 2-functors $F, G : \mathcal{B} \to \mathrm{Cat}$. Fix a shape category $\mathcal{I}$. If G is valued in categories closed under limits of shape $\mathcal{I}$, then $\mathrm{Lax}(F, G)$ is closed under limits of shape $\mathcal{I}$ as well. Furthermore, the object component on $X : \mathcal{B}$ of the limiting lax natural transformation of a diagram $D : \mathcal{I} \to \mathrm{Lax}(F, G)$ is given by the limit of the objectwise diagram $D^X : \mathcal{I} \to [FX, GX]$.*

Proof. Fix a diagram $D : \mathcal{I} \to \mathrm{Lax}(F, G)$. Let us try to find an easier description of the category of cones over D:

$$\begin{aligned}
\mathrm{Cone}(D) &\simeq \int^{t:\mathrm{Lax}(F,G)} [\mathcal{I}, \mathrm{Lax}(F, G)](\Delta t, D) \\
&\simeq \int^{t:\mathrm{Lax}(F,G)} \int_{A:\mathcal{I}} \mathrm{Lax}(F, G)(t, DA) \\
&\simeq \int^{t:\mathrm{Lax}(F,G)} \int_{A:\mathcal{I}} \int^{l}_{X:\mathcal{B}} [FX, GX](t_X, D(A)_X) \\
&\simeq \int^{t:\mathrm{Lax}(F,G)} \int^{l}_{X:\mathcal{B}} \int_{A:\mathcal{I}} [FX, GX](t_X, D^X(A)) \\
&\simeq \int^{l}_{X:\mathcal{B}} \int^{t:[GX,FX]} \int_{A:\mathcal{I}} [FX, GX](t, D^X(A)) \\
&\simeq \int^{l}_{X:\mathcal{B}} \int^{t:[FX,GX]} [\mathcal{I}, [FX, GX]](\Delta t, D^X) \\
&\simeq \int^{l}_{X:\mathcal{B}} \mathrm{Cone}(D^X)
\end{aligned}$$

Here, the end annotated with a superscript l denotes a lax weighted limit replacing the weighted limit in the definition of enriched ends, and ante-penultimate step uses a form of the axiom of choice. Thus, a cone over D is a coherent family of cones over D^X with $X : \mathcal{B}$.

A candidate cone Let us try construct the limiting cone as an element of the last category in the above chain of equivalences. For each $X : \mathcal{B}$, we have to choose a cone over D^X, and we simply choose the locally limiting cone, call it (L_X, ω_X) where $L_X : [FX, GX]$ and $\omega_X : [\mathcal{I}, [FX, GX]](\Delta L_X, D^X)$, which exists by assumption. If we succeed in completing the definition of this cone and showing that it is indeed globally limiting, then re-traversing the above chain of equivalences verifies that the object components of our global limit are indeed given by the local limits, i.e. that the limit is computed pointwise.

For coherence, for each $f : X \to Y$ in $\mathcal{B}$, we also have to specify a morphism $\beta_f : Gf \circ L_X \to L_Y \circ Ff$ in $[FX, GY]$ such that the following diagram over $[\mathcal{I}, [FX, GX]]$ commutes:

$$\begin{array}{ccc}
(Gf \circ \cdot) \circ \Delta L_X & \xrightarrow{\Delta \beta_f} & \Delta L_Y \circ (\cdot \circ Ff) \\
\downarrow{\scriptstyle (Gf \circ \cdot) \circ \omega_X} & & \downarrow{\scriptstyle \omega_Y \circ (\cdot \circ Ff)} \\
(Gf \circ \cdot) \circ D^X & \xrightarrow{D^f} & D^Y \circ (\cdot \circ Ff)
\end{array} \tag{2.4}$$

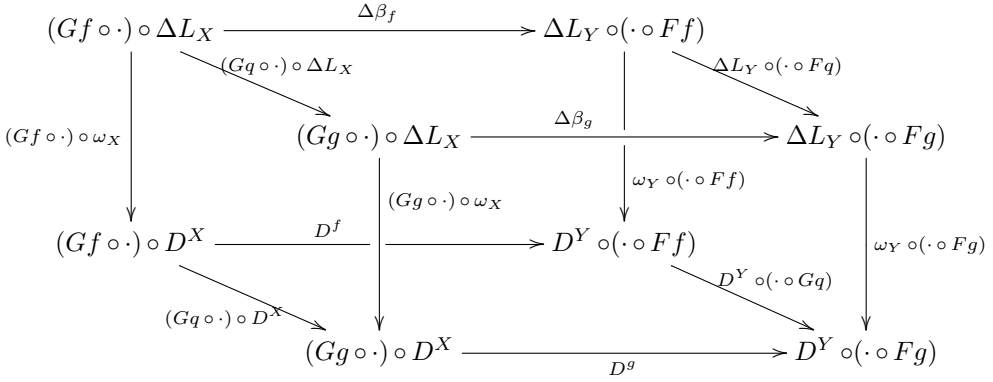
We warn that the composition operator is overloaded here. Recall that $(\cdot \circ Ff) : [FY, GY] \to [FX, GY]$ and $(Gf \circ \cdot) : [FX, GX] \to [FX, GY]$ denote the functorial pre- and postcomposition operations. The morphism D^f is defined pointwise for $A : \mathcal{B}$ as the 1-component of the lax natural transformation $D(A)$ on f. It is natural in A since D is a natural family of lax natural transformations.

Note that the composite data consisting of the map β_f and the above commuting square can also be read as a morphism

$$\overline{\beta}_f : (Gf \circ L_X, D^f \circ((Gf \circ \cdot) \circ \omega_X)) \to (L_Y \circ Ff, \omega_Y \circ (\cdot \circ Ff))$$

in the category $\mathrm{Cone}(D^Y \circ (\cdot \circ Ff))$. The key insight now is that the limit of D^Y being computed pointwise means it is preserved under functor precomposition. In particular, this means $(L_Y \circ Ff, \omega_Y \circ (\cdot \circ Ff))$ will still be the limiting cone over $D^Y \circ Ff$, i.e. the terminal object in $\mathrm{Cone}(D^Y \circ (\cdot \circ Ff))$. This uniquely determines the coherence data, completing the specification of our candidate for the limiting cone of D.

Given $X, Y : \mathcal{B}$, we also have to check that the assignment of $f : X \to Y$ to β_f is natural in f with respect to the domain category $\mathcal{B}(X, Y)$. For this, let a 2-cell $q : f \Rightarrow g$ with $f, g : \mathcal{B}(X, Y)$ be given. Consider the following diagram:



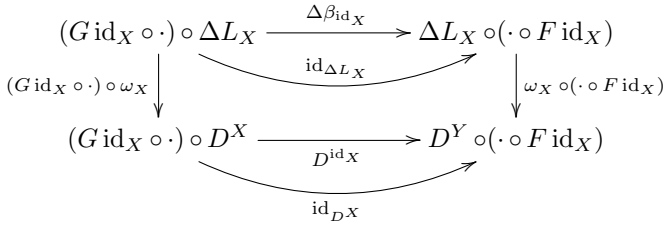
The back and front squares commute by construction of the maps β_f and β_g. The left and right squares commute by 2-functoriality of G and F, respectively, and compositionality. Commutativity of the bottom square is just the 2-component of the family of lax natural transformations D. Altogether, this makes $\beta_g \circ (Gq \circ L_X)$ and $(L_Y \circ Fq) \circ \beta_f$ into morphisms in $\mathrm{Cone}(D^Y \circ (\cdot \circ Fg))$ of type

$$(Gf \circ L_X, (D^Y \circ (\cdot \circ Gq)) \circ D^f \circ ((Gf \circ \cdot) \circ \omega_X)) \to (\Delta L_Y \circ Fg, \omega_Y \circ (\cdot \circ Fg)).$$

Once again, the latter cone is limiting, implying $\beta_g \circ (Gq \circ L_X) = (L_Y \circ Fq) \circ \beta_f$, thus verifying the above naturality.

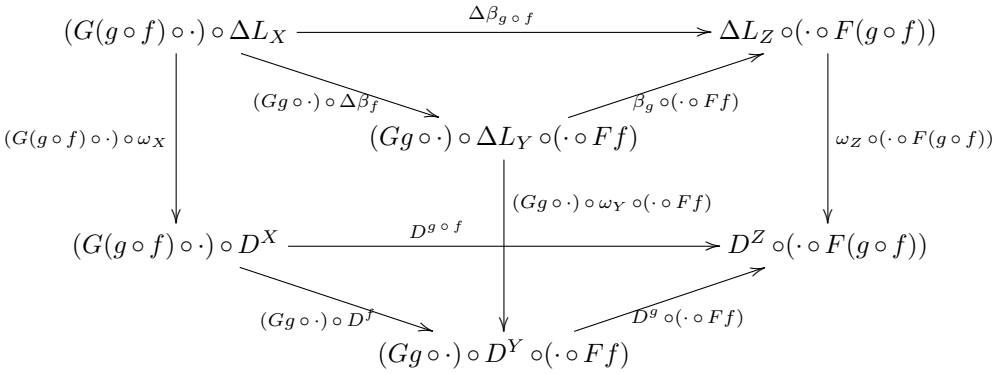
On another level of coherence, we have to check closure properties of β under identity and composition. All together, this will make (L, β) into a lax monoidal transformation $F \to G$. In fact, under the above chain of equivalences, this lax monoidal transformation corresponds to the first component of the global cone we are constructing.

For the identity on $X : \mathcal{B}$, note that in the diagram



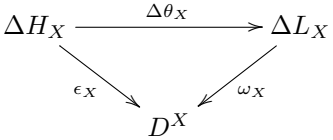
both square faces commute. Commutativity of the lower two-sided face thus makes id_{L_X} into a candidate map making the original square commute. Uniqueness of maps β_{id_X} with this property implies $\beta_{\mathrm{id}_X} = \mathrm{id}_{L_X}$.

For the composition of $f : X \to Y$ and $g : Y \to Z$ in $\mathcal{B}$, note that in the diagram



all three squares commute. Pasting together the two small squares and using commutativity of the lower triangle makes $(\beta_g \circ (\cdot \circ Ff)) \circ ((Gg \circ \cdot) \circ \beta_f)$ into a candidate map making the original square commute. Uniqueness of maps $\beta_{g \circ f}$ with this property implies $\beta_{g \circ f} = (\beta_g \circ (\cdot \circ Ff)) \circ ((Gg \circ \cdot) \circ \beta_f)$.

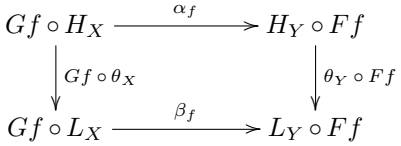
Showing the candidate cone limiting Now fix any other cone over D, given as a family of local cones $(H_X, \epsilon_X) : \mathrm{Cone}(D^X)$ coherent via a coherent family of maps $\alpha_f : Gf \circ H_X \to H_Y \circ Ff$ for $f : X \to Y$ in $\mathcal{B}$ as seen above. A morphism from $((H_X, \epsilon_X)_{X:\mathcal{B}}, \alpha)$ to $((L_X, \omega_X)_{X:\mathcal{B}}, \beta)$ consists first of a family of cone morphisms over D^X from (H_X, ϵ_X) to (L_X, ϵ_X) given by a map $\theta_X : H_X \to L_X$ with the commuting condition



(2.5)

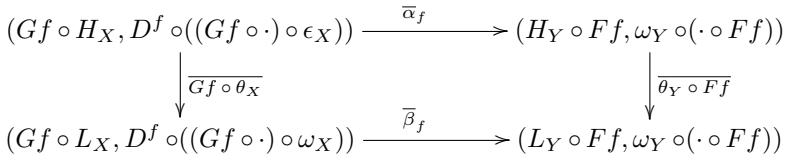
where $X : \mathcal{B}$. Since (L_X, ϵ_X) is locally a limiting cone, this data is fully constrained and given by the unique morphism to the terminal object in $\mathrm{Cone}(D^X)$.

Second, this family is subject to coherence: for $f : X \to Y$ in $\mathcal{B}$, we need to have commutativity of the following diagram over $[FX, GY]$:



(2.6)

This diagram is part of a larger picture of morphisms in $\mathrm{Cone}(D^Y \circ (\cdot \circ Ff))$:



(2.7)

Here, coherence (2.4) of α_f and β_f makes the horizontal lines into cone morphisms $\overline{\alpha}_f$ and $\overline{\beta}_f$ as explained above. For the left morphism, the cone morphism $\overline{Gf \circ \theta_X}$ is given by postfixing

the commuting triangle (2.5) for θ_X by $(Gf \circ \cdot)$ and then postcomposing with D_f:

$$\begin{array}{ccc} (Gf \circ \cdot) \circ \Delta H_X & \xrightarrow{(Gf \circ \cdot) \circ \Delta \theta_X} & (Gf \circ \cdot) \circ \Delta L_X \\ & {\scriptstyle (Gf \circ \cdot) \circ \epsilon_X} \searrow \quad \swarrow {\scriptstyle (Gf \circ \cdot) \circ \omega_X} & \\ & (Gf \circ \cdot) \circ D^X & \\ & \downarrow {\scriptstyle D^f} & \\ & D^Y \circ (\cdot \circ Ff) & \end{array}$$

For the right morphism, the cone morphism $\overline{\theta_Y \circ Ff}$ is simply given by prefixing the commuting triangle (2.5) for θ_X by $(\cdot \circ Gf)$:

$$\begin{array}{ccc} \Delta H_X \circ (\cdot \circ Ff) & \xrightarrow{\Delta \theta_X \circ (\cdot \circ Ff)} & \Delta L_X \circ (\cdot \circ Ff) \\ & {\scriptstyle \epsilon_X \circ (\cdot \circ Ff)} \searrow \quad \swarrow {\scriptstyle \omega_X \circ (\cdot \circ Ff)} & \\ & D^X \circ (\cdot \circ Ff) & \end{array}$$

Now, note that the lower right object of the enlarged diagram (2.7) is the terminal object in the category of cones over $D^Y \circ (\cdot \circ Ff)$ as already shown via functor precomposition limit preservation of pointwise limits. This makes the diagram (2.7), and by extension the diagram (2.6) commute automatically. We have thus verified that the global cone over D represented by $((L_X, \omega_X)_{X:\mathcal{B}}, \beta)$ is indeed limiting. □

Corollary 2.7 (Dual of Lemma 2.26)**.** *If G is valued in categories closed under colimits of shape $\mathcal{I}$, then* $\mathrm{OpLax}(F, G)$ *is closed under colimits of shape $\mathcal{I}$ as well. Furthermore, the object component on $X : \mathcal{B}$ of the colimiting oplax natural transformation of a diagram $D : \mathcal{I} \to \mathrm{Lax}(F, G)$ is given by the colimit of the objectwise diagram $D_X : \mathcal{I} \to [FX, GX]$.*

Returning to our original context, we derive:

Corollary 2.8. *Fix a cartesian-closed category $\mathcal{D}$ and a cartesian functor $R : \mathcal{C} \to \mathcal{D}$. The initial object functor $0 : 1 \to \mathcal{C}$ is 0-traversable. The coproduct functor $\cdot + \cdot : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ is 2-traversable.*

Proof. For the initial object functor, the statement is given by Corollary 2.7 with $\mathcal{B} = \underline{\mathscr{A}}$, $F = 1$, $G = I$, and applied to the initial object diagram, i.e. the diagram of shape the empty category.

Write $\cdot + \cdot$ as the coproduct of π_1 and π_2 in the functor category $[\mathcal{C} \times \mathcal{C}, \mathcal{C}]$. Since π_1 and π_2 are 2-traversable by Lemma 2.24, it remains to verify that traversability is closed under coproducts in $[\mathcal{C} \times \mathcal{C}, \mathcal{C}]$. But this is the statement of Corollary 2.7 with $\mathcal{B} = \underline{\mathscr{A}}$, $F = I \times I$, $G = I$, and applied to coproduct diagrams, i.e. diagrams of shape the discrete category on 2 objects. □

The below result may be summarized as stating that under suitable conditions, terminal coalgebras of endofunctors over categories of lax natural transformations are computed pointwise. The structure of the proof we give here is quite similar to the one of the previous lemma, and in fact this redundancy gives credit to the idea that they are both instances of a more general theorem establishing pointwise computation of a wider range of universal constructions characterized by terminality. Again, we suspect this result to be known in the enriched category theory community and proved elegantly as well as shortly using standard methods. We apologize to the reader in advance, for we have not yet been able to express the following proof in greater abstraction.

Lemma 2.27. *Let a 2-category $\mathcal{B}$ and 2-functors $F, G : \mathcal{B} \to \mathrm{Cat}$ be given. Fix a lax natural transformation $u : F \times G \to G$ and introduce the functor*

$$\begin{aligned} \overline{u} &: \mathrm{Lax}(F,G) \to \mathrm{Lax}(F,G), \\ \overline{u}(t) &= u \circ (F \times t) \circ \Delta_F. \end{aligned}$$

On a given object $X : \mathcal{B}$, note that u_X analogously induces a functor

$$\begin{aligned} \overline{u_X} &: FX \to GX, \\ \overline{u_X}(t) &= u_X \circ (FX \times t) \circ \Delta_{FX}. \end{aligned}$$

Recall that for any $X : \mathcal{B}$, if $u_X(C,\cdot) : GX \to GX$ has a terminal coalgebra for any $C : FX$, then so does $\overline{u_X}$ and on an object $C : FX$ it is given by the terminal coalgebra of $u_X(C,\cdot)$. If this assumption is fulfilled for all $X : \mathcal{B}$, then $\overline{u}$ has a terminal coalgebra as well, having the terminal coalgebra of $\overline{u_X}$ as 0-component for any object $X : \mathcal{C}$.

Proof. For a morphism $f : X \to Y$ in $\mathcal{B}$, define an endofunctor

$$\begin{aligned} h_f &: [FX, GY] \to [FX, GY], \\ h_f(v) &= u_Y \circ \langle Ff, v \rangle \\ &= u_Y \circ (Ff \times v) \circ \Delta_{FX}. \end{aligned}$$

This is a generalization of the endofunctor $\overline{u_X}$ on $[FX, GX]$ since $\overline{u_X} = h_{\mathrm{id}_X}$ for $X : \mathcal{B}$.

Note that h behaves stably under precomposition: we have

$$h_g(v) \circ Ff = h_{g \circ f}(v \circ Ff)$$

for $v : [FY, GZ]$ with morphisms $f : X \to Y$ and $g : Y \to Z$ in $\mathcal{B}$. In particular, given $f : X \to Y$ in $\mathcal{B}$, we have $h_{\mathrm{id}_Y}(v) \circ Ff = h_f(v \circ Ff)$, i.e. $h_f(v \circ Ff) = \overline{u_X}(v) \circ Ff$. Since the terminal coalgebra over $\overline{u_X}$ as a functor is calculated pointwise, it is stable under context change, i.e. precomposition with Ff. That is, if v is (the object part of) the terminal coalgebra over $\overline{u_X}$, then $v \circ Ff$ will be (the object part of) the terminal coalgebra over h_f.

For postcomposition, we only have a lax version of the above identity. For any $v : [FX, GY]$, we have

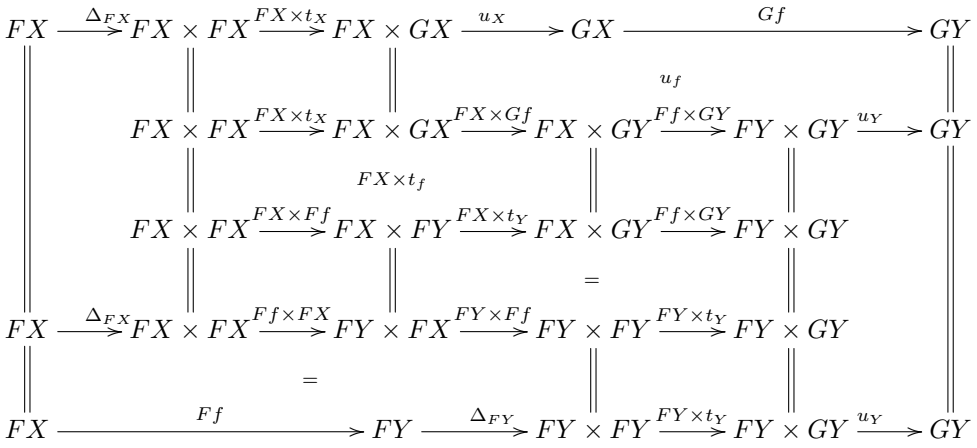
$$u_g \circ \langle Ff, g \rangle : Gg \circ h_f(v) \to h_{g \circ f}(Gg \circ v).$$

with morphisms $f : X \to Y$ and $g : Y \to Z$ in $\mathcal{B}$.

A candidate coalgebra Returning to the original scenario, a coalgebra over $\overline{u}$ consists of a lax natural transformation $t : F \to G$ together with a modification $n : t \to \overline{u}(t)$. On an object $X : \mathcal{B}$, this amounts to a functor $t_X : FX \to GX$ and a map $n_X : t_X \to \overline{u}(t)_X$, i.e. a $\overline{u_X}$-coalgebra as $\overline{u}(t)_X = \overline{u_X}(t_X)$. For a morphism $f : X \to Y$ in $\mathcal{B}$, we have to supply a 1-component $t_f : Gf \circ t_X \to t_Y \circ Ff$ and a proof that n respects the corresponding 1-components t_f and $\overline{u}(t)_f$, i.e. that the following diagram commutes:

$$\begin{array}{ccc} Gf \circ t_X & \xrightarrow{Gf \circ n_X} & Gf \circ \overline{u_X}(t_X) \\ {\scriptstyle t_f}\downarrow & & \downarrow{\scriptstyle \overline{u}(t)_f} \\ t_Y \circ Ff & \xrightarrow{n_Y \circ Ff} & \overline{u_Y}(t_Y) \circ Ff \end{array} \tag{2.8}$$

Here, $\overline{u}(t)_f$ is the top-down composition illustrated in the below drawing:



Using our identities for h, we may thus recast the diagram (2.8) as follows:

$$\begin{array}{ccc} Gf \circ t_X & \xrightarrow{t_f} & t_Y \circ Ff \\ {\scriptstyle Gf \circ n_X}\downarrow & & \downarrow{\scriptstyle n_Y \circ Ff} \\ Gf \circ h_{\mathrm{id}_X}(t_X) & & h_{\mathrm{id}_Y}(t_Y) \circ Ff \\ {\scriptstyle u_f \circ \langle \mathrm{id}_{FX}, t_X \rangle}\downarrow & & \| \\ h_f(Gf \circ t_X) & \xrightarrow{h_f(t_f)} & h_f(t_Y \circ Ff) \end{array} \tag{2.9}$$

This transforms the choice of t_f together with the constraint (2.8) from the coalgebra

$$(Gf \circ t_X, (u_f \circ \langle \mathrm{id}_{FX}, t_X \rangle) \circ (Gf \circ n_X))$$

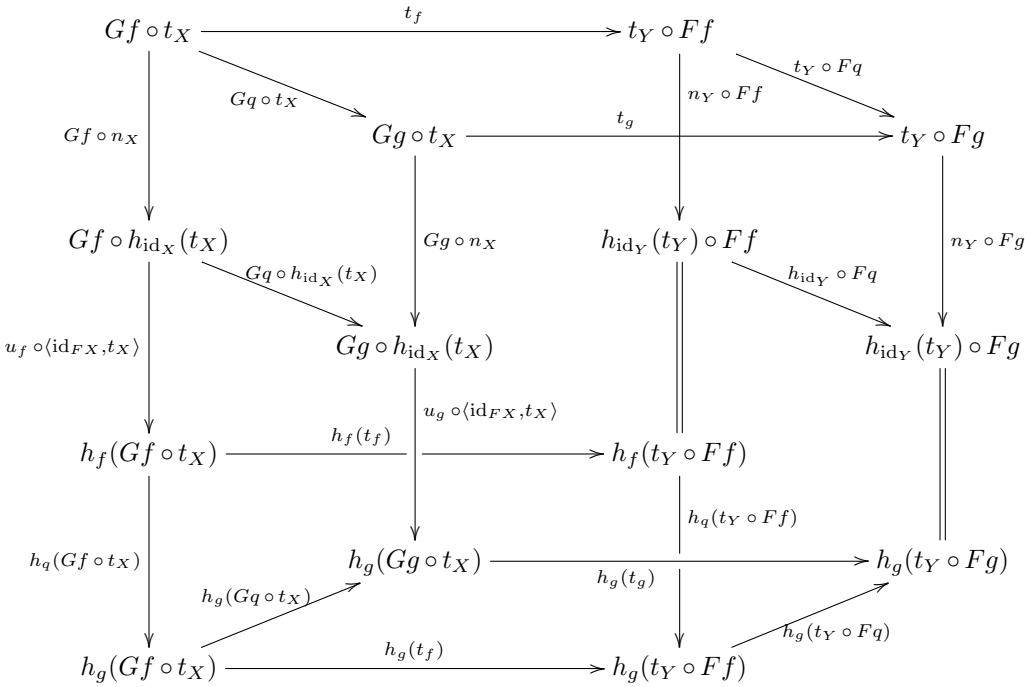
into the coalgebra

$$(t_Y \circ Ff, n_Y \circ Ff)$$

over the functor h_f. Since $(t_Y \circ Ff, n_Y \circ Ff)$ is the terminal coalgebra over h_f as explained after the construction of h, this makes this choice unique. Note that the derivation of the constraint (2.8) was by equivalence transformations: any coalgebra morphism over $\overline{u}$ will satisfy a corresponding constraint. This fact will become important when verifying global terminality.

Given $X, Y : \mathcal{B}$, we also have to check that the assignment of $f : X \to Y$ to t_f is natural in f with respect to the domain category $\mathcal{B}(X, Y)$. For this, let a 2-cell $q : f \Rightarrow g$ with $f, g : \mathcal{B}(X, Y)$

be given. Consider the following diagram:



The back top face and and the front face commute by construction of t_f and t_g as coalgebra morphisms (2.9). The left and right top faces commute by interchange. The right bottom and left bottom pentagons commute by 2-functoriality of F and G, respectively.

Pasting together the back faces, note that t_f forms a morphism between coalgebras over h_g. Pasting together the left and right faces, note that $Gq \circ t_X$ and $t_Y \circ Fq$, respectively, form morphisms between coalgebras over h_g. Composing the pasting of the back faces with the pasting of the right faces as well as the pasting of the left faces with the the front face, we get coalgebra morphisms $(t_Y \circ Fq) \circ t_f$ and $t_g \circ (Gq \circ t_X)$ from the coalgebra

$$(Gf \circ t_X, h_q(Gf \circ t_X) \circ (u_f \circ \langle \mathrm{id}_{FX}, t_X \rangle) \circ (Gf \circ n_X))$$

to the coalgebra

$$(t_Y \circ Fg, n_Y \circ Fg)$$

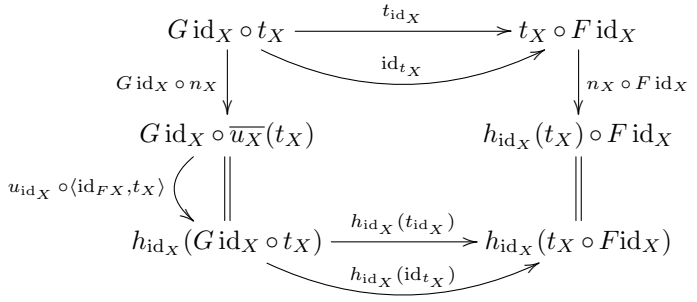
over the functor h_g. Recalling that the latter coalgebra is terminal, it follows that

$$(t_Y \circ Fq) \circ t_f = t_g \circ (Gq \circ t_X),$$

thus verifying the 2-naturality of t.

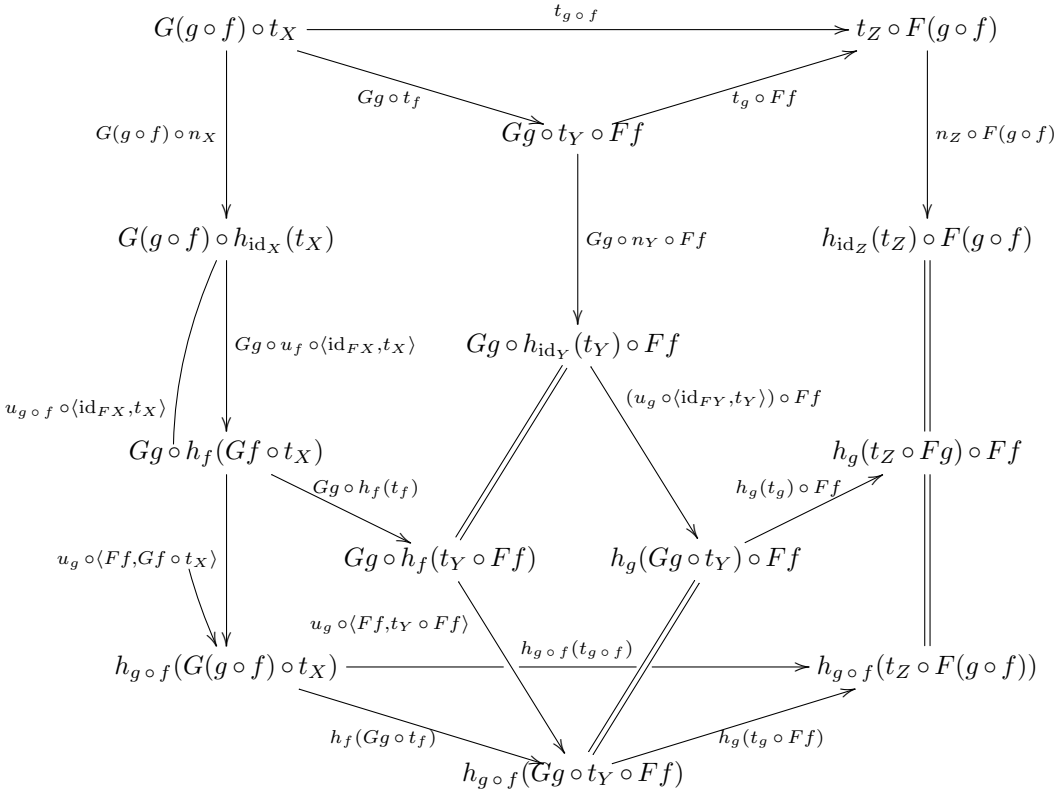
On another level of coherence, we have to check closure properties of t under identity and composition. All together, this will make t into a lax monoidal transformation $F \to G$.

For the identity on $X : \mathcal{B}$, consider the following diagram:



The two-sided face with an equality assertion on the right side commutes by unitarity of u. The almost-straight pentagon containing of the curved arrows only the one on the left commutes by construction of t_X (2.9). The curved square (identifying the equal objects) commutes as it comprises only identities. Both t_{id_X} and id_{t_X} thus form parallel coalgebra morphisms into the terminal coalgebra over h_{id_X}. We must hence have $t_{\mathrm{id}_X} = \mathrm{id}_{t_X}$.

For the composition of $f : X \to Y$ and $f : Y \to Z$ in $\mathcal{B}$, gaze at the diagram



The back, front left top, and front right top hexagons commute by construction of $t_{g \circ f}$, t_f, and t_g, respectively (2.9). The front diamond as well as the front right bottom square are simple identities. The front left bottom square commutes by interchange. Most importantly, but somewhat obscured, the outer left bottom triangle commutes by linearity of u. Pasting the front left faces makes $Gg \circ t_f$ into a coalgebra morphism over $h_{g \circ f}$. Pasting the front right faces makes $t_g \circ Ff$ into a coalgebra morphism over $h_{g \circ f}$. Finally, using the front diamond and the outer left bottom triangle to compose these to coalgebra morphisms makes $(t_g \circ Ff) \circ (Gg \circ t_f)$ into a coalgebra morphism from the coalgebra

$$(G(g \circ f) \circ t_X, (u_{g \circ f} \circ \langle \mathrm{id}_{FX}, t_X \rangle) \circ (G(g \circ f) \circ n_X))$$

to the coalgebra

$$(t_Z \circ F(g \circ f), n_Z \circ F(g \circ f))$$

over the functor $h_{g \circ f}$. Both $t_{g \circ f}$ and $(t_g \circ Ff) \circ (Gg \circ t_f)$ thus form parallel coalgebra morphisms into the terminal coalgebra over h_{id_X}. We must hence have $t_{g \circ f} = (t_g \circ Ff) \circ (Gg \circ t_f)$.

Showing the candidate coalgebra terminal Now fix any other coalgebra over $\overline{u}$, comprising a lax natural transformation $s : F \to G$ and a modification $m : s \to \overline{u}(s)$. A morphism from (s, m) to (t, n) in the category of coalgebras over $\overline{u}$ consists of a modification $\theta : s \to t$ such that $\overline{u}(\theta) \circ m = n \circ \theta$. This amounts to a coherent family of maps $\theta_X : s_X \to t_X$ for $X : \mathcal{B}$

such the following diagram over $[FX, GX]$ commutes:

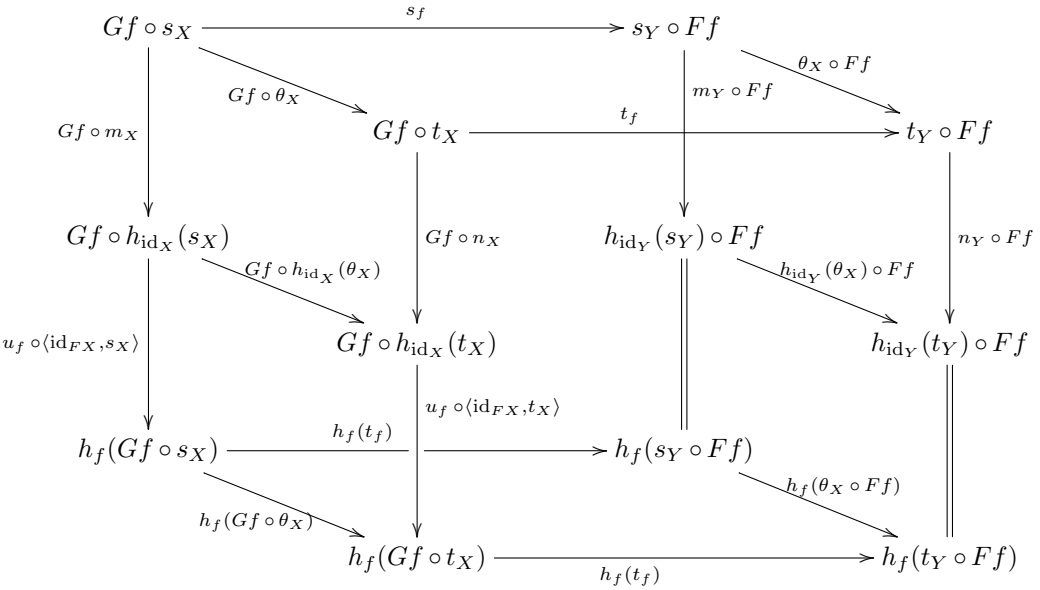
$$\begin{array}{ccc} s_X & \xrightarrow{\theta_X} & t_X \\ \downarrow{\scriptstyle m_X} & & \downarrow{\scriptstyle n_X} \\ \overline{u_X}(s_X) & \xrightarrow{\overline{u_X}(\theta_X)} & \overline{u_X}(t_X) \end{array} \tag{2.10}$$

Observing θ_X thus constitutes a coalgebra morphism from the coalgebra (s_X, m_X) to the coalgebra (t_X, n_X) over $\overline{u_X}$. Since the latter coalgebra was chosen terminal, this makes the choice of the map θ_X under the above constraint unique.

However, we also have to verify coherence of the family θ in order for it to constitute a modification. Given $f : X \to Y$ in $\mathcal{C}$, we need to verify commutativity of the following diagram over $[FX, GY]$:

$$\begin{array}{ccc} Gf \circ s_X & \xrightarrow{s_f} & s_Y \circ Ff \\ \downarrow{\scriptstyle Gf \circ \theta_X} & & \downarrow{\scriptstyle \theta_X \circ Ff} \\ Gf \circ t_X & \xrightarrow{t_f} & t_Y \circ Ff \end{array}$$

This diagram is part of a larger picture:



Here, back and front faces commute by property (2.8) as fulfilled by any coalgebra morphism over $\overline{u}$. The left top and right top squares commute by assumption (2.10). The left bottom square commutes by interchange, while the right bottom square is a simple identity.

Pasting together the left and right faces, note that $Gf \circ \theta_X$ and $\theta_X \circ Fq$, respectively, form morphisms between coalgebras over h_f. Composing the back face with the pasting of the right faces as well as the pasting of the left faces with the the front face, we get coalgebra morphisms $(\theta_X \circ Ff) \circ s_f$ and $t_f \circ (Gf \circ \theta_X)$ from the coalgebra

$$(Gf \circ t_X, (u_f \circ \langle \mathrm{id}_{FX}, s_X \rangle) \circ (Gf \circ m_X))$$

to the coalgebra

$$(t_Y \circ Ff, n_Y \circ Ff)$$

over the functor h_f. Recalling that the latter coalgebra is terminal, it follows that

$$(\theta_X \circ Ff) \circ s_f = t_f \circ (Gf \circ \theta_X).$$

We have thus verified that the global coalgebra (t, n) is indeed terminal. □

Corollary 2.9 (Dual of Lemma 2.27)**.** *Fix an oplax natural transformation $u : F \times G \to G$ and introduce the functor*

$$\overline{u} : \mathrm{Lax}(F, G) \to \mathrm{Lax}(F, G),$$
$$\overline{u}(t) = u \circ (F \times t) \circ \Delta_F.$$

On a given object $X : \mathcal{B}$, note that u analogously induces a functor

$$\overline{u_X} : FX \to GX,$$
$$\overline{u_X}(t) = u_X \circ (FX \times t) \circ \Delta_{FX}.$$

Recall that for any $X : \mathcal{B}$, if $u_X(C, \cdot) : GX \to GX$ has an initial algebra for any $C : FX$, then so does $\overline{u_X}$ and on an object $C : FX$ it is given by the initial algebra of $u_X(C, \cdot)$. If this assumption is fulfilled for all $X : \mathcal{B}$, then $\overline{u}$ has an initial algebra as well, having the initial algebra of $\overline{u_X}$ as 0-component for any object $X : \mathcal{C}$.

Returning to our original context, we derive:

Corollary 2.10. *Fix a cartesian-closed category $\mathcal{D}$ and a cartesian functor $R : \mathcal{C} \to \mathcal{D}$. Let a functor $S : \mathcal{D} \times \mathcal{C} \to \mathcal{C}$ be given and consider its parametric initial algebra*

$$T : \mathcal{D} \to \mathcal{C},$$
$$T(Y) = \mu X.S(X, Y),$$

assuming it exists. If S is $R \times I$-traversable, then T is R-traversable.

Proof. Apply Corollary 2.9 with $\mathcal{A} = \not\Delta$, $F = R \circ I$, $G = I$ to the oplax natural transformation $(R \circ I) \times I \to I$ that has S as 0-component. □

Definition 2.5. *The inductive type of (syntax for)* regular functors *on a bicartesian-closed category $\mathcal{C}$ is given by (constructors for)*

- *variable selectors $\pi^n_i : \mathcal{C}^n \to \mathcal{C}$ for $i < n$ and n-ary functor composition given $n \in \mathbb{N}$,*
- *the terminal object $1 : 1 \to \mathcal{C}$ and products $\cdot \times \cdot : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$,*
- *the initial object $0 : 1 \to \mathcal{C}$ and coproducts $\cdot + \cdot : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$,*
- *parametric formation of initial algebras.*

Theorem 2.5. *Given a bicartesian-closed category $\mathcal{C}$, all regular functors on $\mathcal{C}$ are traversable.*

Proof. A combination of Lemma 2.24, Lemma 2.25, Corollary 2.8, and Corollary 2.10. □

2.5.2 Practical Internal Language

So far, we have striven to work as much on the level of types as possible. For the remainder of our task, we will have to get our hands dirty, manipulating terms internal to our model and verifying equalities of such terms explicitly. As such, we are going to require a certain library of basic functions. Since the functions we are going to present are rather well-known from practical applications of the lambda calculus in form of functional programming languages, we have opted to omit their definitions, listing only their type signatures. All free type variables are assumed universally quantified, making the respective terms natural transformations.

We start with a series of general purpose operations listed in Figure 2.1. The functor variable F is assumed to denote a regular endofunctor.

One of the most basic types is the Booleans $\mathbf{B} = 1 + 1$. They feature prominently in internalization arguments involving propositional logic. Of course, the internal predicates whose truth value they denote can only ever be computable since they must be expressed in the term

$$
\begin{array}{ll}
\textsc{init}_F & : F(\mu F) \to \mu F \\
\textsc{elim}_A^F & : (F(A) \to A) \to (\mu F \to A) \\
\textsc{eval} & : (A \to B) \times A \to B \\
\textsc{curry} & : (A \times B \to C) \to (A \to B \to C) \\
\textsc{uncurry} & : (A \to B \to C) \to (A \times B \to C) \\
\textsc{const} & : B \to (A \to B) \\
\textsc{swap} & : A \times B \to B \times A \\
\textsc{flip} & : (A \to B \to C) \to (B \to A \to C) \\
\textsc{diag} & : A \to A \times A
\end{array}
$$

Figure 2.1: General purpose operations

$$
\begin{array}{ll}
\textsc{tr}, \textsc{fl} & : \mathbf{B} \\
\textsc{not} & : \mathbf{B} \to \mathbf{B} \\
\cdot \textsc{ and } \cdot & : \mathbf{B} \times \mathbf{B} \to \mathbf{B} \\
\cdot \textsc{ or } \cdot & : \mathbf{B} \times \mathbf{B} \to \mathbf{B} \\
\cdot \textsc{ implies } \cdot & : \mathbf{B} \times \mathbf{B} \to \mathbf{B} \\
\textsc{if} \cdot \textsc{then} \cdot \textsc{else} \cdot & : \mathbf{B}^X \times A^X \times A^X \to A^X
\end{array}
$$

Figure 2.2: Boolean operations

model. Basic operations on the Booleans are listed in Figure 2.2. Saving on notation, we are going to use these operations (sans $\textsc{if} \cdot \textsc{then} \cdot \textsc{else} \cdot$) for any internal Boolean algebra, most notably $A \to \mathbf{B}$ for any type A, not just the Booleans. As such, one would define them over a type class in certain functional programming languages. The Booleans, and more generally any internal Boolean algebra, forms a monoid in two canonical ways, one given by $(\mathbf{B}, \textsc{tr}, \textsc{and})$ and the other one given by $(\mathbf{B}, \textsc{fl}, \textsc{or})$. When not giving any extra details, we will always mean the first.

$$
\begin{array}{ll}
0, 1 & : \mathbf{N} \\
\mathrm{S} & : \mathbf{N} \to \mathbf{N} \\
\textsc{rec}_A & : A \times (A \to A) \to (\mathbf{N} \to A) \\
(\cdot + \cdot), (\cdot - \cdot) & : \mathbf{N} \times \mathbf{N} \to \mathbf{B} \\
(\cdot \leq \cdot), (\cdot < \cdot), (\cdot \geq \cdot), (\cdot > \cdot) & : \mathbf{N} \times \mathbf{N} \to \mathbf{N} \\
\textsc{max}, \textsc{min} & : \mathbf{N} \times \mathbf{N} \to \mathbf{N} \\
(\cdot) \cdot (\cdot) & : \mathbf{N} \times \mathbf{N} \to \mathbf{N} \\
(\cdot)^{\cdot} \text{ or } (\cdot \hat{} \cdot) & : (A \to A) \times \mathbf{N} \to (A \to A)
\end{array}
$$

Figure 2.3: Natural number operations

Next, we have the internal natural numbers defined as $\mathbf{N} = \mu Y.\, 1 + Y$, with operations listed in Figure 2.3. Note that the subtraction operation only gives sensible results for certain arguments. Note that $\mathbf{N}$ forms an internal monoid in several ways, most notably $(\mathbf{N}, 0, +)$

and $(\mathbf{N}, 0, \text{MAX})$. This will feature prominently in traversals, for example in the definition of degrees.

$$\begin{aligned}
\text{NIL} &: \text{List}(X) \\
\cdot :: \cdot &: X \times \text{List}(X) \to \text{List}(X) \\
\text{FOLD}_A &: A \times (X \times A \to A) \to (\text{List}(X) \to A) \\
\cdot \oplus \cdot &: \text{List}(X) \times \text{List}(X) \to \text{List}(X) \\
\text{SINGLETON} &: X \to \text{List}(X) \\
\text{CONCAT} &: \text{List}(\text{List}(X)) \to \text{List}(X) \\
\text{LENGTH} &: \text{List}(X) \to \mathbf{N}
\end{aligned}$$

Figure 2.4: Basic list operations

Recalling the definition $\text{List}(X) = \mu Y.\, 1 + X \times Y$, lists are a generalization of natural numbers. Basic operations can be found in Figure 2.4. Note that $(\text{List}(X), \text{NIL}, \oplus)$ forms a monoid for any $X : \mathcal{C}$. Furthermore, lists form a monad with unit SINGLETON and join CONCAT. Again, both structures will feature in traversals.

$$\begin{aligned}
\text{FILTER} &: (A \to \mathbf{B}) \times \text{List}(A) \to \text{List}(A) \\
\text{ALL, ANY} &: \text{List}(\mathbf{B}) \to \mathbf{B} \\
\text{SUM} &: \text{List}(\mathbf{N}) \to \mathbf{N} \\
\text{COUNT} &: (A \to \mathbf{N}) \times \text{List}(A) \to \mathbf{N} \\
\text{REPEAT} &: \mathbf{N} \times A \to \text{List}(A)
\end{aligned}$$

Figure 2.5: Miscellaneous list operations

Less primitive list operations can be found in Figure 2.5. Recall that List is the primordial traversable functor, so one may give rather brief definitions for many of these operations, e.g. $\text{ALL} = \delta^{\text{List}}_{\mathbf{B}}$ and $\text{SUM} = \delta^{\text{List}}_{(\mathbf{N},0,+)}$. Note here that monoids may be seen as constant applicative functors.

$$\begin{aligned}
\text{GENEQUALS} &: F(\vec{A}) \times F(\vec{B}) \to 1 + F(\vec{A} \times \vec{B}) \\
\cdot \text{ EQUALS } \cdot &: A \times A \to \mathbf{B}
\end{aligned}$$

Figure 2.6: Internal (generalized) equality predicates

The internal (generalized) equality predicates are shown in Figure 2.6. A future subsection will elaborate more on these, where we will use traversals to provide for an elegant definition.

List functions related to the internal equality predicate are listed in Figure 2.7. Note that the function AT requires a morphism $1 \to A$ to be available in its calling context, i.e. has a hidden dummy argument of type A, for otherwise we could not define it in a total way. Some functions only give sensible results when the obvious appropriate preconditions on their arguments are met. For example, we have

$$\text{EQUALS}(i(x), \text{INDEX}(l(x), \text{AT}(i(x), l(x)))) = \text{TR}$$

$$\begin{aligned}
\textsc{at} &: \mathbf{N} \times \mathrm{List}(A) \to A \\
\textsc{index} &: \mathrm{List}(A) \to \mathbf{N} \\
\textsc{prefix} &: \mathbf{N} \times \mathrm{List}(A) \to \mathrm{List}(A) \\
\textsc{isprefix} &: \mathrm{List}(A) \times \mathrm{List}(A) \to \mathbf{B} \\
\textsc{contains} &: A \times \mathrm{List}(A) \to \mathbf{B} \\
\textsc{unique} &: A \times \mathrm{List}(A) \to \mathbf{B} \\
\textsc{distinct} &: \mathrm{List}(A) \to \mathbf{B}
\end{aligned}$$

Figure 2.7: List operations related to equality (possibly requiring it)

for any morphisms $l : X \to \mathrm{List}(A)$ and $i : X \to \mathbf{N}$ such that

$$\begin{aligned}
(\cdot < \cdot) \circ \langle i, \textsc{length} \circ l \rangle &= \textsc{const}(\textsc{tr}), \\
\textsc{unique} \circ \langle \textsc{at} \circ \langle i, l \rangle, l \rangle &= \textsc{const}(\textsc{tr}).
\end{aligned}$$

While left implicit, the proofs of many of these invariants benefit from the internal induction principles presented in the next subsection.

Special attention should be paid to the fact that all equations using the internal logic are always assumed to implicitly quantify universally over their free variables on both sides. For example, the next to last equation formally reads

$$\lambda x.\ \textsc{equals}\,(\textsc{index}(l(x), \textsc{at}(i(x), l(x)))) = \lambda x.\ \textsc{tr}\,.$$

2.5.3 Internal Induction-Like Principles

Let us prove a general structural-induction-like principle for the internal Boolean logic. As always for the internal language, all equations are assumed to implicitly be universally quantified over their free variables. Formally, given an expression

$$S[v_0, \ldots, v_{k-1}] = T[v_0, \ldots, v_{k-1}]$$

between expressions of type B where we have variables $v_i : A_i$ for $i < k$, we interpret the equation as stating that $S = T$ where we interpret S and T as morphisms from $A_0 \times \ldots \times A_{k-1}$ to B.

Lemma 2.28 (Internal Structural Induction)**.** *Consider a traversable functor $F : \mathcal{C} \to \mathcal{C}$ and a function $f : \mu F \to \mathbf{B}$. Assume that the following equation over $F(\mu F) \to \mathbf{B}$ holds:*

$$(\textsc{implies}) \circ \left\langle \delta^F_{\mathbf{B}} \circ F(f), f \circ \textsc{init}_F \right\rangle = \textsc{const}(\textsc{tr}).$$

In pointed notation, this equivalently reads

$$\delta^F_{\mathbf{B}}(F(f)(w))\ \textsc{implies}\ f(\textsc{init}_F(w)) = \textsc{tr}\,.$$

Then, $f = \textsc{const}(\textsc{tr})$.

Proof. Let us define an F-algebra over $\mu F \times \mathbf{B}$ as follows:

$$\begin{aligned}
g &: F(\mu F \times \mathbf{B}) \to \mu F \times \mathbf{B} \\
g &= \left\langle \textsc{init}_F \circ \pi_1, (\textsc{or}) \circ (f \circ \textsc{init}_F) \times \delta^F_{\mathbf{B}} \right\rangle \circ \langle F(\pi_1), F(\pi_2) \rangle
\end{aligned}$$

We claim that

$$\begin{aligned}
\langle \mathrm{id}, f \rangle &: \mu F \to \mu F \times \mathbf{B}, \\
\langle \mathrm{id}, \textsc{const}(\textsc{tr}) \rangle &: \mu F \to \mu F \times \mathbf{B}
\end{aligned}$$

both form algebra morphisms from $\textsc{init}_F$ to g. The claim then follows by initiality of $\textsc{init}_F$ in the category of algebras over F.

One half Let us verify

$$\langle \mathrm{id}, f\rangle \circ \mathrm{INIT}_F = g \circ F\,\langle \mathrm{id}, f\rangle$$

over $F(\mu F) \to \mu F \times \mathbf{B}$. The equality on the first projection holds trivially. For the second projection, we need to check

$$f \circ \mathrm{INIT}_F = (\mathrm{OR}) \circ (f \circ \mathrm{INIT}_F) \times \delta^F_{\mathbf{B}} \circ \langle F(\pi_1), F(\pi_2)\rangle \circ F\,\langle \mathrm{id}, f\rangle$$

over $F(\mu F) \to \mathbf{B}$. Let us rewrite the right-hand side as follows:

$$\begin{aligned}
&(\mathrm{OR}) \circ (f \circ \mathrm{INIT}_F) \times \delta^F_{\mathbf{B}} \circ \langle F(\pi_1), F(\pi_2)\rangle \circ F\,\langle \mathrm{id}, f\rangle \\
&= (\mathrm{OR}) \circ (f \circ \mathrm{INIT}_F) \times \delta^F_{\mathbf{B}} \circ \langle \mathrm{id}, F(f)\rangle \\
&= (\mathrm{OR}) \circ \left\langle f \circ \mathrm{INIT}_F, \delta^F_{\mathbf{B}} \circ F(f)\right\rangle \\
&= (\mathrm{AND}) \circ \left\langle (\mathrm{OR}) \circ \left\langle f \circ \mathrm{INIT}_F, \delta^F_{\mathbf{B}} \circ F(f)\right\rangle, \mathrm{CONST}(\mathrm{TR})\right\rangle
\end{aligned}$$

(using the assumption of the lemma in slightly rewritten form)

$$\begin{aligned}
&= (\mathrm{AND}) \circ \left\langle (\mathrm{OR}) \circ \left\langle f \circ \mathrm{INIT}_F, \delta^F_{\mathbf{B}} \circ F(f)\right\rangle, (\mathrm{OR}) \circ \left\langle f \circ \mathrm{INIT}_F, \mathrm{NOT} \circ \delta^F_{\mathbf{B}} \circ F(f)\right\rangle\right\rangle \\
&= (\mathrm{OR}) \circ \left\langle f \circ \mathrm{INIT}_F, (\mathrm{AND}) \circ \langle \mathrm{id}, \mathrm{NOT}\rangle \circ \delta^F_{\mathbf{B}} \circ F(f)\right\rangle \\
&= (\mathrm{OR}) \circ \langle f \circ \mathrm{INIT}_F, \mathrm{CONST}(\mathrm{FL})\rangle \\
&= f \circ \mathrm{INIT}_F\,.
\end{aligned}$$

Here, we implicitly used some trivial equations for the Boolean connective morphisms.

The other half Let us verify

$$\langle \mathrm{id}, \mathrm{CONST}(\mathrm{TR})\rangle \circ \mathrm{INIT}_F = g \circ F\,\langle \mathrm{id}, \mathrm{CONST}(\mathrm{TR})\rangle$$

Again, the equality on the first projection holds trivially. For the second projection, we need to check

$$\mathrm{CONST}(\mathrm{TR}) \circ \mathrm{INIT}_F = (\mathrm{OR}) \circ (f \circ \mathrm{INIT}_F) \times \delta^F_{\mathbf{B}} \circ \langle F(\pi_1), F(\pi_2)\rangle \circ F\,\langle \mathrm{id}, \mathrm{CONST}(\mathrm{TR})\rangle$$

over $F(\mu F) \to \mathbf{B}$. This immediately simplifies to

$$\mathrm{CONST}(\mathrm{TR}) = (\mathrm{OR}) \circ \left\langle f \circ \mathrm{INIT}_F, \delta^F_{\mathbf{B}} \circ F(\mathrm{CONST}(\mathrm{TR}))\right\rangle.$$

Now just note that

$$\begin{aligned}
(\mathrm{OR}) \circ \left\langle f \circ \mathrm{INIT}_F, \delta^F_{\mathbf{B}} \circ F(\mathrm{CONST}(\mathrm{TR}))\right\rangle &= (\mathrm{OR}) \circ \left\langle f \circ \mathrm{INIT}_F, \delta^F_{\mathbf{B}} \circ F(\mathrm{CONST}(\mathrm{TR}) \circ \mathrm{CONST}(\bullet))\right\rangle \\
&= (\mathrm{OR}) \circ \left\langle f \circ \mathrm{INIT}_F, \mathrm{CONST}(\mathrm{TR}) \circ \delta^F_{\bullet} \circ F(\mathrm{CONST}(\bullet))\right\rangle \\
&= (\mathrm{OR}) \circ \langle f \circ \mathrm{INIT}_F, \mathrm{CONST}(\mathrm{TR})\rangle \\
&= \mathrm{CONST}(\mathrm{TR})
\end{aligned}$$

since $\mathrm{CONST}(\mathrm{TR})$ is a monoid homomorphism from the terminal monoid $\bullet$ to $\mathbf{B}$, exploiting naturality of traversals in the applicative functor argument. □

Note that the statement and proof of the above lemma can in fact be generalized to any Boolean algebra, not just $\mathbf{B}$. This is just a literal transcription, replacing Boolean operations with the corresponding operations of the Boolean algebra. We have chosen to present the proof only for the apparent special case of the Booleans simply to avoid the need of having to introduce new notation.

In particular, note that $\mathbf{B}^A$ forms a Boolean algebra for any type $A : \mathcal{C}$. Reusing the notation for Boolean operations for the corresponding pointwise operations on $\mathbf{B}^A$, we thus obtain a version of the internal induction principle in any context $A : \mathcal{C}$.

Lemma 2.29 (Internal Structural Induction in a Context)**.** *Consider a traversable functor $F : \mathcal{C} \to \mathcal{C}$, an object $A : \mathcal{C}$ to serve as context, and a function $f : \mu F \to \mathbf{B}^A$. Assume that the following equation over $F(\mu F) \to \mathbf{B}^A$ holds:*

$$(\text{IMPLIES}) \circ \left\langle \delta^F_{\mathbf{B}^A} \circ F(f), f \circ \text{INIT}_F \right\rangle = \text{CONST}(\text{TR}).$$

In pointed notation, this equivalently reads

$$\delta^F_{\mathbf{B}^A}(F(f)(w))(a) \text{ IMPLIES } f(\text{INIT}_F(w))(a) = \text{TR}\,.$$

Then, $f = \text{CONST}(\text{TR})$, i.e. $f(v)(a) = \text{TR}$.

Proof. A direct transcription of the proof of Lemma 2.28. □

Note that it would be in general invalid to write the pointed version of the above assumption as

$$\delta^F_{\mathbf{B}}(F(\lambda v.\, f(v)(a))(w)) \text{ IMPLIES } f(\text{INIT}_F(w))(a) = \text{TR}$$

since functor application is not an internal operation. However, the functors we will be dealing with will usually have an associated strength, allowing us to get around this restriction. Using this notation then makes implicit use of this strength.

In another direction, the above principle generalizes canonically to mutual initial algebras.

Lemma 2.30 (Internal Mutual Structural Induction in a Context)**.** *For a finite set M, consider a functor $K : \mathcal{C}^M \to \mathcal{C}^M$ that is componentwise traversable, an object $A : \mathcal{C}$ to serve as context, and a family of functions $f_m : (\mu K)_m \to \mathbf{B}^A$ for $m : M$. Assume that the following equation over $K(\mu K) \to \langle \mathbf{B}^A \rangle_{m:M}$ holds:*

$$(\text{IMPLIES}) \circ \left\langle \delta^K_{\mathbf{B}} \circ K(f), f \circ \text{INIT}_K \right\rangle = \text{CONST}(\text{TR}).$$

In pointed notation, this equivalently reads

$$\delta^{K_m}_{\mathbf{B}^A}(K_m(f)(w))(a) \text{ IMPLIES } f_m(\text{INIT}_{K,m}(w))(a) = \text{TR}$$

for $m : M$. Then, $f = \text{CONST}(\text{TR})$, i.e. $f_m(n)(a) = \text{TR}$ for $m : M$.

Proof. Analogous to the proof of Lemma 2.29. □

Recalling that the internal naturals $\mathbf{N} : \mathcal{C}$ were defined as the initial algebra of a trivially traversable functor, we retrieve the usual form of induction for the internal Boolean logic.

Corollary 2.11 (Internal Induction in a Context)**.** *Consider an object $A : \mathcal{C}$ to serve as context and a function $f : \mathbf{N} \to \mathbf{B}^A$, and assume that the following two equations hold:*

$$f(0) = \text{TR}$$
$$f(n) \text{ IMPLIES } f(\text{S}(n)) = \text{TR}$$

Then, $f = \text{CONST}(\text{TR})$.

2.5.4 Internal Generalized Equality Predicates

Motivating Considerations Any functor $F : \mathcal{D} \to \mathcal{C}$ distributes over products in the sense that

$$\text{split}^F : F \circ (\times) \to (\times) \circ \Delta(F),$$

i.e.

$$\text{split}^F_{X,Y} : F(X \times Y) \to F(X) \times F(Y),$$
$$\text{split}^F_{X,Y} = \langle F(\pi_1), F(\pi_2) \rangle$$

natural in $X, Y : \mathcal{D}$. We are interested in as to what degree $\mathrm{split}^F_{X,Y}$ has an inverse. Since we cannot expect a full affirmation, let us weaken our goal to that of constructing a morphism

$$e^F : (\times) \circ \Delta(F) \to M \circ F \circ (\times),$$

i.e.

$$e^F_{X,Y} : F(X) \times F(Y) \to MF(X \times Y)$$

natural in $X, Y : \mathcal{C}^N$, for some pointed endofunctor $M : \mathcal{C} \to \mathcal{C}$. For the laws, we require the diagram

$$\begin{array}{ccc} F(X \times Y) & \xrightarrow{\mathrm{split}^F_{X,Y}} & F(X) \times F(Y) \\ {\scriptstyle \eta_{F(X\times Y)}}\downarrow & \swarrow_{e^F_{X,Y}} & \vdots\, t \\ MF(X \times Y) & \xrightarrow[M\,\mathrm{split}^F_{X,Y}]{} & M(F(X) \times F(Y)) \end{array} \tag{2.11}$$

to commute where t as defined here needs to have some partial correctness property depending on M.

In what will later turn out to be the principal case $M = 1 + \cdot$, the requirement is that t, as a partial function, acts as the identity whenever defined in the sense of the below Lemma 2.31, i.e. that

$$\lambda z.\, [\lambda \bullet .\, z, \mathrm{id}](t(z)) = \mathrm{id}\,. \tag{2.12}$$

In total, this will make $e^F_{X,Y}$ a partial inverse to $\mathrm{split}^F_{X,Y}$ according to Definition 2.6.

Note that if split^F already happens to be a natural isomorphism for some choice of F, then condition (2.11) determines the choice $e^F = \eta \circ (\mathrm{split}^F)^{-1}$. In particular, we have this case if F is a right adjoint functor, as it will then preserve products. Obviously, this validates condition (2.12) as well.

Partial Inverses Consider two morphisms from a type A to a type B, the first one being partial in the sense of having signature $A \to 1 + B$. The following lemma gives us several equivalent options of making precise the proposition that both morphisms have equal action on defined inputs:

Lemma 2.31. *Consider morphisms $f : A \to B$ and $g : A \to 1 + B$ in a bicartesian-closed category $\mathcal{C}$. Then the following conditons are equivalent:*

$$\lambda a.\, [\lambda \bullet .\, f(a), \mathrm{id}](g(a)) = f, \tag{2.13}$$

$$\lambda a.\, (1 + \mathrm{const}(f(a)))(g(a)) = g. \tag{2.14}$$

Proof. Assume (2.14). Applying the inverted equation to $\lambda k, a.\, [\lambda \bullet .\, f(a), \mathrm{id}](k(a))$, we get

$$\begin{aligned} \lambda a.\, [\lambda \bullet .\, f(a), \mathrm{id}](g(a)) &= \lambda a.\, [\lambda \bullet .\, f(a), \mathrm{id}]((1 + \mathrm{const}(f(a)))(g(a))) \\ &= \lambda a.\, [\lambda \bullet .\, f(a), \mathrm{const}(f(a))](g(a)) \\ &= \lambda a.\, \mathrm{const}(f(a))(g(a)) \\ &= f. \end{aligned}$$

For the reverse implication, assume (2.13). Applying the inverted equation to $\lambda h, a.\, [1 + \mathrm{const}(h(a))](g(a))$, we get

$$\begin{aligned} \lambda a.\, [1 + \mathrm{const}(f(a))](g(a)) &= \lambda a.\, [1 + \mathrm{const}([\lambda \bullet .\, f(a), \mathrm{id}](g(a)))](g(a)) \\ &= \lambda a.\, (\lambda s.\, [1 + \mathrm{const}([\lambda \bullet .\, f(a), \mathrm{id}](s))](s))(g(a)) \\ &= \lambda a.\, \left[\begin{array}{l} \lambda \bullet .\, [1 + \mathrm{const}([\lambda \bullet .\, f(a), \mathrm{id}](\mathrm{inl}(\bullet)))](\mathrm{inl}(\bullet)), \\ \lambda b.\, [1 + \mathrm{const}([\lambda \bullet .\, f(a), \mathrm{id}](\mathrm{inr}(b)))](\mathrm{inr}(b)) \end{array}\right] (g(a)) \\ &= \lambda a.\, [\lambda \bullet .\, \mathrm{inl}(\bullet), \lambda b.\, \mathrm{inr}(b)](g(a)) \\ &= g \end{aligned}$$

where we heavily exploited η-expansion for coproducts to simplify the duplicated occurrence of $g(a)$ in the right-hand side of the first line. □

Definition 2.6. *Consider a morphism $f : A \to B$ in a bicartesian-closed category $\mathcal{C}$. A morphism $g : B \to 1 + A$ is called a* partial inverse *to f if $g \circ f = \mathrm{inr}$ and $f \circ g : B \to 1 + B$ as a partial function behaves as the identity whenever defined in the sense of Lemma 2.31.*

Note that only monomorphisms can have partial inverses.

Lemma 2.32. *In the context of Definition 2.6, the pseudo-inverse g is unique if it exists.*

Proof. Consider two pseudo-inverses $g_1, g_2 : B \to 1 + A$ to f. Since $g \circ f = \mathrm{inr}$ and inr is mono, we know that f, and with it $1 + f$, is mono as well. It thus suffices to prove $(1 + f) \circ g_1 = (1 + f) \circ g_2$.

For $i = 1, 2$, define

$$\begin{aligned} s_i &: B \to \mathbf{B}, \\ s_i &= [\lambda \bullet .\ \textsc{fl}, \mathrm{const}(\textsc{tr})] \circ (1 + f) \circ g_i. \end{aligned}$$

By Lemma 2.31, we have

$$\begin{aligned} (1 + f) \circ g_i = \lambda b.\, (1 + \textsc{const}(b))(((1 + f) \circ g_i)(b)) \quad = \\ \lambda b.\ \textsc{if}\ s_i(b)\ \textsc{then}\ \mathrm{inr}(b)\ \textsc{else}\ \mathrm{inl}(\bullet) \end{aligned} \tag{2.15}$$

It will thus be enough to show that $s_1 = s_2$.

Now note that

$$\begin{aligned} (1 + (1 + f) \circ g_2) \circ (1 + f) \circ g_1 &= (2 + f) \circ (1 + g_2 \circ f) \circ g_1 \\ &= (2 + f) \circ (1 + \mathrm{inr}) \circ g_1 \\ &= (1 + \mathrm{inr}) \circ (1 + f) \circ g_1 \end{aligned}$$

with the usual convention of universally abstracted free variables in place. Precomposing with $[\lambda \bullet .\ \textsc{tr}, \lambda \bullet .\ \textsc{fl}, \textsc{const}(\textsc{tr})]$ and rewriting using (2.15), we get

$$\textsc{if}\ s_1(b)\ \textsc{then}\ (\textsc{if}\ s_2(b)\ \textsc{then}\ \textsc{tr}\ \textsc{else}\ \textsc{fl})\ \textsc{else}\ \textsc{tr} = \textsc{if}\ s_1(b)\ \textsc{then}\ \textsc{tr}\ \textsc{else}\ \textsc{tr},$$

that is

$$\textsc{if}\ s_1(b)\ \textsc{then}\ s_2(b)\ \textsc{else}\ \textsc{tr} = \textsc{tr},$$

or equivalently $s_1(b)\ \textsc{implies}\ s_2(b) = \textsc{tr}$. By symmetry, we also have $s_2(b)\ \textsc{implies}\ s_1(b) = \textsc{tr}$. Combining both assertions with the usual η-rewriting for $\mathbf{B}$, we conclude $s_1 = s_2$. □

Constructing it In the following, we will try to construct e^F subject to the law (2.11) for a regular functor $F : \mathcal{C}^N \to \mathcal{C}$ by structural induction on a compositional representation of the shape of F. Notably, certain cases will require a strengthening of the properties of M: these will be motivated individually. In the end, the requirement will be that M is a strong monad with a zero morphism $1 \to M$. For the choice of the maybe monad $M = 1 + \cdot$, it will turn out that the law (2.12) is also fulfilled. Defining our internal (structural) equality predicate as an irrelevant (i.e. codomain $\mathbf{B}$) version of e, all its expected properties are derivable from (2.11) and (2.12).

Variable selector Let $F : \mathcal{C}^N \to \mathcal{C}$ be the variable selector for a variable $n : N$, i.e. $F = \pi^N_n$. Variable selection is the middle entry of an adjoint triple and thus a right adjoint functor.

Finite products Let $F : \mathcal{C}^I \to \mathcal{C}$ be the finite product of its variables given by the finite parameter index set I, i.e. $F = \prod_I$. Of course, then F is right adjoint to the diagonal functor $\Delta : \mathcal{C} \to \mathcal{C}^I$.

Finite coproducts Let $F : \mathcal{C}^I \to \mathcal{C}$ be the finite coproduct of its variables given by the finite parameter index set I, i.e. $F = \sum_I$. Recall distributivity of products over sums, i.e.

$$\theta_{X,Y} : \sum_I X \times \sum_I Y = \sum_I X \times Y + \sum_{(i,j)\in(I\times I)\setminus\Delta(I)} X_i \times Y_j,$$

where we have split the distributed sum into its diagonal and non-diagonal part. Here, everything is natural in $X, Y : \mathcal{C}^I$. Note that

$$\mathrm{split}^{\sum_I}_{X,Y} = \theta^{-1}_{X,Y} \circ \mathrm{inl}\,.$$

To satisfy condition (2.11), we are thus constrained to set

$$e^{\sum_I} = \left[\eta_{\sum_I X\times Y}, f_{X,Y}\right] \circ \theta_{X,Y}$$

for some

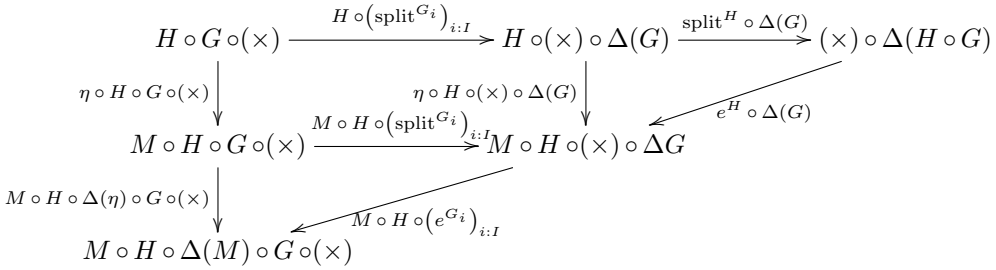
$$f_{X,Y} : \sum_{(i,j)\in(I\times I)\setminus\Delta(I)} X_i \times Y_j \to M\left(\sum_I X \times Y\right).$$

This non-diagonal case proves problematic, and in fact provides the reason for introducing the endofunctor M in the first place. To be able to handle this scenario, we assert the existence of a zero natural transformation $z : 1 \to M$. We then choose

$$f_{X,Y} = z_{\sum_I X\times Y} \circ !$$

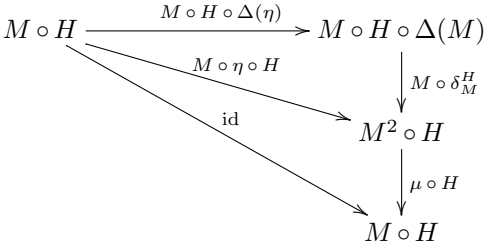
where the latter morphism denotes the unique morphism to the terminal object. Using the usual η-rewriting, one easily checks that condition (2.12) is satisfied for the canonical choice z of the zero morphism for the maybe monad $M = 1 + \cdot$.

Composition Let $F : \mathcal{C}^N \to \mathcal{C}$ be the composition of a regular functor $H : \mathcal{C}^I \to \mathcal{C}$ with regular functors $(G_i)_{i:I} : \mathcal{C}^N \to \mathcal{C}^I$. Assume that we already have e_H and e_{G_i} for $i : I$ subject to the pseudo left-inverse law (2.11). Pasting these given triangular diagrams together, we construct a larger triangle as follows:



Of course, now the composite left vertical morphism is no longer just the unit of M, but a composition of it with something else.

Let us try finding a morphism $M \circ H \circ \Delta(M) \to M$ that cancels out the apparent extra morphism $M \circ H \circ \Delta(\eta)$. First, note that we have available a generalized traversal operation $\delta^H_M : H \circ \Delta(M) \to M \circ H$ if M is applicative. Second, if we require M to actually be a monad, then we have a join operation $\mu : M \circ M \to M$. Let us investigate the coherence requirements needed for the desired cancellation:



(2.16)

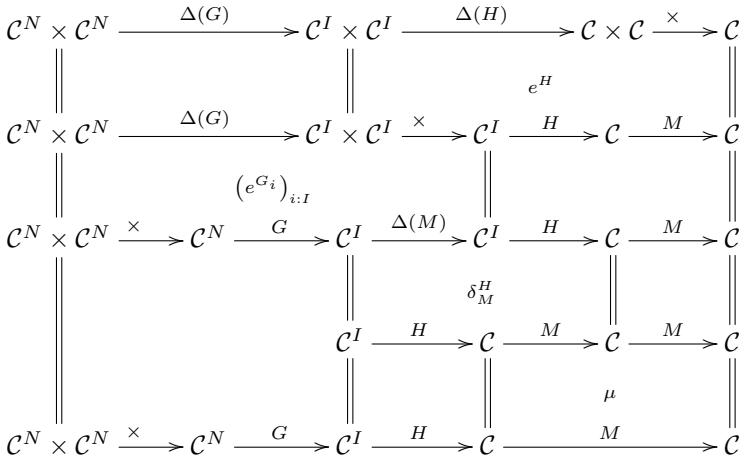


Figure 2.8: An artist's depiction of the compositional structure of $e^{H \circ G}$.

For the lower triangle to commute, it is enough to specify that η is actually the unit of the monad structure we have just postulated, i.e. that we have a monad (M, η, μ). The upper triangle commutes by naturality of traversals in the applicative functor argument if we make η into an applicative morphism from the identity applicative functor to M as an applicative functor. This will hold if we assume M to be strong as a monad and the multiplication of M as an applicative functor be given by the join and strength of M as a monad. In total, we have strengthened the requirements from M being pointed to M being a strong monad. The final expression for e^F such that condition (2.11) is fulfilled is

$$e^{H \circ G} = (\mu \circ H \circ G \circ (\times)) \circ \left(M \circ \delta^H_M \circ G \circ (\times)\right) \circ \left(M \circ H \circ \left(e^{G_i}\right)\right) \circ \left(e^H \circ \Delta(G)\right)$$

as seen in Figure 2.8.

In the particular case of the maybe monad $M = 1 + \cdot$, one checks that condition (2.12) holding for e^H and e^{G_i} with $i : I$ implies condition (2.12) for $e^{H \circ G}$.

Parametric initial algebras Finally, consider the case $F(X) = \mu X'.\, G(X, X')$ with a regular functor $G : \mathcal{C}^{N+1} \to \mathcal{C}$ for which we already have constructed e^G subject to the law (2.11). All following constructions and reasoning steps will be natural in $X, Y : \mathcal{C}^N$. Let u denote the algebra over the functor $G(X, \cdot)$ with carrier $F(Y) \to MF(X \times Y)$ given in uncurried form as the following composition shown in Figure 2.9. With this, we define $e^F_{X,Y}$ as

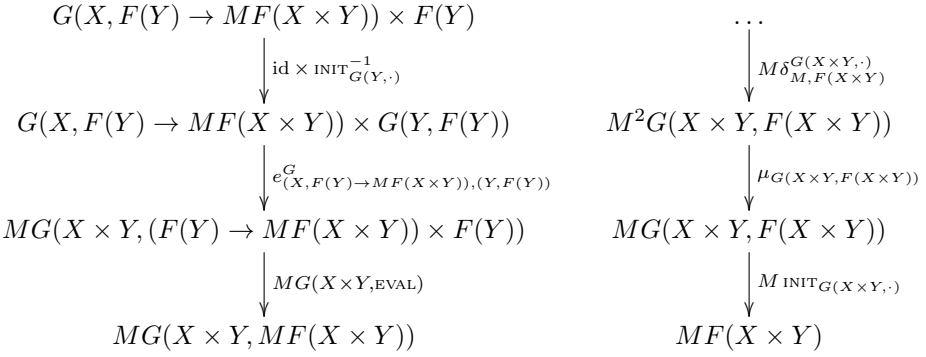


Figure 2.9: Definition of the algebra morphism over $G(X, \cdot)$ with carrier $F(Y) \to MF(X \times Y)$.

the uncurried form of the unique algebra morphism over $G(X,\cdot)$ from the initial algebra to u:

$$e^F_{X,Y} = \mathrm{uncurry}\left(\mathrm{ELIM}^{G(X,\cdot)}_{F(Y)\to MF(X\times Y)}\right).$$

Note that this morphism is natural in $X, Y : \mathcal{C}^N$.

Observe that

$$\begin{aligned}
& e^F_{X,Y} \circ \mathrm{split}^F_{X,Y} \circ \mathrm{INIT}_{G(X\times Y,\cdot)} \\
&= e^F_{X,Y} \circ \left(\mathrm{INIT}_{G(X,\cdot)} \times \mathrm{INIT}_{G(Y,\cdot)}\right) \circ \mathrm{split}^G_{(X,F(X)),(Y,F(Y))} \circ G\left(X\times Y, \mathrm{split}^F_{X,Y}\right) \\
&= \quad M\,\mathrm{INIT}_{G(X\times Y,\cdot)} \circ \mu_{G(X\times Y,F(X\times Y))} \circ M\delta^{G(X\times Y,\cdot)}_{M,F(X\times Y)} \\
&\quad \circ MG(X\times Y, \mathrm{EVAL}) \circ e^G_{(X,F(Y)\to M(F(X\times Y))),(Y,F(Y))} \\
&\quad \circ \left(G\left(X, \mathrm{curry}\left(e^F_{X,Y}\right)\right) \times G(Y,F(Y))\right) \circ \mathrm{split}^G_{(X,F(X)),(Y,F(Y))} \circ G\left(X\times Y, \mathrm{split}^F_{X,Y}\right)
\end{aligned}$$

(by naturality of split^G)

$$\begin{aligned}
&= \quad M\,\mathrm{INIT}_{G(X\times Y,\cdot)} \circ \mu_{G(X\times Y,F(X\times Y))} \circ M\delta^{G(X\times Y,\cdot)}_{M,F(X\times Y)} \\
&\quad \circ MG(X\times Y, \mathrm{EVAL}) \circ e^G_{(X,F(Y)\to M(F(X\times Y))),(Y,F(Y))} \\
&\quad \circ \mathrm{split}^G_{(X,F(X)),(Y,F(Y))} \circ G\left(X\times Y, \left(\mathrm{curry}\left(e^F_{X,Y}\right) \times F(Y)\right) \circ \mathrm{split}^F_{X,Y}\right)
\end{aligned}$$

(by the cancellation law (2.11) for split^G followed by e^G)

$$\begin{aligned}
&= \quad M\,\mathrm{INIT}_{G(X\times Y,\cdot)} \circ \mu_{G(X\times Y,F(X\times Y))} \circ M\delta^{G(X\times Y,\cdot)}_{M,F(X\times Y)} \\
&\quad \circ MG(X\times Y, \mathrm{EVAL}) \circ \eta_{G(X\times Y,(F(Y)\to M(F(X\times Y))\times F(Y))} \\
&\quad \circ G\left(X\times Y, \left(\mathrm{curry}\left(e^F_{X,Y}\right) \times F(Y)\right) \circ \mathrm{split}^F_{X,Y}\right)
\end{aligned}$$

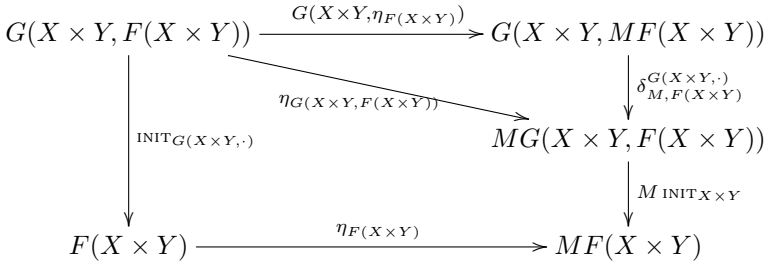
(by naturality properties and unit absorption of the monad (M,η,μ))

$$\begin{aligned}
&= \quad M\,\mathrm{INIT}_{G(X\times Y,\cdot)} \circ \delta^{G(X\times Y,\cdot)}_{M,F(X\times Y)} \circ G(X\times Y, \mathrm{EVAL}) \\
&\quad \circ G\left(X\times Y, \left(\mathrm{curry}\left(e^F_{X,Y}\right) \times F(Y)\right) \circ \mathrm{split}^F_{X,Y}\right) \\
&= \quad M\,\mathrm{INIT}_{G(X\times Y,\cdot)} \circ \delta^{G(X\times Y,\cdot)}_{M,F(X\times Y)} \circ G(X\times Y, M(F(X\times Y))) \\
&\quad \circ G\left(X\times Y, e^F_{X,Y} \circ \mathrm{split}^F_{X,Y}\right),
\end{aligned}$$

yielding the following algebra morphism:

$$\begin{array}{ccc}
G(X\times Y, F(X\times Y)) & \xrightarrow{G(X\times Y, e^F_{X,Y} \circ \mathrm{split}^F_{X,Y})} & G(X\times Y, MF(X\times Y)) \\
\Big\downarrow{\scriptstyle \mathrm{INIT}_{G(X\times Y,\cdot)}} & & \Big\downarrow{\scriptstyle \delta^{G(X\times Y,\cdot)}_{M,F(X\times Y)}} \\
F(X\times Y) & & MG(X\times Y, F(X\times Y)) \\
 & & \Big\downarrow{\scriptstyle M\,\mathrm{INIT}_{X\times Y}} \\
F(X\times Y) & \xrightarrow{e^F_{X,Y} \circ \mathrm{split}^F_{X,Y}} & M(F(X\times Y))
\end{array}$$

On the other hand, we have a simpler morphism



between the same algebras, with commutativity of the triangle following since η is an applicative morphism between applicative functors Id and M — the same reasoning as for diagram (2.16) applies.

Since the domain algebra of both algebra morphisms $e^F_{X,Y} \circ \text{split}^F_{X,Y}$ and $\eta_{F(X\times Y)}$ is the initial one, they must hence be equal. This proves the cancellation law (2.12) for F.

In the particular case of the maybe monad $M = 1 + \cdot$, a similarly tedious calculation verifies that property (2.12) holding for e^G makes it true for e^F as well.

There seems to be opportunity for some refactoring. Certain parts of the construction and verification of laws for the composition of regular functors appear duplicated in the above details for the parametric initial algebra case. How this redundancy may precisely be eliminated eludes us for the moment.

Generalized Equality Predicate Let us now make the particular choice of the maybe monad $M(X) = 1 + X$. Observe that this monad has zero given by $\text{const}(\text{inl}(\bullet)) : \text{Id} \to M$. We may then define the internal generalized equality predicate GENEQUALS^F for any regular functor F:

$$\begin{aligned}&\text{GENEQUALS}^F_{X,Y} : F(X) \times F(Y) \to 1 + F(X \times Y),\\ &\text{GENEQUALS}^F_{X,Y} = e^F,\end{aligned}$$

having exhibited it as the partial inverse to $\text{split}^F_{X,Y}$. Recall from condition (2.12) that

$$[\lambda \bullet . (a, b), \text{split}^F_{X,Y}](\text{GENEQUALS}^F_{X,Y}(a, b)) = (a, b)$$

with the usual convention of universally abstracted free variables. Using Lemma 2.31, we may restate this as

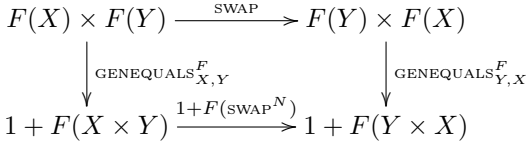
$$(1 + \text{CONST}(a, b))(\text{GENEQUALS}^F_{X,Y}(a, b)) = (1 + \text{split}^F_{X,Y})(\text{GENEQUALS}^F_{X,Y}(a, b)). \tag{2.17}$$

The function GENEQUALS satisfies the expected properties, for example we have the following statement:

Lemma 2.33 (Generalized Symmetry)**.** *Given regular $F : \mathcal{C}^N \to \mathcal{C}$ and $X, Y : \mathcal{C}^N$, we have*

$$(1 + F(\text{SWAP}^N))(\text{GENEQUALS}^F_{X,Y}(a, b)) = \text{GENEQUALS}^F_{Y,X}(b, a).$$

In other words, the following diagram commutes:



Proof. Just observe that $\text{GENEQUALS}^F_{X,Y}$ and $(1 + F(\text{SWAP}^N)) \circ \text{GENEQUALS}^F_{Y,X} \circ \text{SWAP}$ both form partial inverses to $\text{split}^F_{X,Y}$ since the latter commutes trivially with swapping. Conclude by applying Lemma 2.32. □

Equality predicate In case F is of empty arity, i.e. $F : \mathcal{C}^0 \to \mathcal{C}$, note the signature

$$\textsc{genequals}^F_{\bullet,\bullet} : F(\bullet) \times F(\bullet) \to 1 + F(\bullet).$$

We proceed as already evident in the proof of Lemma 2.32 to define the internal equality predicate as a non-relevant special case of the generalized one, setting

$$\begin{aligned} &\cdot \textsc{equals}^A \cdot : \ A \times A \to \mathbf{B}, \\ &\cdot \textsc{equals}^A \cdot = [\lambda \bullet . \ \textsc{fl}, \textsc{const}(\textsc{tr})] \circ \textsc{genequals}^{\lambda \bullet .\, A}_{\bullet,\bullet} . \end{aligned}$$

Note that in contrast to GENEQUALS, we use infix notation for EQUALS.

In this case, note that (2.17) may be given the particular simple form of

$$\textsc{if}\ a\ \textsc{equals}^A\ b\ \textsc{then}\ \mathrm{inr}(a,b)\ \textsc{else}\ \mathrm{inl}(\bullet) = (1 + \textsc{diag})(\textsc{genequals}^{\lambda \bullet .\, A}_{\bullet,\bullet}(a,b)).$$

Applying this equation to either $1 + \pi_1$ or $1 + \pi_2$ to cancel out the occurrence of $1 + \textsc{diag}$, we derive

$$\begin{aligned} \textsc{genequals}^{\lambda \bullet .\, A}_{\bullet,\bullet}(a,b) &= \textsc{if}\ a\ \textsc{equals}^A\ b\ \textsc{then}\ \mathrm{inr}(a)\ \textsc{else}\ \mathrm{inl}(\bullet) \\ &= \textsc{if}\ a\ \textsc{equals}^A\ b\ \textsc{then}\ \mathrm{inr}(b)\ \textsc{else}\ \mathrm{inl}(\bullet), \end{aligned} \tag{2.18}$$

showing (again) that EQUALS and GENEQUALS (for nullary functors) are mutually interdefinable.

The internal equality function satisfies the expected properties. We list a couple of representative statements.

Lemma 2.34 (Reflection)**.** *Consider morphisms $f, g : A \to B$ with B regular. Then from $f(x)\ \textsc{equals}^B\ g(x) = \textsc{tr}$ we can conclude $f = g$.*

Proof. Using the assumption to β-reduce the right-hand sides of (2.18), we derive $\mathrm{inr} \circ f = \mathrm{inr} \circ g$. The conclusion follows since inr is mono. □

Lemma 2.35. *Let $f : A \times B \to C$ and $g : A \to C$ with B regular. Then*

$$\begin{aligned} &\textsc{if}\ b_1\ \textsc{equals}^B\ b_2\ \textsc{then}\ f(a,b_1)\ \textsc{else}\ g(a) \\ =\ &\textsc{if}\ b_1\ \textsc{equals}^B\ b_2\ \textsc{then}\ f(a,b_2)\ \textsc{else}\ g(a) \end{aligned}$$

Proof. Both sides are equal to

$$[\lambda \bullet . \, g(a), f(a,\cdot)](\textsc{genequals}^{\lambda \bullet .\, B}(b_1,b_2))$$

by (2.18). □

Lemma 2.36. *Let $s, t, u : A \times B^2 \to C$ with B regular. Assume that $s(a,b,b) = t(a,b,b)$. Then*

$$\begin{aligned} &\textsc{if}\ b_1\ \textsc{equals}^B\ b_2\ \textsc{then}\ s(a,b_1,b_2)\ \textsc{else}\ u(a,b_1,b_2) \\ =\ &\textsc{if}\ b_1\ \textsc{equals}^B\ b_2\ \textsc{then}\ t(a,b_1,b_2)\ \textsc{else}\ u(a,b_1,b_2) \end{aligned}$$

Proof. Two opposite applications of the previous lemma reduce the goal to

$$\begin{aligned} &\textsc{if}\ b_1\ \textsc{equals}^B\ b_2\ \textsc{then}\ s(a,b_1,b_1)\ \textsc{else}\ u(a,b_1,b_2) \\ =\ &\textsc{if}\ b_1\ \textsc{equals}^B\ b_2\ \textsc{then}\ t(a,b_1,b_1)\ \textsc{else}\ u(a,b_1,b_2), \end{aligned}$$

which is true by assumption. □

Lemma 2.37 (Internal Leibniz property)**.** *Consider a morphism $f : A \times B \to C$ with B and C regular. Then*

$$b_1\ \textsc{equals}^B\ b_2\ \textsc{implies}\ f(a,b_1)\ \textsc{equals}^C\ f(a,b_2) = \textsc{tr}\,.$$

Proof. We rewrite the goal as

$$\text{IF } b_1 \text{ EQUALS}^B b_2 \text{ THEN } (f(a, b_1) \text{ EQUALS}^C f(a, b_2)) \text{ ELSE TR} = \text{IF } b_1 \text{ EQUALS}^B b_2 \text{ THEN TR ELSE TR}$$

and recognize it as a special case of the previous lemma. □

As trivial corollaries, we derive that internal equality forms an equivalence relation in the internal sense.

Lemma 2.38 (Reflexivity)**.** *We have*

$$x \text{ EQUALS}^A x = \text{TR}$$

Proof. By definition. □

Lemma 2.39 (Symmetry)**.** *We have*

$$x \text{ EQUALS}^A y \text{ IMPLIES } y \text{ EQUALS}^A x = \text{TR}\,.$$

Proof. A corollary of Lemma 2.33. Alternatively, we can use the internal Leibniz property □

Lemma 2.40 (Transitivity)**.** *We have*

$$x \text{ EQUALS}^A y \text{ AND } y \text{ EQUALS}^A z \text{ IMPLIES } x \text{ EQUALS}^A z = \text{TR}\,.$$

Proof. Follows from the internal Leibniz property. □

2.5.5 Internal Polynomials

Fix a regular constant $R : \mathcal{C}$ that admits a semiring structure internal to $\mathcal{C}$. Assume there is a decomposition $R = 1 + R'$ with regular $R' : \mathcal{C}$ such that the left summand on the right-hand side is mapped to the zero of the ring R. The *internal polynomials* over R are defined as $\mathrm{Poly}(R) = 1 + \mathrm{List}(R) \times R'$. The intuition here is that a polynomial is a list of (possibly zero) coefficients, and we want polynomials to always be represented by their normal form: the leading coefficient should be non-zero if the polynomial itself is non-zero. This leads to the above representation, where we already give $\mathrm{Poly}(R)$ decomposed into additive unit and other elements as is suitable for iteration of polynomial ring formation. For sake of definiteness, say that coefficients as ordered by degree are represented in traversing order in $\mathrm{List}(R)$. We have obvious encoding and decoding functions

$$\begin{aligned} u &: \mathrm{List}(R) \to \mathrm{Poly}(R), \\ v &: \mathrm{Poly}(R) \to \mathrm{List}(R). \end{aligned}$$

These functions satisfy $u \circ v = \mathrm{id}$, as an easy internal structural induction shows, but not the other way around: lists with final element zero will not be represented. However, defining arithmetic operations over polynomials is easier to be done in $\mathrm{List}(R)$, and we may use u as encoding or $v \circ u$ as normalization function. We may thus implicitly work with polynomials $\mathrm{Poly}(R)$ as if they were represented by $\mathrm{List}(R)$, with the understanding of implicit de- and encoding.

Polynomials $\mathrm{Poly}(R)$ over R form a semiring extension $R \mapsto \mathrm{Poly}(R)$ internal to $\mathcal{C}$ as follows. The embedding is given by mapping an element of R to the singleton list of that element. Additive and multiplicative units of $\mathrm{Poly}(R)$ are prescribed by this embedding.

Addition of polynomials is given by first right-concatenating the arguments with zero lists so as to make them List-structure equal, and then mapping addition over the witness. Unitality and associativity are easily verified by internal induction. On singleton list, this addition coincides with the addition inherited from R. Note that $\mathrm{Poly}(R)$ will have additive inverses if R does, preserving its ring structure.

Multiplication of a polynomial P with a scalar $r : R$ is given by mapping multiplication with r over P. Again, on singleton lists this coincides with the multiplication inherited from R. Note that every polynomial may be uniquely decomposed into a linear combination of monomials with non-zero scalar factors. Here, the *monomial* M_k of degree $k : \mathbb{N}$ is given by the multiplicative unit left-concatenated with a zero list of length k. Anticipating distributivity, it thus suffices to define multiplication of polynomials only for monomials. The product of M_i and M_j with $i, j : \mathbb{N}$ is defined as M_{i+j}. Again, one verifies the laws of unitality and associativity for multiplication and distributivity of multiplication over addition in a straightforward fashion, utilizing repeated internal structural induction.

The polynomial (semi-)ring $\mathrm{Poly}(R)$ satisfies the following universal property: given any other (semi-)ring $S : \mathcal{C}$ internal to $\mathcal{C}$ and an internal (semi-)ring homomorphism $R \to S$ as well as a designated map $x : \mathcal{C}(1, S)$, there is a unique internal (semi-)ring homomorphism $\mathrm{Poly}(R) \to S$ extending the given one such that M_1 is mapped to x.

Internal multivariate polynomials In the context of the preceding paragraph, we may define the *internal multivariate polynomials* $\mathrm{Poly}_M(R)$ over R with respect to a finite parameter index set M. As alluded to above, this proceeds by iteratively forming the internal (semi-)ring of polynomials, with the number of iterations given by the size of M. Simultaneously, one may define an appropriate notion of *(multivariate) monomials* and verify the appropriate universal property of multivariate polynomial rings. In an entirely straightforward but lengthy calculation, one may convince oneself that the concrete isomorphism $M = 1 + \ldots + 1$ used for this iterative construction does not influence the end result, constructing canonical identification internal (semi-)ring isomorphisms between instances of $\mathrm{Poly}_M(R)$ for different decompositions of M and showing that they preserve monomials and universal property data.

Given a polynomial (semi-)ring $\mathrm{Poly}_I(R)$ and a (semi-)ring S, let

$$\textsc{subst} : (R \to S) \times S^I \to (\mathrm{Poly}_I(R) \to S)$$

be the substitution map witnessing the universal property. The resulting map will be a (semi-)ring homomorphism if the first argument is one.

Note that the operation of forming a polynomial (semi-)ring can be regarded not just as an endofunctor over a certain subcategory of (semi-)rings and (semi-)ring homomorphisms, but also as a monad. Concretely, we have a unit

$$\eta_{R[X^I]} : R \to \mathrm{Poly}_I(R)$$

constituting the canonical embedding, and a join operation

$$\begin{aligned} \mu_{R[X^I]} &: \ \mathrm{Poly}_I(\mathrm{Poly}_I(R)) \to \mathrm{Poly}_I(R), \\ \mu_{R[X^I]} &= \textsc{subst}(\mathrm{id}, \mathbf{X}), \end{aligned}$$

fulfilling the expected laws.

2.5.6 Internal Power Series

Fix $R : \mathcal{C}$ admitting a (semi-)ring structure internal to $\mathcal{C}$. Fix a finite parameter index set M. Then $\mathbf{N}^M \to R$ will serve as the base type of our internal representation of power series.

Let describe the basic internal arithmetic operations on power series. We will continue to overload our syntax, reusing the symbols for arithmetic over $\mathbf{N}$.

The constant zero and the addition operator are defined pointwise for power series $\mathbf{N}^M \to R$. They are easily seen to inherit unitality and associativity from R. They same holds for inverse laws in case R has additive inverses.

The finitary product of power series $p_0, \ldots, p_{j-1} : \mathbf{N}^M \to R$ is informally defined as

$$\begin{aligned} p_0 \cdot \ldots \cdot p_{j-1} \quad &: \mathbf{N}^M \to R, \\ (p_0 \cdot \ldots \cdot p_{j-1})(k) &= \sum_{k_0 \leq k} \ldots \sum_{k_{j-1} \leq k} \begin{cases} p_0(k_0) \cdot \ldots \cdot p_{j-1}(k_{j-1}) & \text{if } k_0 + \ldots + k_{j-1} = k, \\ 0 & \text{else.} \end{cases} \end{aligned}$$

For fixed arity j, in particular for $j = 0$ and $j = 2$, this definition is easily seen to be translatable as a λ-term internal to $\mathcal{C}$. As a special case, the constant one has non-zero coefficient only at index 0, with the coefficient being one there. It is straightforward by internal structural induction to verify the laws of unitality and associativity for multiplication, and distributivity of multiplication over addition.

This makes power series $\mathbf{N}^M \to R$ into a (semi-)ring internal to $\mathcal{C}$. However, as already elaborated upon above, they lack an internal equality predicate.

One may define an internal embedding $\mathrm{Poly}_M(R) \to (\mathbf{N}^M \to R)$ of internal polynomials into internal power series using the list accessor operations. Lookup operations exceeding the length of the list of coefficients of the given polynomial return the neutral element 0 of addition of R.

Analogously to case of polynomials, formation of the power series (semi-)ring $\mathbf{N}^I \to R$ can be regarded as an endofunctor over the subcategory of (semi-)rings and (semi-)ring homomorphisms. It also forms a monad, having unit

$$\begin{aligned} \eta_{R[\![X^I]\!]} \quad &: R \to \left(\mathbf{N}^I \to R\right), \\ \eta_{R[\![X^I]\!]}(r)(k) &= \text{IF } k \text{ EQUALS } 0 \text{ THEN } r \text{ ELSE } 0, \end{aligned}$$

and join

$$\begin{aligned} \mu_{R[\![X^I]\!]} \quad &: \left(\mathbf{N}^I \to \mathbf{N}^I \to R\right) \to \left(\mathbf{N}^I \to R\right), \\ \mu_{R[\![X^I]\!]}(q)(k) &= \sum_{\substack{k_1, k_2 \leq k, \\ k_1 + k_2 \text{EQUALS} k}} q(k_1)(k_2). \end{aligned}$$

Note the use of shorthand summation notation for the obvious list monadic generation and filtering followed by a traversal. Again, one may verify the corresponding laws in a more or less straightforward fashion.

2.5.7 Derived Concepts

Similar to the above internalizations, we may derive internal presentation of fields, Euclidean domains, GCD domains, and the field of fractions over GCD domains, where in the latter case we exploit constructive normalizability of fractions. We may internally derive basic theorems of algebra about multivariate polynomials, showing them to exhibit unique factorization. This justifies taking the field of fractions and eventually allows us to define internally what it means for a power series to be algebraic. An explicit development of all these internally concepts would lead to an explosion in size of exposition, with none of the internalization procedures interesting enough to offset this. We have thus only briefly sketched the direction of these technical developments.

2.5.8 Interlude: Classification of Regular Constants

The purpose of this subsection is to sketch how to use the tools developed so far to prove that in the initial model, and thus any model, any regular constant, i.e. any regular functor of arity zero, is isomorphic to either a finite union of unit types or the internal naturals.

By following a similar strategy as in the proof of Theorem 2.1, it can be seen that it is sufficient to consider the following case:

Theorem 2.6. *Fix a finite parameter index set M. Let $S : \mathcal{C}^M \to \mathcal{C}^M$ denote the generalized polynomial successor functor $S(X) = 1 + X$. Consider an arbitrary generalized polynomial functor $F : \mathcal{C}^M \to \mathcal{C}^M$. Then, $\mu(K) = (\mathbf{N}, \ldots, \mathbf{N})$ for $K = S + F$.*

Let us fix notation as in the theorem and develop its proof.

Degrees Let $\mathbf{N}^M : \mathcal{C}^M$ denote the constant vector $\mathbf{N}^M = (\mathbf{N}, \ldots, \mathbf{N})$. Recall that $\mathbf{N}$ forms a monoid in $\mathcal{C}$ with respect to the maximum operator MAX and constant 0. Since K_m is traversable for $m : M$, we have a traversal

$$\begin{aligned} \mathrm{MAX}_K &: K(\mathbf{N}^M) \to \mathbf{N}^M, \\ (\mathrm{MAX}_K)_m &= \delta^{K_m}_{(\mathbf{N},0,\mathrm{MAX})}. \end{aligned}$$

Let the family of *degree* functions denote the unique algebra morphism from μK to the algebra $\mathrm{S}^M \circ \mathrm{MAX}_K$:

$$\begin{aligned} &\mathrm{DEG} : \mu K \to \mathbf{N}^M \\ &\mathrm{DEG} = \mathrm{ELIM}_K \left(\mathrm{S}^M \circ \mathrm{MAX}_K \right). \end{aligned}$$

Values Recall that $\mathrm{List} : \mathcal{C} \to \mathcal{C}$ forms a monad with respect to the constant SINGLETON and the product CONCAT. [18] Note that this makes $\mathrm{List}^M : \mathcal{C}^M \to \mathcal{C}^M$ into a monad on $\mathcal{C}^M$ as well. Fix $m : M$. Since K_m is traversable, we have a traversal

$$\delta^{K_m}_{\mathrm{List},X} : K_m \left(\mathrm{List}^M(X) \right) \to \mathrm{List}\left(K_m(X) \right)$$

with respect to the monad structure natural in $X : \mathcal{C}^M$. Since K_m is polynomial, it has a strength

$$\sigma^{K_m}_{X,Y} : \prod X \times K_m(Y) \to K_m(X \times Y)$$

natural in $X, Y : \mathcal{C}^M$. Mixing these ingredients, we get the *tabulating iterator*

$$\begin{aligned} \mathrm{TAB}_X &: \prod_{m:M} \mathrm{List}(X_m) \to \prod_{m:M} \mathrm{List}\left(K_m(X) \right), \\ \mathrm{TAB}_X(\mathsf{xs})_m &= \bigoplus_{k:K_m(1)} \delta^{K_m}_{\mathrm{List},X} \left(\sigma^{K_m}_{\mathrm{List}^M(X),1}(\mathsf{xs}, k) \right). \end{aligned}$$

Note here that $K(1)$ is isomorphic to a finite coproduct of unit types, justifying the fixed-length concatenation, and that we swept conversion of $\mathrm{List}^M(X) \times 1$ to $\mathrm{List}^M(X)$ under the carpet. Also note that while we have chosen for clarity to make the use of a strength explicit here, we may not always do so in the future. For example, we allow ourselves to write the above definition as

$$\mathrm{TAB}_X(\mathsf{xs})_m = \bigoplus_{k:K_m(1)} \delta^{K_m}_{\mathrm{List},X} \left(K_m \left(\mathrm{CONST}(\mathsf{xs}_i) \right)_{i:M} (k) \right).$$

The tabulating iterator lifts to an endofunction

$$\prod \mathrm{List}^M(\mathrm{INIT}_K) \circ \mathrm{TAB}_{\mu K} : \prod_{m:M} \mathrm{List}\left((\mu K)_m \right) \to \prod_{m:M} \mathrm{List}\left((\mu K)_m \right),$$

enabling us to define the enumeration of values of μK with bounded degree

$$\begin{aligned} \mathrm{VALUES} &: \mathbf{N} \to \prod_{m:M} \mathrm{List}\left((\mu K)_m \right), \\ \mathrm{VALUES}(n) &= \left(\prod \mathrm{List}^M(\mathrm{INIT}_K) \circ \mathrm{TAB}_{\mu K} \right)^n (\mathrm{NIL}, \ldots, \mathrm{NIL}), \end{aligned}$$

[18] As usual, we view a monad on $\mathcal{C}$ as a monoid in the category of endofunctors on $\mathcal{C}$ with respect to functor composition as the monoidal operation.

and the enumeration of values of μK with fixed degree

$$\begin{aligned}\textsc{valuesfix} &: \mathbf{N} \to \prod_{m:M} \mathrm{List}\left((\mu K)_m\right),\\ \textsc{valuesfix}_m(n) &= \textsc{filter}(\lambda v.\ \textsc{deg}_m(v)\ \textsc{equals}\ n, \textsc{values}_m(n)),\end{aligned}$$

Let us show that $\textsc{values}(n)$ contains only values with degree bounded by n. For obvious reasons of length, we will not usually be as explicit as in the proof of this fact. Its main purpose is to show how to execute an internal induction proof in every detail.

Lemma 2.41. *For all $m : M$, we have*

$$\textsc{all}\left(\mathrm{List}\left(\lambda v.\ \textsc{deg}_m(v) \le n\right)\left(\textsc{values}_m(n)\right)\right) = \textsc{tr}\,.$$

Proof. Define $f : \mathbf{N} \to \mathbf{B}$ by

$$\begin{aligned}f(n) &= (\textsc{and})_{m:M} f_m(n)\\ &= (\textsc{and})_{m:M}\ \textsc{all}\left(\mathrm{List}\left(\lambda v.\ \textsc{deg}_m(v) \le n\right)\left(\textsc{values}_m(n)\right)\right).\end{aligned}$$

Our goal is to show $f = \textsc{const}(\textsc{tr})$.

By simple evaluation, we find

$$\begin{aligned}f(0) &= (\textsc{and})_{m:M}\ \textsc{all}\left(\mathrm{List}\left(\lambda v.\ \textsc{deg}_m(v) \le n\right)\left(\textsc{values}_m(0)\right)\right.\\ &= (\textsc{and})_{m:M}\ \textsc{all}\left(\mathrm{List}\left(\lambda v.\ \textsc{deg}_m(v) \le n\right)\left(\textsc{nil}\right)\right.\\ &= (\textsc{and})_{m:M}\ \textsc{all}(\textsc{nil})\\ &= \textsc{tr}\,.\end{aligned}$$

Now, let us try to prove

$$f(n)\ \textsc{implies}\ f(\mathrm{S}(n)) = \textsc{tr}\,.$$

By trivial Boolean logic, it will suffice to fix $m : M$ and prove

$$f(n)\ \textsc{implies}\ f_m(\mathrm{S}(n)) = \textsc{tr}\,.$$

Let us calculate

$$\begin{aligned}&\textsc{all}\left(\mathrm{List}\left(\lambda v.\ \textsc{deg}_m(v) \le \mathrm{S}(n)\right)\left(\textsc{values}_m(\mathrm{S}(n))\right)\right)\\ =\ &\textsc{all}\left(\mathrm{List}\left(\lambda v.\ \textsc{deg}_m(v) \le \mathrm{S}(n)\right)\left(\mathrm{List}(\textsc{init}_{K,m})(\textsc{tab}_{\mu K}(\textsc{values}(n))_m)\right)\right)\\ =\ &\textsc{all}\left(\mathrm{List}\left(\lambda w.\ \textsc{deg}_m(\textsc{init}_{K,m}(w)) \le \mathrm{S}(n)\right)\left(\textsc{tab}_{\mu K}(\textsc{values}(n))_m\right)\right).\end{aligned}$$

At this point, we can trigger a reduction for the recursively defined degree function:

$$\begin{aligned}\lambda w.\ \textsc{deg}_m\left(\textsc{init}_{K,m}(w)\right) \le \mathrm{S}(n) &= \lambda w.\ \mathrm{S}\left(\left(\textsc{max}_K\right)_m\left(K_m(\textsc{deg})(w)\right)\right) \le \mathrm{S}(n)\\ &= \lambda w.\ \left(\textsc{max}_K\right)_m\left(K_m(\textsc{deg})(w)\right) \le n\\ &= (\cdot \le n) \circ \delta^{K_m}_{(\mathbf{N},0,\textsc{max})} \circ K_m(\textsc{deg})\\ &= \delta^{K_m}_{(\mathbf{B},\textsc{tr},\textsc{and})} \circ K_m\left((\cdot \le n) \circ \textsc{deg}_i\right)_{i:M}\end{aligned}$$

since $(\cdot \le n)$ is a monoid homomorphism from $(\mathbf{N}, 0, \textsc{max})$ to $(\mathbf{B}, \textsc{tr}, \textsc{and})$ and thus forms an applicative morphism between the applicative functors that constantly return the respective monoids.

Furthermore, since List can be seen as a functor from $\mathcal{C}$ to the category of monoids in the cartesian-closed category $\mathcal{C}$ and ALL is a monoid homomorphism from $(\mathrm{List}(\mathbf{B}), \textsc{nil}, \oplus)$ to $(\mathbf{B}, \textsc{tr}, \textsc{and})$, we can move the fixed-length concatenation at the start of $\textsc{tab}_{\mu K}$ upwards. Once again, it will suffice to fix $k : K_m(1)$ and prove $f(n)\ \textsc{implies}\ \alpha(n) = \textsc{tr}$ where

$$\alpha(n) = \textsc{all}\left(\mathrm{List}\left(\delta^{K_m}_{(\mathbf{B},\textsc{tr},\textsc{and})} \circ K_m\left((\cdot \le n) \circ \textsc{deg}_i\right)_{i:m}\right)\left(\delta^{K_m}_{\mathrm{List},X}(\sigma^{K_m}_{\mathrm{List}(X),1}(\textsc{values}(n), k))\right)\right).$$

Now we calculate

$$\begin{aligned}
&\textsc{all} \circ \left(\mathrm{List}\left(\delta^{K_m}_{(\mathbf{B},\textsc{tr},\textsc{and})} \circ K_m\left((\cdot \leq n) \circ \textsc{deg}_i\right)_{i:M}\right)\right) \circ \delta^{K_m}_{\mathrm{List},X} \\
&= \textsc{all} \circ \mathrm{List}\left(\delta^{K_m}_{(\mathbf{B},\textsc{tr},\textsc{and})}\right) \circ \delta^{K_m}_{\mathrm{List},X} \circ K_m\left(\mathrm{List}\left((\cdot \leq n) \circ \textsc{deg}_i\right)\right)_{i:M} \\
&= \textsc{all} \circ \delta^{K_m}_{\mathrm{List}(\mathbf{B},\textsc{tr},\textsc{and})} \circ K_m\left(\mathrm{List}\left((\cdot \leq n) \circ \textsc{deg}_i\right)\right)_{i:M} \\
&= \delta^{K_m}_{(\mathbf{B},\textsc{tr},\textsc{and})} \circ K_m\left(\textsc{all} \circ \mathrm{List}\left((\cdot \leq n) \circ \textsc{deg}_i\right)\right)_{i:M}
\end{aligned}$$

by naturality and linearity of traverals, and noting that ALL forms a monoid homomorphism from $\mathrm{List}(\mathbf{B}, \textsc{tr}, \textsc{and})$ to $(\mathbf{B}, \textsc{tr}, \textsc{and})$, with traversals being natural in their applicative functor argument. By naturality of the strength, we thus have

$$\begin{aligned}
\alpha(n) &= \delta^{K_m}_{(\mathbf{B},\textsc{tr},\textsc{and})}\left(\sigma^{K_m}_{\mathbf{B},1}\left(\left(\textsc{all}\left(\mathrm{List}\left((\cdot \leq n) \circ \textsc{deg}_i\right)\left(\textsc{values}_i(n)\right)\right)\right)_{i:M}, k\right)\right) \\
&= \delta^{K_m}_{(\mathbf{B},\textsc{tr},\textsc{and})}\left(\sigma^{K_m}_{\mathbf{B},1}\left(\left(f_i(n)\right)_{i:M}, k\right)\right),
\end{aligned}$$

so

$$\begin{aligned}
f(n) \textsc{ implies } \alpha(n) &= \left((\textsc{and})_{i:M} f_i(n)\right) \textsc{ implies } \delta^{K_m}_{(\mathbf{B},\textsc{tr},\textsc{and})}\left(\sigma^{K_m}_{\mathbf{B},1}\left(\left(f_i(n)\right)_{i:M}, k\right)\right) \\
&= \left((\textsc{and})_{i:M} f_i(n)\right) \textsc{ implies } \delta^{K_m}_{(\mathbf{B},\textsc{tr},\textsc{and})}\left(\sigma^{K_m}_{\mathbf{B},1}\left((\textsc{tr})_{i:M}, k\right)\right).
\end{aligned}$$

In reverse, we reduce

$$\begin{aligned}
&\delta^{K_m}_{(\mathbf{B},\textsc{tr},\textsc{and})}\left(\sigma^{K_m}_{\mathbf{B},1}\left((\textsc{tr})_{i:M}, k\right)\right) \\
&= \delta^{K_m}_{(\mathbf{B},\textsc{tr},\textsc{and})}\left(\sigma^{K_m}_{\mathbf{B},1}\left((\textsc{const}(\textsc{tr})(\bullet))_{i:M}, k\right)\right) \\
&= \delta^{K_m}_{(\mathbf{B},\textsc{tr},\textsc{and})}\left(K_m\left(\textsc{const}(\textsc{tr})\right)_{i:M}\left(\sigma^{K_m}_{\mathbf{B},1}(\bullet, k)\right)\right) \\
&= \textsc{const}(\textsc{tr})\left(\delta^{K_m}_{\bullet}\left(K_m\left(\sigma^{K_m}_{\mathbf{B},1}(\bullet, k)\right)\right)\right) \\
&= \textsc{tr}
\end{aligned}$$

by naturality of the strength and since $\textsc{const}(\textsc{tr})$ is a monoid homomorphism from the terminal monoid $\bullet$ to $(\mathbf{B}, \textsc{tr}, \textsc{and})$, again exploiting naturality of traversals in their applicative functor argument.

It follows that $f(n) \textsc{ implies } \alpha(n) = \textsc{tr}$ and hence, backtracing the chain of reasoning, that

$$f(n) \textsc{ implies } f(\mathrm{S}(n)) = \textsc{tr}\,.$$

We may finally apply Corollary 2.11 and derive $f(n) = \textsc{tr}$. □

Lemma 2.42. *For all $m : M$, we have*

$$\textsc{filter}\left(\lambda v.\ \textsc{deg}_m(v) \leq k, \textsc{values}_m(n)\right) \textsc{ equals } \textsc{values}_m(\textsc{min}(i, n)).$$

Proof. It suffices to show

$$\textsc{filter}\left(\lambda v.\ \textsc{deg}_m(v) \leq k, \textsc{values}_m(k + d)\right) \textsc{ equals } \textsc{values}_m(k).$$

For this, apply internal induction on the naturals, i.e. Corollary 2.11, on k in context $d : \mathbf{N}$. The induction base uses that $\textsc{deg}_m(v) \geq 1$. The induction step is proved in the same fashion is for Lemma 2.41. □

Lemma 2.43. *For all $m : M$, we have*

$$\textsc{unique}(v, \textsc{values}_m(\textsc{deg}_m(v))) = \textsc{tr}\,.$$

Proof. Analogous to the part of the proof of Lemma 2.51 dealing with parametric initial algebras. □

Lemma 2.44. *For all $m : M$, we have*

$$\textsc{length}(\textsc{valuesfix}_m(n)) \geq 1 = \textsc{tr}\,.$$

Proof. A consequence of $K = S + F$. □

Let us now introduce the perhaps weirdly named technical concept of a well-stocked fibration.

Definition 2.7. *A* finitely fibred fibration over $\mathbf{N}$ *of an object $A : \mathcal{C}$ consists of functions*

$$\begin{aligned} d &: A \to \mathbf{N}, \\ e &: \mathbf{N} \to \mathrm{List}(A) \end{aligned}$$

such that the following two equations hold:

$$\textsc{all}(\mathrm{List}(\lambda x.\, d(x)\ \textsc{equals}\ n)(e(n))) = \textsc{tr}, \tag{2.19}$$

$$\textsc{unique}(x, e(d(x))) = \textsc{tr} \tag{2.20}$$

We call such a fibration well-stocked *if in addition there is a function $q : \mathbf{N} \to \mathbf{N}$ such that*

$$q(\mathrm{S}(n)) > \mathrm{S}(q(n)) = \textsc{tr},$$

$$\textsc{length}(e(q(n))) \geq 1 = \textsc{tr}\,.$$

We see that $\textsc{deg}_m$ and $\textsc{values}_m$ form a well-stocked fibration for any $m : M$. The following lemma will now take care of Theorem 2.6:

Lemma 2.45. *For any well-stocked finitely fibred fibration over $\mathbf{N}$ of an object $A : \mathcal{C}$, we have $A = \mathbf{N}$.*

Proof. Let us first introduce some preliminaries. Define contiguous sequences of natural numbers

$$\begin{aligned} \textsc{sequence} &: \mathbf{N} \times \mathbf{N} \to \mathrm{List}(\mathbf{N}), \\ \textsc{sequence}(a, \cdot) &= \textsc{rec}_{\mathrm{List}(\mathbb{N})}(\textsc{nil}, a :: \mathrm{List}(\mathrm{S})(\cdot)). \end{aligned}$$

Note by elementary internal inductions that we alternatively have

$$\textsc{sequence}(a, \cdot) = \pi_2 \circ \textsc{rec}_{\mathbf{N} \times \mathrm{List}(\mathbb{N})}((0, \textsc{nil}), \lambda(b, v).\, v \oplus (a + b :: \textsc{nil})).$$

Some easy to prove properties include

$$\begin{aligned} \textsc{length}(\textsc{sequence}(a, i)) &= i, \\ \textsc{sequence}(a, i) \oplus \textsc{sequence}(a + i, j) &= \textsc{sequence}(a, i + j), \\ \textsc{count}(n, \textsc{sequence}(a, i)) &= \textsc{if}\ n \geq a\ \textsc{and}\ n < a + i\ \textsc{then}\ 1\ \textsc{else}\ 0\,. \end{aligned} \tag{2.21}$$

Let (d, e, q) denote the given well-stocked finitely fibred fibration of some object $A : \mathcal{C}$ over $\mathbf{N}$. Abbreviate

$$u(n) = \textsc{concat}(\mathrm{List}(e)(\textsc{sequence}(0, n))).$$

Our strategy will be to construct a "limiting stream" of u as n goes to infinity. The extra well-stockedness data in form of q enables us to explicitely state a candidate such stream:

$$\begin{aligned} g &: \mathbf{N} \to A, \\ g(n) &= \textsc{at}\left(n, u\left(q\left(\mathrm{S}^2(n)\right)\right)\right). \end{aligned}$$

Recall that the AT operator is only well-defined since we assumed A to come with a default inhabitant that may be returned when the supplied index argument is out of range. This stream will associate to each element of A a unique index, which can be computed by bounded lookup and forms the inverse to the list index accessor function. The challenge — which in this case is admittedly rather elementary since we assumed A to be endowed with an internal equality predicate — is to verify that this all goes through in the internal equational theory with our restricted form of induction with truth values the internal Booleans.

Using well-stockedness We claim that

$$q(n) \geq n = \mathrm{TR}, \tag{2.22}$$

$$\mathrm{LENGTH}(u(q(\mathrm{S}(n)))) \geq n = \mathrm{TR}\,. \tag{2.23}$$

The first claim is shown by trivial internal induction. Let us elaborate on the second claim.

By various elementary monoidal structure preservation properties, we see that

$$\begin{aligned}&\mathrm{LENGTH}(\mathrm{CONCAT}(\mathrm{List}(e)(\mathrm{SEQUENCE}(q(n), q(\mathrm{S}(n)) - q(n)))))\\ &= \mathrm{LENGTH}(e(q(n))) + \mathrm{LENGTH}(\mathrm{CONCAT}(\mathrm{List}(e)(\mathrm{SEQUENCE}(\mathrm{S}(q(n)), q(\mathrm{S}(n)) - q(n) - 1)))).\end{aligned}$$

Note that we used $q(\mathrm{S}(n)) \geq \mathrm{S}(q(n)) = \mathrm{TR}$ in order to have $(q(\mathrm{S}(n)) - q(n) - 1) + 1 = q(\mathrm{S}(n)) - q(n)$. Since

$$\mathrm{LENGTH}(e(q(n))) \geq 1 = \mathrm{TR},$$

this implies

$$\mathrm{LENGTH}(\mathrm{CONCAT}(\mathrm{List}(e)(\mathrm{SEQUENCE}(q(n), q(\mathrm{S}(n)) - q(n))))) \geq 1 = \mathrm{TR}\,.$$

Noting that

$$u(q(\mathrm{S}(n))) = u(q(n)) \oplus \mathrm{CONCAT}(\mathrm{List}(e)(\mathrm{SEQUENCE}(q(n), q(\mathrm{S}(n)) - q(n))))),$$

we then have claim (2.23) by basic arithmetic and internal induction on n.

Stability of u Note that property (2.23) ensures that the call to u will return a list of length at least $\mathrm{S}(n)$, thus ensuring that accessing this list at the n-th position will return a sensible result. However, for the definition of g to make sense, we need to ensure that the result does not depend on this particular arbitrary choice, i.e. that

$$\mathrm{LENGTH}(u(i)) > n \text{ IMPLIES } \mathrm{AT}(n, u(i)) \text{ EQUALS } g(n) = \mathrm{TR}\,. \tag{2.24}$$

Incorporating property (2.23), a more abstract and general way to state our intentions is

$$\mathrm{LENGTH}(u(i)) > n \text{ AND } \mathrm{LENGTH}(u(j)) > n \text{ IMPLIES } \mathrm{AT}(n, u(i)) \text{ EQUALS } \mathrm{AT}(n, u(j)) = \mathrm{TR},$$

which in turn may be generalized to

$$\begin{aligned}&\mathrm{LENGTH}(u(i)) \geq n \text{ AND } \mathrm{LENGTH}(u(j)) \geq n\\ &\text{IMPLIES } \mathrm{PREFIX}(n, u(i)) \text{ EQUALS } \mathrm{PREFIX}(n, u(j)) = \mathrm{TR}\,.\end{aligned}$$

By monotonicity of the antecedent and reverse monotonicity of the consequent, it sufficies to handle the case where we have substituted $\mathrm{MAX}(\mathrm{LENGTH}(u(i)), \mathrm{LENGTH}(u(j)))$ for n. Using monotonicity of $\mathrm{LENGTH} \circ u$, let us break symmetry by introducing a constraint $i \leq j$. We may then write the goal simply as

$$i \leq j \text{ IMPLIES } \mathrm{PREFIX}(\mathrm{LENGTH}(u(i)), u(i)) \text{ EQUALS } \mathrm{PREFIX}(\mathrm{LENGTH}(u(i)), u(j)) = \mathrm{TR},$$

i.e.

$$i \leq j \text{ IMPLIES } u(i) \text{ EQUALS } \mathrm{PREFIX}(\mathrm{LENGTH}(u(i)), u(j)) = \mathrm{TR},$$

i.e.

$$i \leq j \text{ IMPLIES ISPREFIX}(u(i), u(j)) = \text{TR},$$

which is a simple consequence of

$$u(i+t) = u(i) \oplus \text{CONCAT}(\text{List}(e)(\text{SEQUENCE}(i,t))),$$

thus proving (2.24).

Unique containment Let us now state the reverse direction of the isomophism explicitely:

$$\begin{aligned} f &: A \to \mathbf{N}, \\ f(v) &= \text{INDEX}(v, u(\text{S}(d(v)))) \end{aligned}$$

For this definition to make sense, we need to ensure that $\text{UNIQUE}(v, u(\text{S}(d(v)))) = \text{TR}$. A more general way to state this is to say

$$d(v) < n \text{ IMPLIES UNIQUE}(v, u(n)) = \text{TR}\,. \tag{2.25}$$

Note that requirement (2.19) may be reformulated as

$$\text{CONTAINS}(v, e(n)) \text{ IMPLIES } d(v) \text{ EQUALS } n = \text{TR}\,.$$

By case distinction, with our internal decidable equality on the naturals, using condition (2.20) in the affirmative case, we have

$$\text{COUNT}(\cdot \text{ EQUALS } v, e(n)) = \text{IF } d(v) \text{ EQUALS } n \text{ THEN } 1 \text{ ELSE } 0\,.$$

With this, we may rewrite

$$\begin{aligned} &\text{COUNT}(\cdot \text{ EQUALS } v)(u(n))) \\ &= \text{SUM}(\text{List}(\lambda i.\ \text{COUNT}(\cdot \text{ EQUALS } v, e(i)))(\text{SEQUENCE}(0,n))) \\ &= \text{SUM}(\text{List}(\lambda i.\ \text{IF } d(v) \text{ EQUALS } i \text{ THEN } 1 \text{ ELSE } 0)(\text{SEQUENCE}(0,n))) \\ &= \text{COUNT}(d(v), \text{SEQUENCE}(0,n)) \\ &= \text{IF } d(v) < n \text{ THEN } 1 \text{ ELSE } 0\,. \end{aligned}$$

using the usual monoidal preservation properties as well as property (2.21), proving claim 2.25 and general pairwise distrinction of the values of u in the form of

$$\text{MAX}(i,j) < \text{LENGTH}(u(n)) \text{ AND AT}(i, u(n)) \text{ EQUALS AT}(j, u(n)) \text{ IMPLIES } i \text{ EQUALS } j. \tag{2.26}$$

Noting that

$$\begin{aligned} g(m) &= \text{AT}\left(m, u\left(q\left(\text{S}^2(\text{MAX}(m,n))\right)\right)\right), \\ g(n) &= \text{AT}\left(n, u\left(q\left(\text{S}^2(\text{MAX}(m,n))\right)\right)\right) \end{aligned}$$

by property (2.24) and monotonicity of $u \circ q \circ S^2$, we hence prove injectivity of the "stream limit" g of u:

$$g(m) \text{ EQUALS } g(n) \text{ IMPLIES } m \text{ EQUALS } n. \tag{2.27}$$

Verifying the laws Let us first prove $g \circ f = \mathrm{id}_A$. By well-definedness

$$\textsc{unique}(v, u(S(d(v)))) = \textsc{tr}$$

of f, we have

$$f(v) < \textsc{length}(u(S(d(v)))).$$

Using that for the antecedent of property 2.24, it follows that

$$\begin{aligned} g(f(v)) &= \textsc{at}(f(v), u(S(d(v)))) \\ &= \textsc{at}(\textsc{index}(v, u(\mathrm{S}(d(v)))), u(S(d(v)))) \\ &= v, \end{aligned}$$

where the last step again uses well-definedness of f, i.e. property (2.25).

Now let us show $f \circ g = \mathrm{id}_{\mathbf{N}}$. By (2.27), it will be enough to verify $g(f(g(b))) = g(n)$, but this is just a special case of the last paragraph.

□

2.6 The Initial Model

The technique for dealing with the guarded part internally will involve defining an internal notion of listings of possible shapes with the same positions, even though we lack a proper notion of containers. This listing — together with associated completeness properties — may then be used as a lookup structure (using the internal equality predicate), transforming the possible shapes with given positions into easily transferable list indices. Using data extraction and injection techniques closely resembling the framework of shapely functors, we may then construct isomorphisms between guarded regular functors by simply showing that the associated listings have equal lengths for any given cardinality of positions.

Data Extraction and Injection

For ease of presentation, we will only consider the case of a unary regular functors $F : \mathcal{C} \to \mathcal{C}$. The translation of the derived concepts and lemmata is entirely analogous for regular functors with multiple arguments.

Let us first deal with data extraction. Let $U : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ denote the state monad over

$$S(Y) = \mathrm{List}(Y) \to \mathrm{List}(Y).$$

Recall that the functorial structure of U is given by $U(Y, Z) = S(Y) \to Z \times S(Y)$. We have an operation

$$\begin{aligned} \textsc{put}_Y &: Y \to U(Y, 1), \\ \textsc{put}_Y(y)(s) &= (\bullet, s \circ (y :: \cdot)) \end{aligned}$$

natural $Y : \mathcal{C}$. Using this, we may define our data extraction function as

$$\begin{aligned} \textsc{extract}_Y &: F(Y) \to F(1) \times \mathrm{List}(Y), \\ \textsc{extract}_Y &= \langle \mathrm{id}, \textsc{eval}(\cdot, \textsc{nil}) \rangle \circ \textsc{eval}(\cdot, \mathrm{id}) \circ \delta_U^F \circ F(\textsc{put}_Y) \end{aligned}$$

natural in $Y : \mathcal{C}$.

Data injection is a bit more technical due to partiality issues in need of resolution. Let $V : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ denote the state monad transformer over $T(Y) = \mathrm{List}(Y)$ composed with the maybe monad. Recall that the functorial structure of V is given by $V(Y, Z) = T(Y) \to 1 + Z \times T(Y)$. We have an operation

$$\begin{aligned} \textsc{get}_Y &: V(Y, Y), \\ \textsc{get}_Y &= \textsc{init}_{\mathrm{List}}^{-1} \end{aligned}$$

natural in $Y : \mathcal{C}$, extracting the first element of the state list. Note the use of the aborting functionality of the maybe monad to alleviate partiality issues in case the state list is empty. Let further denote

$$\begin{aligned} \textsc{check}_Y &: F(Y) \times \mathrm{List}(Y) \to 1 + F(Y), \\ \textsc{check}_Y(v, t) &= [\![\mathrm{inr}(v), \mathrm{inl}(\bullet)]\!] \left(\textsc{init}_{\mathrm{List}}^{-1}(t)\right) \end{aligned}$$

the operation of Kleisli arrow type over the maybe monad checking that all of the given list data input has been consumed. Using this, we may define our data injection function as

$$\begin{aligned} \textsc{inject}_Y &: F(1) \times \mathrm{List}(Y) \to 1 + F(Y), \\ \textsc{inject}_Y(v, t) &= \left(\delta_V^F \left(F \left(\lambda \bullet \,.\, \textsc{get}_Y\right)(v)\right)(t)\right) ;_{1+.}\, \textsc{check}_Y \end{aligned}$$

natural in $Y : \mathcal{C}$. Here, the intuition is that the function will return in the right summand if the exact amount of data needed to fill the shape argument of type $F(1)$ has been delivered in the list data argument of type $\mathrm{List}(Y)$. Recall that $\cdot ;_{1+.} \cdot$ denotes the monadic bind operation over the maybe monad.

Lemma 2.46. *Injection and extraction form inverses in the following sense. Let* $q : F(1) \times \mathrm{List} \to 1 + F(1) \times \mathrm{List}$ *denote the natural transformation given by*

$$q_Y(v,t) = \text{IF DEG}(v) \text{ EQUALS LENGTH}(t) \text{ THEN } \mathrm{inr}(v,t) \text{ ELSE } \mathrm{inl}(\bullet).$$

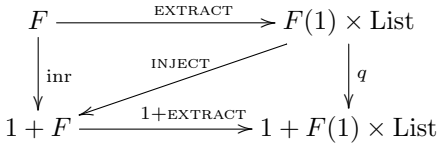
We have

$$\text{INJECT} \circ \text{EXTRACT} = \mathrm{inr}$$

and

$$(1 + \text{EXTRACT}) \circ \text{INJECT} = q.$$

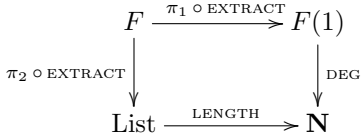
In other words, the following diagram commutes:



Proof. Using the laws of traversals (no internal induction necessary). □

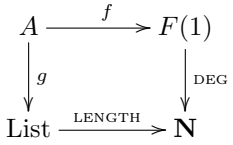
In the presence of richer categorical structure, we would have wished to directly define INJECT over the pullback of the degree and length morphisms into the internal naturals. Even though we lack pullbacks in general, we still recover one in this specific instance. This makes regular functors into shapely functors over any bicartesian-closed category with finitary inductive types. Previous work on shapely functors assumed extensive categories, i.e. stability of coproduct diagrams under pullbacks, as the ambient framework in which to derive this result, an assumption far stronger than we are willing to make.

Lemma 2.47. *The following diagram forms a pullback square:*

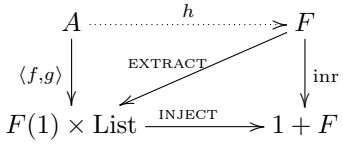


Proof. A more or less direct consequence of the preceeding lemma. □

Lemma 2.48. *Consider* $A : \mathcal{C} \to \mathcal{C}$ *with natural transformations* $f : A \to F(1)$ *and* $g : A \to \mathrm{List}$ *such that the following diagram commutes:*



Then $\text{INJECT} \circ \langle f, g \rangle$ *lifts uniquely through* inr*, i.e. there exists a unique natural transformation* $h : A \to F$ *such that the outer square in the following diagram commutes:*



Furthermore, the upper triangle commutes as well. Together with the preceeding lemma, this makes all of the diagram commute.

Proof. A consequence of the pullback property Lemma 2.47. □

Let us consider the following generalization of Definition 2.7:

Definition 2.8. *Consider types $A, B : \mathcal{C}$ with an internal equality predicate. A* finitely fibred fibration *of A over B consists of functions*

$$\begin{aligned} d &: A \to B, \\ e &: B \to \mathrm{List}(A) \end{aligned}$$

such that the following two equations hold:

$$\textsc{all}(\mathrm{List}(\lambda a.\, d(a) \textsc{ equals } b)(e(b))) = \textsc{tr}, \tag{2.28}$$

$$\textsc{unique}(a, e(d(a))) = \textsc{tr} \tag{2.29}$$

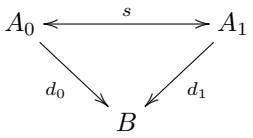
Lemma 2.49. *Assume we are given an object $A_i : \mathcal{C}$ with a finitely fibred fibration*

$$\begin{aligned} d_i &: A_i \to B, \\ e_i &: B \to \mathrm{List}(A_i) \end{aligned}$$

over B for both $i = 0$ and $i = 1$. Assume further that both A_0 and A_1 have a default inhabitant, and that

$$\textsc{length} \circ e_0 = \textsc{length} \circ e_1.$$

Then there exists an isomorphism $s : A_0 \simeq A_1$ of fibrations, i.e. fulfilling commutativity of the following diagram:



Proof. Since we have assumed default inhabitants for A_0 and A_1, we may use list accessing functions. Let us define maps

$$\begin{aligned} s_0 \quad &: A_0 \to A_1, \\ s_0(v) &= \textsc{at}\left(\textsc{index}\left(v, e_0\left(d_0(v)\right)\right), e_1(d_0(v))\right) \end{aligned}$$

and

$$\begin{aligned} s_1 \quad &: A_1 \to A_0, \\ s_1(w) &= \textsc{at}\left(\textsc{index}\left(w, e_1\left(d_1(w)\right)\right), e_0(d_1(w))\right). \end{aligned}$$

First of all, note that the properties of the finitely fibred fibration (d_0, e_0) guarantee that

$$\textsc{index}\left(v, e_0\left(d_0(v)\right)\right) < \textsc{length}(e_0\left(d_0(v)\right)) = \textsc{tr}$$

and thus

$$\textsc{index}\left(v, e_0\left(d_0(v)\right)\right) < \textsc{length}(e_1\left(d_0(v)\right)) = \textsc{tr},$$

guaranteeing that the index lookup and accessing functions in the definition of s_0 behave as expected. In particular, using the properties of the finitely fibred fibration (d_1, e_1), we conclude

$$d_1(s_0(v)) = d_0(v).$$

Symmetrically, the corresponding assertions hold for s_1 as well.

Noting that

$$\begin{aligned}&\textsc{index}\left(\textsc{at}\left(\textsc{index}\left(v, e_0\left(d_0(v)\right)\right), e_1(d_0(v))\right), e_1(d_1(s_0(v)))\right)\\ =\ &\textsc{index}\left(\textsc{at}\left(\textsc{index}\left(v, e_0\left(d_0(v)\right)\right), e_1(d_1(s_0(v)))\right), e_1(d_1(s_0(v)))\right)\\ =\ &\textsc{index}\left(v, e_0\left(d_0(v)\right)\right)\end{aligned}$$

again using completeness of the listing e_1, we have

$$\begin{aligned}s_1(s_0(v)) &= \textsc{at}\left(\textsc{index}\left(v, e_0\left(d_0(v)\right)\right), e_0(d_1(s_0(v)))\right)\\ &= \textsc{at}\left(\textsc{index}\left(v, e_0\left(d_0(v)\right)\right), e_0(d_0(v))\right)\\ &= v\end{aligned}$$

using completeness of the listing e_0. Symmetrically, we verify $s_0 \circ s_1 = \mathrm{id}$, concluding that s_0 and s_1 form an isomorphism $A_0 = A_1$. Preservation of the fibrations d_0 and d_1 has already been shown. □

2.6.1 Listings

Given a regular functor $F : \mathcal{C}^M \to \mathcal{C}$, a *listing* for F with respect to $I \subseteq M$ is a function $e : \mathbf{N}^I \to \mathrm{List}(F(1))$ such that

$$\textsc{all}(\mathrm{List}(\lambda v.\ \textsc{deg}_I(v)\ \textsc{equals}\ k)(e(k))) = \textsc{tr}\,.$$

With this intuition, let us introduce helper functions

$$\begin{aligned}&\textsc{lcontains} \quad : F(1) \times \left(\mathbf{N}^I \to \mathrm{List}(F(1)\right) \to \mathbf{B},\\ &\textsc{lcontains}(v, e) = \textsc{contains}(v, e(\textsc{deg}_I(v)))\end{aligned}$$

and

$$\begin{aligned}&\textsc{lunique} \quad : F(1) \times \left(\mathbf{N}^I \to \mathrm{List}(F(1)\right) \to \mathbf{B},\\ &\textsc{lunique}(v, e) = \textsc{unique}(v, e(\textsc{deg}_I(v))).\end{aligned}$$

Again returning to the context of the listing e, we say that e is *sound* if

$$\textsc{distinct}(e(k)) = \textsc{tr},$$

and that e is *complete* if

$$\textsc{lcontains}(v, e) = \textsc{tr}\,.$$

Together, soundness and completeness synthesize into

$$\textsc{lunique}(v, e) = \textsc{tr}\,.$$

Note that any function $\mathbf{N}^I \to \mathrm{List}(F(1))$ may be transformed into a listing by a filtering operation, removing elements of incorrect degree. If the original function fulfilled the soundness or completeness condition of listings, the resulting listing will be sound or complete correspondingly.

Given a regular functor $F : \mathcal{C}^M \to \mathcal{C}$ and parameter index subsets $I \subseteq J \subseteq M$, one may convert any listing $e : \mathbf{N}^I \to \mathrm{List}(F(1))$ into a listing $\mathbf{N}^I \to \mathrm{List}(F(1))$ by precomposing with restriction to J and then applying the above filtering operation. Noting that $\textsc{deg}_I = \cdot|_I \circ \textsc{deg}_J$, this operation preserves soundness as well as completeness. In this way, construction of a sound and complete listing with respect to $I \subseteq M$ entails sound and complete listings with respect to $J \subseteq M$ such that $I \subseteq J$.

Bounded listing For technical reasons, let us introduce a function that computes a representation for the set of points with non-negative integral coordinates bounded in all components by such a point:

$$\begin{aligned}\underline{(\cdot)} &: \mathbf{N}^I \to \mathrm{List}(\mathbf{N}^I),\\ \underline{k} &= \delta_{\mathrm{List}}^{(\cdot)^I}(\textsc{sequence}(0, \mathrm{S}(k)))\,.\end{aligned}$$

Here, note that I denotes a finite parameter index set.

Given a listing $e : \mathbf{N}^I \to \mathrm{List}(F(1))$ that fulfills at least the first soundness condition, let us introduce the abbreviation

$$\begin{aligned}e^{\leq} &: \mathbf{N}^I \to \mathrm{List}(\mathbf{N}^I),\\ e^{\leq} &= \textsc{concat}\,(\mathrm{List}(e)\,(\underline{k}))\end{aligned}$$

for the derived listing that collects all values bounded by a given degree sequence. It is quite straightforward to verify the assertions

$$\begin{aligned}&\textsc{all}\,(\lambda i.\ \textsc{distinct}(e, i), \underline{k})\ \textsc{implies}\ \textsc{distinct}\left(e^{\leq}(k)\right) = \textsc{tr},\\ &\deg_I(v) \leq k\ \textsc{and}\ \textsc{lcontains}(v, e)\ \textsc{implies}\ \textsc{contains}\left(v, e^{\leq}(k)\right) = \textsc{tr}\,.\end{aligned}$$

Therefore, we have

$$\textsc{distinct}\left(e^{\leq}(k)\right) = \textsc{tr}$$

if the original listing was sound, and

$$\deg(v) \leq k\ \textsc{implies}\ \textsc{contains}\left(v, e^{\leq}(k)\right) = \textsc{tr}$$

if the original listing was complete, and thus

$$\deg(v) \leq k\ \textsc{implies}\ \textsc{unique}\left(v, e(k)^{\leq}\right) = \textsc{tr}$$

if the original listing was sound and complete.

Composition of listings Consider a regular functor $F : \mathcal{C}^M \to \mathcal{C}$ guarded over $I \subseteq M$ that is the composition $F = H \circ G$ of regular functors $G_n : \mathcal{C}^M \to \mathcal{C}$ guarded over I for $n : N$ with a regular functor $H : \mathcal{C}^N \to \mathcal{C}$ guarded over $J \subseteq N$ such that in addition G_j is shielded over I for $j : J$. Given listings $g_n : \mathbf{N}^I \to \mathrm{List}(G_n(1))$ for G_n with $n : N$ and $h : \mathbf{N}^J \to \mathrm{List}(H(1))$ for H, let us construct a listing $f : \mathbf{N}^I \to \mathrm{List}(F(1))$ for F. We want to do this in such a way that the resulting listing will together inherit soundness and completeness from the original listings.

For intuition, let us first play out how to attack this problem in a more pointful setting. Consider a value $v : H(G(1))$. Using the framework for data injection and extraction, we may decompose v first into a shape $s : H(1)$ and data $d_n : \mathrm{List}(G_n(1))$ for $n : N$. The I-degree of v will then be the sum of the I-degrees of the elements of all lists d_n. Note that guardedness implies that the I-degrees of the elements of d_j will be non-zero for $j : J$.

To generate an enumeration of values of F of given I-degree k, we would thus first enumerate over possible shapes s. Note that guardedness implies that the I-degrees of the elements of d_j will be non-zero for $j : J$. We can thus keep this first enumeration step finite by enumerating only over values $s : H(1)$ such that $|\deg_J(s)| \leq |k|$. Next, we would enumerate over the data points of type $G_n(1)$ with $n : N$ for the shape s. For $n : N$, we require $\deg_n(s)$ such data points. Again, this enumeration step can be kept finite by enumerating only over values $d : G_n(1)$ such that $\deg_I(d) \leq k$. Finally, we must check that the total I-degree matches, discarding the candidate otherwise.

An elegant way to put this together into an internal term is given by traversals over the list monad:

$$\begin{aligned}f' &: \mathbf{N}^I \to \mathrm{List}(H(G(1)))\\ f' &= h^{\leq}\,(|k|)_{j:J}\,;\delta_{\mathrm{List}}^{H} \circ H\left(\lambda \bullet .\, g_n^{\leq}(k)\right)_{n:N}\end{aligned}$$

Note that we made implicit use of the strength of H. We may then define

$$\begin{aligned} f &: \mathbf{N}^I \to \mathrm{List}(F(1)) \\ f(k) &= \textsc{filter}(\lambda v.\ \textsc{deg}_I(v) = k, f'(k)). \end{aligned}$$

As remarked earlier, note that f validates the first soundness condition by definition.

Lemma 2.50. *Assume we are given listings g_n for G_n where $n : N$ and h for H. Constructing the listing f for $F = H \circ G$ as above, we have*

$$\left((\textsc{and})_{n:N}\ \textsc{distinct}\left(g_n^{\leq}(k)\right)\right)\ \textsc{and distinct}\left(h^{\leq}(k)\right)\ \textsc{implies distinct}(f(k)) = \textsc{tr}$$

and

$$\textsc{lcontains}(v, f) = \textsc{lcontains}(H(!)(v), h)\ \textsc{and}\ \delta_{\mathbf{B}}^H\left(H\left(\textsc{lcontains}(\cdot, g_n)\right)_{n:N}(v)\right).$$

If g_n for $n : N$ and h are all sound and complete, then the above assertions certainly imply that f is sound and complete.

Lemma 2.51. *If a regular functor $F : \mathcal{C}^M \to \mathcal{C}$ is guarded over $I \subseteq M$, then it has a sound and complete listing e^F with respect to I. If in addition F is shielded over $J \subseteq I$, then*

$$e^F\left(\begin{cases} 0 & \text{if } i \in J, \\ k_i & \text{else} \end{cases}\right)_{i:I} = \textsc{nil}\,.$$

Proof. By mutual structural induction on a compositional representation of F as a guarded regular functor.

For the regular constants $F : \mathcal{C}^0 \to \mathcal{C}$ where $F = 0$ and $F = 1$, it is trivial to verify that $e^F(\bullet) = \textsc{nil}$ and $e^F(\bullet) = \bullet :: \textsc{nil}$ constitute sound and complete listings, respectively. Furthermore, note that the additional assertion for shieldedness is satisfied in the case $F = 0$.

Let $F : \mathcal{C}^2 \to \mathcal{C}$ be the binary sum operation $F(X, Y) = X + Y$. By an initial remark, it suffices to deal with the case $I = \emptyset \subseteq [2]$. We may set $e^F(\bullet) = \mathrm{inl}(\bullet) :: \mathrm{inr}(\bullet) :: \textsc{nil}$ and verify that this constitutes a sound and complete listing. Note that the derived listing $e^F : \mathbf{N}^2 \to \mathrm{List}(F(1))$ for $I = [2]$ fulfills $e^F(0, 0) = \textsc{nil}$. This covers the case where F is viewed as being shielded over $[2]$.

Let $F : \mathcal{C}^2 \to \mathcal{C}$ be the binary product operation $F(X, Y) = X \times Y$. Again, by an initial remark, it suffices to deal with the case $I = \emptyset \subseteq [2]$. We set $e^F(\bullet) = (\bullet, \bullet) :: \textsc{nil}$ and verify that this constitutes a sound and complete listing. Note that the derived listing $e^F : \mathbf{N} \to \mathrm{List}(F(1))$ for both $I = \{0\} \subseteq [2]$ and $I = \{1\} \subseteq [2]$ fulfills $e^F(0) = \textsc{nil}$. This covers the cases where F is viewed as being shielded over either $\{0\}$ or $\{1\}$.

Let $F : \mathcal{C}^M \to \mathcal{C}$ be given by $F(X) = X_m$ for $m : M$. Set

$$e^F(k) = \textsc{if}\ k = \left(\begin{cases} 1 & \text{if i = m,} \\ 0 & \text{else} \end{cases}\right)_{i:I} \textsc{then}\ \bullet :: \textsc{nil}\ \textsc{else}\ \textsc{nil}\,.$$

If J constrained by $I \subseteq J \subseteq M$ contains m, then F is shielded over J. In that case, since $m \in I$, we see that $e^F(0) = \textsc{nil}$.

Composition Next, consider the case $F = H \circ G$ with regular functors $G_n : \mathcal{C}^M \to \mathcal{C}$ for $n : N$ and $H : \mathcal{C}^N \to \mathcal{C}$. Assume that G_n is guarded over $I \subseteq M$ for $n : N$ and that H is guarded over $J \subseteq N$ such that in addition G_j is shielded over I for $j : J$. Recall that this makes F guarded over I. By induction hypothesis, we have sound and complete listings available for G_n with $n : N$ and H. By Lemma 2.50, we derive a sound and complete listing e^F for F.

Now assume further that H is shielded over $T \subseteq J$ and that G_t is shielded over $S \subseteq I$ for $t : T$. Recall that this makes F shielded over S. Fix $k : I$ such that $k|_S = 0$. Referring to

the construction of f' leading to Lemma 2.50, note that $g_t^{\leq}(k) = \text{NIL}$ for $t : T$. Writing H explicitly as a sum over $t : T$ of products with left factors π_t^N and unravelling the construction of traversals for regular functors through this decomposition, we see that

$$\delta^H_{\text{List}} \circ H\left(\lambda \bullet . g_n^{\leq}(k)\right)_{n:N} = \text{CONST}(\text{NIL}),$$

recalling that NIL is a zero for the list monad. It follows that $e^F(k) = \text{NIL}$.

Parametric initial algebras Finally, consider shielded formation of parametric initial algebras. Let $F : \mathcal{C}^M \to \mathcal{C}$ be given by $F(X) = \mu Y.\, G(X, Y)$ for a regular functor $G : \mathcal{C}^{M+1} \to \mathcal{C}$ shielded over $I \subseteq M \subseteq M + 1$. By induction hypothesis, we have a sound and complete listing $e^G : \mathbf{N}^I \to \text{List}(G(1))$ for G fulfilling $e^G(0) = \text{NIL}$. For any functor $H : \mathcal{C}^M \to \mathcal{C}$ and listing h for H with respect to I, note that Lemma 2.50 gives us a way of constructing a listing $l(h) : G \circ \langle \text{Id}, H\rangle$. This mapping is explicitly given as a natural transformation

$$l_H : \left(\mathbf{N}^I \to \text{List}(H(1))\right) \to \left(\mathbf{N}^I \to \text{List}(G(1, H(1)))\right)$$

over regular functors H. Recall that the proof of $l_H(h)$ constituting a listing did not depend on the corresponding assertion for h, but was derived purely from an a posteriori filtering. For the specific composition $G \circ \langle \text{Id}, H\rangle$ and using the induction hypothesis of e^G constituting a sound and complete listing, the assertions of Lemma 2.50 simplify to

$$\text{DISTINCT}\left(h^{\leq}(k)\right) \text{ IMPLIES DISTINCT}(l(h)(k)) = \text{TR} \tag{2.30}$$

and

$$\text{LCONTAINS}(v, l(h)) = \delta_{\mathbf{B}}^{G(1,\cdot)}(G(1, \text{LCONTAINS}(\cdot, h))(v)) \tag{2.31}$$

still assuming that h is a listing.

From l_H, we derive the iteration step endomorphism

$$\begin{aligned} s \quad &: \left(\mathbf{N}^I \to \text{List}(\mu Y.\, G(1, Y))\right) \to \left(\mathbf{N}^I \to \text{List}(\mu Y.\, G(1, Y))\right), \\ s(e) &= \text{List}\left(\text{INIT}_F\right) \circ l_F(e) \end{aligned}$$

and the stream of candidate listings

$$\begin{aligned} f \quad &: \mathbf{N} \to \left(\mathbf{N}^I \to \text{List}(F(1))\right), \\ f_n &= s^n(\text{CONST}(\text{NIL})). \end{aligned}$$

By a simple internal induction using that $l(f_n)$ is a listing, we may strengthen equation 2.30 to

$$\text{DISTINCT}\left(f_n^{\leq}, k\right) \text{ IMPLIES DISTINCT}\left(f_{\text{S}(n)}^{\leq}, k\right) = \text{TR},$$

where we have also used that INIT_F is an isomorphism. Together with the base that the listing CONST(NIL) is sound, a direct application of internal induction shows

$$\text{DISTINCT}\left(f_n^{\leq}(k)\right),$$

i.e. that the listing f_n is sound for all n.

Parametric initial algebras: the first step to completeness Recall the definition of the *iteration count* $c : F(1) \to \mathbf{B}$ for the initial algebra of $G(1, \cdot)$ as $c = \text{ELIM}_{\mathbf{N}}\left(\text{S} \circ \delta_{\mathbf{N},0,\text{MAX}}\right)$:

$$\begin{array}{ccc} G(1, F(1)) & \xrightarrow{G(1,c)} & G(1, \mathbf{N}) \\ \Big\downarrow{\scriptstyle \text{INIT}_{G(1,\cdot)}} & & \Big\downarrow{\scriptstyle \delta^{G(1,\cdot)}_{(\mathbf{N},0,\text{MAX})}} \\ & & \mathbf{N} \\ & & \Big\downarrow{\scriptstyle \text{S}} \\ F(1) & \xrightarrow{\quad c \quad} & \mathbf{N} \end{array} \tag{2.32}$$

Proving a variation of completeness for the sequence of f_n as a whole is more technically involved. With a more classical approach one is used to e.g. from proofs in denotational semantics, one would try to prove the proposition

$$\forall(v).\, c(v) \leq n \text{ IMPLIES } \textsc{lcontains}(v, f_n) = \textsc{tr}$$

by induction over n. In our setting, recall that internal induction is not strong enough to prove quantified propositions. Thus, the above approach does not transfer: we would have to fix v before inducting, in which case by pursuing induction on n one gains absolutely nothings. The same is true for internal structural induction on $v : \mu Y.\, G(1, Y)$, of course.

Instead, we will prove the assertion

$$\textsc{lcontains}(v, f_{c(v)}) = \textsc{tr} \tag{2.33}$$

by internal structural induction on v. We will endevaour after the most elegant technical presentation, staying on as high a level of abstraction as possible.

For this, we first need to observe that the listings f_n are monotonous in n:

$$m \leq n \text{ IMPLIES } \textsc{lcontains}(v, f_m) \text{ IMPLIES } \textsc{lcontains}(v, f_n). \tag{2.34}$$

This is seen by writing $n = m + i$ and internally inducting on i. The internal induction step

$$\textsc{lcontains}(v, f_i) \text{ IMPLIES } \textsc{lcontains}(v, f_{\mathrm{S}(i)}) = \textsc{tr},$$

essentially asserting that l_F is monotonous, is proved by going through the details of the construction of f' before Lemma 2.50 and verifying all involved operations are monotonous in suitable senses.

Expressing monotonicity of the listing function f in a slightly different way, we may write

$$\textsc{lcontains}\left(v, f_{\textsc{max}(m,n)}\right) = \textsc{lcontains}\left(v, f_m\right) \text{ AND } \textsc{lcontains}\left(v, f_n\right)$$

and see that $\textsc{lcontains}(v, \cdot) \circ f$ is a family of monoid homomorphisms from the maximum monoid $(\mathbf{N}, 0, \textsc{max})$ to $(\mathbf{B}, \textsc{tr}, \textsc{and})$. Alternatively, we may say that

$$\lambda n.\ \textsc{lcontains}(\cdot, f_n) : \mathbf{N} \to F(1) \to \mathbf{B}$$

is a monoid homomorphisms from $(\mathbf{N}, 0, \textsc{max})$ to $(\mathbf{B}^{F(1)}, \textsc{tr}, \textsc{and})$.

Let us abbreviate

$$\begin{aligned} u &: F(1) \times F(1) \to \mathbf{B}, \\ u(v, w) &= \textsc{lcontains}\left(v, f_{c(w)}\right), \end{aligned}$$

i.e. $u = \textsc{lcontains} \circ (\mathrm{id} \times (f \circ c))$. We calculate

$$\begin{aligned} \left(u \circ \textsc{diag} \circ \textsc{init}_F\right)(v) &= \textsc{lcontains}\left(\textsc{init}_F(v), f_{\mathrm{S}\left(\delta^{G(1,\cdot)}_{(\mathbf{N},0,\textsc{max})}(G(1,c)(v))\right)}\right) \\ &= \textsc{lcontains}\left(v, l_F\left(f_{\delta^{G(1,\cdot)}_{(\mathbf{N},0,\textsc{max})}(G(1,c)(v))}\right)\right) \end{aligned}$$

(by equation (2.31))

$$= \delta^{G(1,\cdot)}_{\mathbf{B}}\left(G\left(1, \textsc{lcontains}\left(\cdot, f_{\delta^{G(1,\cdot)}_{(\mathbf{N},0,\textsc{max})}(G(1,c)(v))}\right)\right)(v)\right)$$

(by the remark on a monoid homomorphism preceding this calculation, using the laws of traversals)

$$= \delta^{G(1,\cdot)}_{\mathbf{B}}\left(G\left(1, \delta^{G(1,\cdot)}_{(\mathbf{B}^{F(1)},\textsc{fl},\textsc{or})}(G(1, \textsc{curry}(u))(v))\right)(v)\right).$$

This term fits the right-hand side of the internal implication asserted by Corollary 2.12. We deduce that

$$\delta^{G(1,\cdot)}_{\mathbf{B}}\left(G(1, u \circ \textsc{diag})(v)\right) \text{ IMPLIES } \left(u \circ \textsc{diag} \circ \textsc{init}_F\right)(v) = \textsc{tr}\,.$$

By internal structural induction in a context, we therefore conclude $u \circ \textsc{diag} = \textsc{const}(\textsc{tr})$. We finally have verified equation (2.33).

Parametric initial algebras: bounding the iteration count The technical condition of shieldedness allows us to bound the iteration count of values of type $F(1)$ by their degree with respect to I. In formula:

$$c(v) \leq |\textsc{deg}_I(v)| = \textsc{tr}\,. \tag{2.35}$$

This bound will allow us to transform our first approximation (2.33) to completeness into

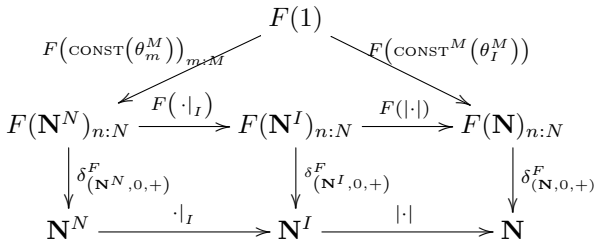
$$\textsc{lcontains}\left(v, f_{|\textsc{deg}_I(v)|}\right) = \textsc{tr}\,. \tag{2.36}$$

by invoking monotonicity (2.34). We may then define the final listing

$$\begin{aligned} f \quad &: \mathbf{N}^I \to \mathrm{List}(F(1)), \\ f(k) &= f_{|k|}(k) \end{aligned}$$

through the obvious diagonalization argument. This denoting a sound listing follows from the individual approximations f_n being sound listings, while (2.36) states that f is complete.

Let us prove equation (2.35) through a standard internal structural induction argument. Given a finite parameter index set T, Let $\theta_S^T : \mathbf{N}^T$ denote the internal characteristic vector of the subset $S \subseteq T$. For a singleton $S = \{s\}$, we write θ_s for θ_S. Commutativity of the following diagram follows from the fact that $\cdot|_I$ and $|\cdot|$ constitute monoid homomorphisms between the designated monoids:



Recalling that $\textsc{deg}_I = (\cdot|_I) \circ \textsc{deg}_N$ where the original degree function $\textsc{deg}_N : F(1) \to \mathbf{N}^k$ is defined as

$$\textsc{deg}_N = \delta^F_{(\mathbf{N}^M,0,+)} \circ F\left(\textsc{const}\left(\theta_m^M\right)\right)_{m:M},$$

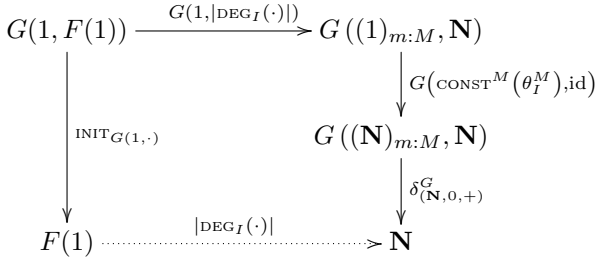
we deduce that

$$|\textsc{deg}_I(\cdot)| = \delta^F_{(\mathbf{N},0,+)} \circ F\left(\textsc{const}^M\left(\theta_I^M\right)\right).$$

Recall from the construction of traversals over regular functors that

$$\delta^{\mu Y.\, G(\mathbf{N},Y)}_{(\mathbf{N},0,+)} = \textsc{elim}^{G(\mathbf{N},Y)}_{\mathbf{N}}\left(\delta_{(\mathbf{N},0,+)}\right).$$

By stability of initial algebras, we hence identify $|\textsc{deg}_I(\cdot)|$ as the following fold:



Comparing this diagram with (2.32), it is easy to see that (2.35) will follow from an application of internal structural induction if we are able to prove

$$(\cdot \leq \cdot) \circ \left\langle \mathrm{S} \circ \delta^{G(1,\cdot)}_{(\mathbf{N},0,\textsc{max})}, \delta^G_{(\mathbf{N},0,+)} \circ G\left(\textsc{const}^M\left(\theta_I^M\right), \mathrm{id}\right) \right\rangle = \textsc{tr}\,.$$

Note that

$$\begin{aligned}
&\delta^{G}_{(\mathbf{N}^I,0,+)} \circ G\left(\textsc{const}^M\left(\theta^M_I\right), \mathrm{id}\right)\\
&= (\cdot + \cdot) \circ \left\langle \delta^{G(\cdot,\mathbf{N})}_{(\mathbf{N},0,+)}, \delta^{G((\mathbf{N})_{m:M},\cdot)}_{(\mathbf{N},0,+)} \right\rangle \circ G\left(\textsc{const}^M\left(\theta^M_I\right), \mathrm{id}\right)\\
&= (\cdot + \cdot) \circ \left\langle \delta^{G(\cdot,1)}_{(\mathbf{N},0,+)} \circ G\left(\textsc{const}^M\left(\theta^M_I\right), \textsc{const}(\bullet)\right), \delta^{G(1,\cdot)}_{(\mathbf{N},0,+)} \right\rangle.
\end{aligned}$$

Since

$$\delta^{G(1,\cdot)}_{(\mathbf{N},0,\textsc{max})}(v) \leq \delta^{G(1,\cdot)}_{(\mathbf{N},0,+)}(v) = \textsc{tr},$$

we are done if we are able to show

$$1 \leq \delta^{G(\cdot,1)}_{(\mathbf{N},0,+)}\left(G\left(\textsc{const}^M\left(\theta^M_I\right), \textsc{const}(\bullet)\right)(v)\right) = \textsc{tr}\,.$$

But this is easily seen to be true by unfolding the construction of traversals for regular functors over the decomposition of G according to its shieldedness over I. □

Let us briefly develop a technical tool needed in the proof of the preceding lemma.

Consider functors $F, G : \mathcal{C} \to \mathcal{C}$ with strengths $\sigma^F_{X,Y} : X \times F(Y) \to F(X \times Y)$ and $\sigma^G_{X,Y} : X \times G(Y) \to G(X \times Y)$. Then there is a natural transformation

$$\begin{aligned}
&m^{F,G}_{X,Y} : F(X) \times G(Y) \to G(F(X \times Y)),\\
&m^{F,G}_{X,Y} = F\left(\sigma^G_{F(X),Y} \circ \textsc{swap}\right) \circ \sigma^F_{G(X),Y} \circ \textsc{swap}
\end{aligned}$$

we call *left multiplication* of F and G, despite the overloaded connotation. There is an analogous right multiplication with codomain $F \circ G$, but since it can be derived from the left multiplication with flipped argument, we will not introduce it explicitly.

Now consider a strong functor $F : \mathcal{C} \to \mathcal{C}$. The *left diagonalization* of F is the natural transformation

$$\begin{aligned}
&d^F_{X,Y} : F(X \times Y) \to F^2(X \times Y),\\
&m^F_{X,Y} = m^{F,F}_{X,Y} \circ \textsc{spl}^F_{X,Y}
\end{aligned}$$

defined using the left multiplication of F with itself. Again, there is an analogous right diagonalization, but it can be derived from the left diagonalization by swapping.

Lemma 2.52. *Consider a strong traversable functor $F : \mathcal{C} \to \mathcal{C}$. Let $u : X \times Y \to \mathbf{B}$ with $X, Y : \mathcal{C}$. Consider the following parallel pair of arrows:*

$$F(X \times Y) \xrightarrow{\qquad F(u) \qquad} F(\mathbf{B}) \xrightarrow{\textsc{all}^F} \mathbf{B}$$

$$F(X \times Y) \xrightarrow{d^F_{X,Y}} F^2(X \times Y) \xrightarrow{F^2(u)} F^2(\mathbf{B}) \xrightarrow{F\left(\textsc{any}^F\right)} F(\mathbf{B}) \xrightarrow{\textsc{all}^F} \mathbf{B}$$

The first arrow p is related to the second arrow q via an internal implication, i.e.

$$\textsc{implies}(p(v), q(v)) = \textsc{tr}\,.$$

Proof. Recall that we have a natural transformation $F \to \mathrm{List}$ preserving strength and traversal operations. Since the codomain of the equation is seen to be free of F, it thus suffices to verify the lemma in the case $F = \mathrm{List}$. Here, the proof is rather straightforward using several internal structural inductions on lists. The intuition is that the element list of $d^{\mathrm{List}}_{X,Y}(v)$ at position i agrees with v at position i for any valid list index i. □

Corollary 2.12. *Consider a strong traversable functor $F : \mathcal{C} \to \mathcal{C}$. Let $u : X \times X \to \mathbf{B}$ with $X : \mathcal{C}$. Making implicit use of the strength, we have*

$$\textsc{all}^F\left(F\left(u \circ \textsc{diag}\right)(v)\right) \textsc{implies} \textsc{all}^F\left(F\left(\textsc{any}^F_{\mathbf{B}^X}\left(F(\textsc{curry}(u))(v)\right)\right)(v)\right) = \textsc{tr}$$

of type $F(X) \to \mathbf{B}$.

Proof. Precompose the statement of Lemma 2.52 with $F(\textsc{diag})$ to get $\textsc{spl}^F_{X,X} \circ F(\textsc{diag}) = \textsc{diag}$. Note further that

$$\textsc{any}^F_{\mathbf{B}^X}\left(F(\textsc{curry}(u))(v)\right)(x) = \textsc{any}^F\left(F(u(\cdot, x))(v)\right)$$

as evaluation at a fixed argument is a monoid homomorphism from $(\mathbf{B}^X, \textsc{fl}, \textsc{or})$ to $(\mathbf{B}, \textsc{fl}, \textsc{or})$, thus constituting an applicative morphism between the constant applicative functors involved in the definition of $\textsc{any}^F_{\mathbf{B}^X}$ and $\textsc{any}^F$ as traversals. The remainder of the argument consists of shifting strengths around. □

Let us finally harvest the fruits of our efforts.

Lemma 2.53. *Consider regular functors $G, H : \mathcal{C}^M \to \mathcal{C}$. If $1 + G = 1 + H$, then $G = H$.*

Proof. Left to the reader. □

Lemma 2.54. *Consider regular functors $F_0, F_1 : \mathcal{C}^M \to \mathcal{C}$ with complete listings*

$$e_0 : \mathbf{N}^M \to \mathrm{List}(F_0(1)),$$
$$e_1 : \mathbf{N}^M \to \mathrm{List}(F_1(1)).$$

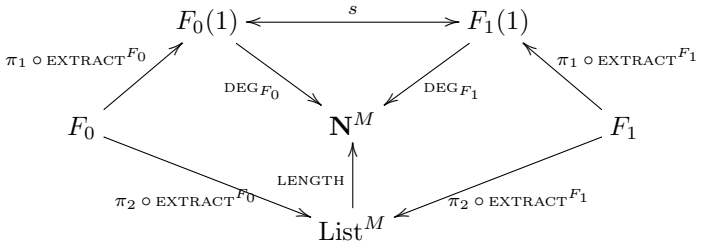
Assume that

$$\textsc{length} \circ e_0 = \textsc{length} \circ e_1.$$

Then $F_0 = F_1$.

Proof. By the following argument, we may without loss of generality assume that both $F_0(1)$ and $F_1(1)$ have a default inhabitant. Replace the functors F_0 and F_1 with $1 + F_0$ and $1 + F_1$, respectively. The complete listings are easily modified to accomodate for this trivial change, and Lemma 2.53 implies that $F_0 = F_1$ will follow from $1 + F_0 = 1 + F_1$.

By Lemma 2.49, we have a degree-preserving isomorphism $s : F_0(1) \to F_1(1)$. Together with Lemma 2.47, this yields the following commutative diagram illustrating the decomposition into shape and data:



Here, the left and right cells form pullback squares. Since pullbacks over isomorphic cospans are isomorphic, it follows that $F_0 = F_1$. □

As a consequence to this result, guarded regular functors $F : \mathcal{C}^M \to \mathcal{C}$ are completely described by their associated power series description $P^F : \mathbf{N}^M \to \mathbf{N}$ where $P^F = \textsc{length} \circ e^F$.

2.6.2 Polynomial Listings

Consider a listing $f : \mathbf{N}^I \to \mathrm{List}(F(1))$ for a regular functor $F : \mathcal{C}^{I+K} \to \mathcal{C}$ guarded over I with the implicit embedding $I \subseteq I + K$. With each value $v : F(1)$, we associate the monomial $m(v) = X^\alpha$ in $\mathrm{Poly}_K(\mathbf{N})$ given by the degree sequence $\alpha = \deg_K(v)$. This way, we derive the *polynomial listing*

$$\begin{aligned}&\overline{f} : \mathbf{N}^I \to \mathrm{Poly}_k(\mathbf{N}),\\ &\overline{f} = \delta^{\mathrm{List}}_{(\mathrm{Poly}_K(\mathbf{N}),0,+)} \circ \mathrm{List}(m) \circ f\end{aligned}$$

associated with f. Note that $\mathbf{N}^I \to \mathrm{Poly}_k(\mathbf{N})$ again forms a semiring with addition computed pointwise. We will continue to overload addition and multiplication symbols. Also note that

$$\delta^{\mathrm{List}}_{(\mathrm{Poly}_K(\mathbf{N}),0,+)} \circ \mathrm{List}(m)$$

forms a monoid homomorphism from $(\mathrm{List}(F(1)), \textsc{nil}, \oplus)$ to $(\mathrm{Poly}_K(\mathbf{N}), 0, +)$.

As will be shown soon, if the listing f is sound and complete, then the regular functor F is characterized by $\overline{f}$ up to equivalence. Let us rephrase the closure of sound and complete listings under all guarded regular functor forming operations as established by Lemma 2.51 in terms of the associated polynomial listings.

Variable Selection Let $F : \mathcal{C}^M \to \mathcal{C}$ be the variable selector $F = \pi^M_m$ for $m : M$. In case $m \in I$, we have

$$\overline{e}^F(k) = \textsc{if}\ k = \begin{pmatrix} 1 & \text{if i = m,} \\ 0 & \text{else} \end{pmatrix}_{i:I} \textsc{then}\ 1\ \textsc{else}\ 0,$$

and in case $m \notin I$, we have

$$\overline{e}^F(k) = \textsc{if}\ k = 0\ \textsc{then}\ \mathbf{X}_m\ \textsc{else}\ 0.$$

Finite sums If $F : \mathcal{C}^{I+K} \to \mathcal{C}$ is given by $F = 0$, then clearly $e^F(k) = \textsc{nil}$ via the nullary decomposition $F = 0 \circ \langle\rangle$ as seen in Lemma 2.51. Using our overloaded semiring notation, we may write $\overline{e}^F = 0$.

Let $F = G + H$ with regular functors $G, H : \mathcal{C}^{I+K} \to \mathcal{C}$ guarded over I. The sound and complete listing e^F for F was derived via the decomposition $F = (+) \circ \langle G, H\rangle$. Since the sound and complete listing associated with $(+)$ is rather trivial, let us unravel the construction from Lemma 2.50 of sound and complete listings for compositions for this particular case. After some straightforward elimination of redundancy, we see that

$$e^F(k) = \mathrm{List}(\mathrm{inl})\left(e^G(k)\right) \oplus \mathrm{List}(\mathrm{inr})\left(e^H(k)\right).$$

Using naturality of traversals in applicative morphisms, this transforms into

$$\overline{e}^F(k) = \overline{e}^G(k) + \overline{e}^H(k),$$

or just $\overline{e}^F = \overline{e}^G + \overline{e}^H$.

Finite products If $F : \mathcal{C}^{I+K} \to \mathcal{C}$ is given by $F = 1$, then clearly

$$e^F(k) = \textsc{if}\ v\ \textsc{equals}\ 0\ \textsc{then}\ \bullet :: \textsc{nil}\ \textsc{else}\ \textsc{nil}$$

via the nullary decomposition $F = 1 \circ \langle\rangle$ as seen in Lemma 2.51. This requires only a very brief unfolding of the listing composition construction from Lemma 2.50. This corresponds directly to the multiplicative unit in our type of polynomial listings. Continuing using our overloaded semiring notation, we write $\overline{e}^F = 1$.

Let $F = G \times H$ with regular functors $G, H : \mathcal{C}^{I+K} \to \mathcal{C}$ guarded over I. The sound and complete listing e^F for F was derived via the decomposition $F = (\times) \circ \langle G, H \rangle$. Again, since the sound and complete listing associated with $(\times)$ is rather trivial, let us unravel the listing composition construction from Lemma 2.50. We see that

$$e^F(k) = \text{FILTER}\left(\text{DEG}_I(\cdot) = k, \mu_{\text{List}}\left(e^{G\leq}(k), e^{H\leq}(k)\right)\right)$$

where μ_{List} denotes the natural transformation witnessing preservation of binary products for the applicative functor List. Since $\text{DEG}_I^{G\times H}(v, w) = \text{DEG}_I^G(v) + \text{DEG}_I^H(w)$, by various naturality properties of traversals it follows that

$$\overline{e}^F(k) = \sum_{\substack{k_g, k_h \leq h, \\ k_g + k_h \,\text{EQUALS}\, k}} \overline{e}^G(k_g) \cdot \overline{e}^H(k_h)$$

where the summation is just shorthand notation for the very verbose expression

$$\begin{aligned} \overline{e}^F = \quad & \text{List}\left(\lambda(k_g, k_h).\, \overline{e}^G(k_g) \cdot \overline{e}^H(k_h)\right) \\ & \circ \text{FILTER}(\lambda(k_g, k_h.\, k_g + k_h \text{ EQUALS } k, \cdot) \\ & \circ \mu_{\text{List}} \circ \text{DIAG} \circ \text{SEQUENCE}(0, \text{S}(\cdot)). \end{aligned}$$

But this is just the definition of multiplication for power series, so that we may conclude $\overline{e}^F = \overline{e}^G \cdot \overline{e}^H$.

In the following, we will continue to use such shorthand notation as above whenever it is clear what internal term it actually stands for.

Parametric algebra formation Let us consider shielded formation of parametric initial algebras. Let $F : \mathcal{C}^M \to \mathcal{C}$ be given by $F(X) = \mu Y.\, G(X, Y)$ with a regular functor $G : \mathcal{C}^{M+1} \to \mathcal{C}$ shielded over $I \subseteq M \subseteq M + 1$. We will represent the polynomial listing associated with e^G as having signature

$$\overline{e}^G : \mathbf{N}^I \to \text{Poly}\left(\text{Poly}_{M \setminus I}(\mathbf{N})\right)$$

Recall from the corresponding section in the proof of Lemma 2.51 that for any functor $H : \mathcal{C}^M \to \mathcal{C}$, we have a way of transforming a listing h for H into a listing $l_H(h)$ for $G \circ \langle \text{Id}, H \rangle$. Let us see how we can factor this map through formation of polynomial listings:

$$\begin{array}{ccc} \mathbf{N}^I \to \text{List}(H(1)) & \xrightarrow{l_H} & \mathbf{N}^I \to \text{List}(G(1, H(1))) \\ \downarrow \overline{\cdot} & & \downarrow \overline{\cdot} \\ \mathbf{N}^I \to \text{Poly}_{M \setminus I}(\mathbf{N}) & \xrightarrow{\bar{l}} & \mathbf{N}^I \to \text{Poly}_{M \setminus I}(\mathbf{N}) \end{array}$$

Note that we expect the map $\bar{l}$ to end up no longer depending on the functor H. Using monadic notation for power series formation, our candidate for the desired map is

$$\bar{l}(h) = \mu_{\text{Poly}_{M \setminus I}[\![X^I]\!]}\left(\text{SUBST}\left(\eta_{\text{Poly}_{M \setminus I}[\![X^I]\!]}, h\right) \circ \overline{e}^G\right).$$

The verification process is not particular surprising, being yet another study in the practical application of the laws for traversals. In the interest of brevity, we skip it.

Note that the polynomial listing associated with $e^{\mu Y.\, G(\cdot, Y)}$ will be a fixpoint of $\bar{l}$. Using the fact that G is shielded over I, i.e. that $\overline{e}^G(0) = 0$, we may see — in a fashion similar to the corresponding part of the proof of Lemma 2.51 — that $\bar{l}(h)$ is a contractive map in the sense that

$$((\text{ALL})_{t<k}(h_1)(t) \text{ EQUALS } (h_2)(t)) \text{ IMPLIES } \bar{l}(h_1)(k) = \bar{l}(h_2)(k) = \text{TR}$$

where the internal finite Boolean quantification is again shorthand notation for the obvious proper internal term. We may thus construct a fixpoint explicitly by for example choosing

$$\begin{aligned}\overline{e}^F &: \mathbf{N}^I \to \mathrm{Poly}_{M\setminus i}(\mathbf{N}),\\ \overline{e}^F(k) &= \bar{l}^{|k|}(0)(k),\end{aligned}$$

observing that $\bar{l}(h)(0) = 0$. An internal version of standard fixpoint theory then proves that this fixpoint is in fact unique, i.e. given f such that $\bar{l}(f) = f$, then $f = \overline{e}^F$.

2.6.3 Verifying Algebraicity of Polynomial Listings

By another induction on the structure of guarded regular functors, we may verify that our polynomial listings fulfill the same polynomial equations as did the associated power series in the case of the set model. This involves an internalization of the arguments presented in the corresponding section, using the basic algebraic operations and properties presented and alluded to earlier.

2.7 Related Work

Decidability and complexity of type isomorphism is a rich and broad subject with relations to automata theory, combinatorics, mathematical logic, rewriting theory, and type theory. In the specific setting of typically ambiguous type isomorphism in a simply typed λ-calculus with sums, products, and recursive types, our results perhaps thematically come closest to work by Fiore [18], which expands on more expository work of the same character like the problem of 'Seven Trees in One' [8]. The crucial difference however, is that Fiore's recursive types are generically recursive instead of being given by initial algebras, leading to a much weaker — though more structural — type-level equational theory based only on the basic isomorphism $\mu F \cong F(\mu F)$ for folding/unfolding, i.e. one-step expanding, recursive types, with isomorphism decision problems reducing to problems of equality in quotients of polynomial semirings. [19] In constrast, the term-level η-equations generated in our settings by uniqueness of eliminators from initial algebras yield a richer, more complicated theory of type isomorphisms.

From a slightly different angle, Backhouse et al. [6] relate recursive type isomorphism in a theory with sums and products to Solitaire-like games and other tiling problems. They show how an understanding of decision procedures for recursive type isomorphism classify solution strategies. Again, their recursive types are generically recursive and the type isomorphisms structural, placing their work closer to Fiore's [18] than ours.

Moving to a semantic notion of type isomorphism (though retaining the initial algebra semantics for recursive types), our results relate to automata theory. Specifically in the set model, as already observed by Altenkirch [4], type isomorphism corresponds to equivalence of context free grammars when allowing terminal symbols to commute and counting the number of possible derivations for each word, known as the ambiguity. Both context free grammars with commuting terminals [46] and context free grammars with count of ambiguity [19] have separately been considered, though we are not aware of any work that combines the two modifications. We again note that the restriction of our situation to indefinite types reduces to Parikh's theorem [46] in the set model.

With regards to possibly novel technical means deployed in our development, our methodology of establishing type-level isomorphisms without having to reason about term equality whenever possible follows the spirit of — and extends where necessary — the μ-calculus of Backhouse et al. [5]. Let us briefly comment on two further aspects.

Traversability of regular functors Our work on traversability of regular functors in any bicartesian-closed category with suitable initial algebras builds on work by Jaskelioff and Rypacek [30]. Without their making rigorous of the categorical meaning of traversability, our results could not have even be stated properly. Previous work on establishing traversability of regular functors relied either on working on very special categories like the category of sets Set [30] where initial algebras can conveniently be descibed as colimits and the framework of containers [1] unfolds much of its power, or was restricted to traversing only with respects to monads [20] instead of general applicative functors. The latter restriction was seemingly born out of the oversight of generalizing to simultaneous traversals in several parameters. The ad hoc fix of requiring a monad structure made the treatment unnecessarily convoluted by mixing together two orthogonal concerns, requiring extra structure alignment requirements [20, Paragraph 5.1] on the monad.

Integrals of quotient containers Our classification result on integrals of cycle quotient containers [3, 22] relates to work by Joyal [32] on combinatorial species. It is superseded by Rajan [49].

[19] This is not to be confused with the use of 'fold' for eliminators from initial algebras in the functional programming community.

Chapter 3

Higher Homotopies in Univalent Universes

3.1 Introduction

Homotopy type theory [56], closely related to the Univalent Foundations program instigated by Vladimir Voevodsky, is a relatively recent interpretation of type theory that introduces a new way of looking at Martin-Löf identity types [42]. Traditionally, these are defined as inductive types $\mathsf{Id}_A(x, y)$ over two indices x, y of a given type A with a canonical inhabitant $\mathsf{refl}_a : \mathsf{Id}_A(a, a)$ for definitionally equal indices. Originally, with this notion of equality, it was unknown whether it is provable that all inhabitants of $\mathsf{Id}_A(x, y)$ are equal, an assertion known as Uniqueness of Identity Proofs (UIP). In the habilitation thesis [55], Streicher presented heuristic evidence [1] that UIP is not derivable, and suggested that to remedy this apparent accident it should be assumed in equivalent form as an additional axiom K [2] allowing us to eliminate over elements of $\mathsf{Id}_A(a, a)$ with $a : A$ fixed. In a seminal paper [27], Hofmann and Streicher later constructed a model of type theory with one universe, the groupoid model, that invalidates axiom K, formally showing that UIP is not derivable.

In contrast, homotopy type theory asserts that the lack of UIP is not a fault, but a feature. An intuitive model interprets types as topological spaces; elements of a type as points in that space varying continuously with respect to the context; and the identity type $\mathsf{Id}_A(x, y)$ as the space of paths between points x and y in the space A. Any space with non-trivial fundamental group, i.e. having loops that cannot be continuously contracted to a point, like the circle will then invalidate UIP.

In the terminology of homotopy type theory, a type equivalent to the unit type is known as *contractible*. A type such that every two of its elements are equal is known as *propositional*. A type that validates UIP is known as a *set*, or 0-type. Proponents of homotopy type theory argue that the usefulness of UIP when working with algebraic structures can be recovered by restricting oneself to the subuniverse of types that are sets.

Homotopy type theory introduces a more structural view of the identity type. Under a suitable notion of equivalence, equality of (dependent) pairs is essentially a (dependent) pair of equalities. Similarly, equality of (dependent) functions should be equivalent to a dependent function valued in equalities, an assumption known as function extensionality. Going further, homotopy type theory inquires about the suitable structural equality on type universes. The correct notion of equality of two types turns out to be equivalence itself. This assumption, known as Voevodsky's Univalence Axiom (UA) [58], in fact implies function extensionality and directly contradicts UIP in the presence of a universe. It is, however, validated by the groupoid model and other homotopy theoretic models like the simplicial set model [33] and Coquand's

[1] in terms of failed approaches and philosophical reasoning

[2] named in reference to the standard identity type eliminator J

et al. cubical set model [7]. Equality of (co-)inductive types then turns out to be equivalent to a (co-)inductive type of equalities of the same structure.

One is inclined to give equality a more primitive, structural status [3], but the general question of computation rules for equality is still under debate. This question is related to the problem of recovering a version of canonicity: under the Univalence Axiom, no longer does every closed term $s : \mathbf{2}$ of Boolean type reduce, i.e. is definitionally equal, to either $0_{\mathbf{2}}$ or $1_{\mathbf{2}}$. However, it is expected that there is an algorithm for deriving a closed inhabitant of either $\mathsf{Id}_{\mathbf{2}}(s, 0_{\mathbf{2}})$ or $\mathsf{Id}_{\mathbf{2}}(s, 1_{\mathbf{2}})$.

In this chapter, we present a solution to an open problem from the special year on Univalent Foundations of Mathematics at the Institute for Advanced Study 2012/2013. An n-type is a type such that all higher equalities above level n are trivial. In the context of a hierarchy of cumulative universes $\mathcal{U}_0, \mathcal{U}_1, \ldots$, is it possible to construct types of arbitrarily high truncation level? [4] Specifically, we show that $\mathcal{U}_n^n$, the n-th universe restricted to n-types, has truncation level strictly $n + 1$.

An Agda development [37] of the contents of this chapter is available.

3.2 Type Theory

We work in a type theory essentially identical to the one described in appendix $A.2$ of the current main reference in the subject of homotopy type theory [56]. Rule names and most naming conventions have deliberately been left in place. Exploiting an infinite hierarchy of universes, we are able to restrict ourselves to only three basic types of judgements: context wellformedness $\Gamma\ \mathsf{ctx}$, term typing $\Gamma \vdash a : A$, and typed definitional equality $\Gamma \vdash a \equiv a' : A$. In a typing judgement $\Gamma \vdash X : \mathcal{U}_i$, we will refer to X as a *type*.

Substitution and structural weakening rules need not be assumed, they are admissible. Similarly, subject reduction, normalization, and canonicity of the presented type theory (without the Univalence Axiom, and with a suitable notion of reduction) are folklore [42] and need not concern us here.

3.2.1 Context Rules

$$\frac{}{\cdot\ \mathsf{ctx}}\ \textsf{ctx-EMP} \qquad \frac{x_1{:}A_1, \ldots, x_{n-1}{:}A_{n-1} \vdash A_n : \mathcal{U}_i}{(x_1{:}A_1, \ldots, x_n{:}A_n)\ \mathsf{ctx}}\ \textsf{ctx-EXT} \qquad \frac{(x_1{:}A_1, \ldots, x_n{:}A_n)\ \mathsf{ctx}}{x_1{:}A_1, \ldots, x_n{:}A_n \vdash x_i : A_i}\ \textsf{Vble}$$

3.2.2 Definitional Equality

Definitional equality is postulated to be an equivalence relation:

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv b : A}{\Gamma \vdash b \equiv a : A} \qquad \frac{\Gamma \vdash a \equiv b : A \qquad \Gamma \vdash b \equiv c : A}{\Gamma \vdash a \equiv c : A}$$

It may also be used for rewriting types:

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a : B} \qquad \frac{\Gamma \vdash a \equiv b : A \qquad \Gamma \vdash A \equiv B : \mathcal{U}_i}{\Gamma \vdash a \equiv b : B}$$

In the following subsections, we omit the repetitive rules establishing closure of definitional equality under application of type formers, constructors, and eliminators.

[3] while prohibiting so-called *indexed* inductive types, of which equality is the prime example

[4] Note that this is in a setting without higher inductive types, which artificially introduce higher path generators.

3.2.3 Type Universes

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}}\ \mathcal{U}\text{-INTRO} \qquad \frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash A : \mathcal{U}_{i+1}}\ \mathcal{U}\text{-CUMUL}$$

3.2.4 Dependent Functions

For this and the next subsection, fix judgements $\Gamma \vdash A : \mathcal{U}_i$ and $\Gamma, x{:}A \vdash B[x] : \mathcal{U}_i$. Formation and introduction are as follows:

$$\frac{\dots}{\Gamma \vdash \prod_{(x:A)} B[x] : \mathcal{U}_i}\ \Pi\text{-FORM} \qquad \frac{\dots \qquad \Gamma, x{:}A \vdash b[x] : B[x]}{\Gamma \vdash \lambda(x:A).\, b[x] : \prod_{(x:A)} B[x]}\ \Pi\text{-INTRO}$$

Elimination and associated computational behaviour are given by:

$$\frac{\dots \qquad \Gamma \vdash f : \prod_{(x:A)} B[x] \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B[a]}\ \Pi\text{-ELIM}$$

$$\frac{\dots \qquad \Gamma, x{:}A \vdash b : B \qquad \Gamma \vdash a : A}{\Gamma \vdash (\lambda(x:A).\, b[x])(a) \equiv b[a] : B[a]}\ \Pi\text{-COMP}$$

We also have definitional uniqueness of dependent functions:

$$\frac{\dots \qquad \Gamma \vdash f : \prod_{(x:A)} B[x]}{\Gamma \vdash f \equiv \lambda x.\, f(x) : \prod_{(x:A)} B[x]}\ \Pi\text{-UNIQ}$$

Notation Given $A : \mathcal{U}_i$ and $B : A \to \mathcal{U}_i$, we may write $\Pi_A B$ for $\prod_{(x:A)} B(x)$.

Given $A, B : \mathcal{U}_i$, let $A \to B$ be an abbreviation for $\prod_{(x:A)} B$.

3.2.5 Dependent Pairs

Formation and introduction are as follows:

$$\frac{\dots}{\Gamma \vdash \sum_{(x:A)} B[x] : \mathcal{U}_i}\ \Sigma\text{-FORM} \qquad \frac{\dots \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a]}{\Gamma \vdash (a, b) : \sum_{(x:A)} B[x]}\ \Sigma\text{-INTRO}$$

Fix an elimination type $\Gamma, z{:}\sum_{(x:A)} B[x] \vdash C[z] : \mathcal{U}_i$ with witness

$$\Gamma, x{:}A, y{:}B[x] \vdash c[x, y] : C[(x, y)]$$

and shorten $\mathsf{ind}_{\sum_{(x:A)} B[x]}(C, c, \cdot)$ by writing $\mathsf{ind}(\cdot)$. Elimination and associated computational behaviour are given by:

$$\frac{\dots \qquad \Gamma \vdash s : \sum_{(x:A)} B[x]}{\Gamma \vdash \mathsf{ind}(s) : C[s]}\ \Sigma\text{-ELIM} \qquad \frac{\dots \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : B[a]}{\Gamma \vdash \mathsf{ind}(a, b) \equiv c[a, b] : C[(a, b)]}\ \Sigma\text{-COMP}$$

Notation The projections

$$\mathsf{pr}_1 : \textstyle\sum_{(x:A)} B(x) \to A$$
$$\mathsf{pr}_2 : \textstyle\prod_{(s:\sum_{(x:A)} B(x))} \to B(\mathsf{pr}_1(s))$$

are defined in terms of the eliminator. They compute according to $\mathsf{pr}_1(a, b) \equiv a$ and $\mathsf{pr}_2(a, b) \equiv b$).

Given $A : \mathcal{U}_i$ and $B : A \to \mathcal{U}_i$, we may write $\Sigma_A B$ for $\sum_{(x:A)} B(x)$.

Given $A, B : \mathcal{U}_i$, let $A \times B$ be an abbreviation for $\sum_{(a:A)} B$.

3.2.6 Coproducts

For this subsection, fix judgements $\Gamma \vdash A : \mathcal{U}_i$ and $\Gamma \vdash B : \mathcal{U}_i$. Formation and introduction rules are as follows:

$$\frac{\dots}{\Gamma \vdash A + B : \mathcal{U}_i}\ +\text{-FORM} \qquad \frac{\dots \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathrm{inl}(a) : A + B}\ +\text{-INTRO}_1 \qquad \frac{\dots \qquad \Gamma \vdash b : B}{\Gamma \vdash \mathrm{inr}(b) : A + B}\ +\text{-INTRO}_2$$

Fix an elimination type $\Gamma, z{:}(A + B) \vdash C[z] : \mathcal{U}_i$ with witnesses

$$\Gamma, x{:}A \vdash c_l[x] : C[\mathrm{inl}(x)],$$
$$\Gamma, y{:}B \vdash c_r[y] : C[\mathrm{inr}(y)]$$

and shorten $\mathsf{ind}_{A+B}(C, c_l, c_r, \cdot)$ by writing $\mathsf{ind}(\cdot)$. The elimination rule is then given by

$$\frac{\dots \qquad \Gamma \vdash s : A + B}{\Gamma \vdash \mathsf{ind}(s) : C[s]}\ +\text{-ELIM}$$

and has computational behaviour as follows:

$$\frac{\dots \qquad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{ind}(\mathrm{inl}(a)) \equiv c_l[a] : C[\mathrm{inl}(a)]}\ +\text{-COMP}_1 \qquad \frac{\dots \qquad \Gamma \vdash b : B}{\Gamma \vdash \mathsf{ind}(\mathrm{inr}(b)) \equiv c_r[b] : C[\mathrm{inr}(b)]}\ +\text{-COMP}_2$$

3.2.7 Unit Type

Formation and introduction are as follows:

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbf{1} : \mathcal{U}_0}\ \mathbf{1}\text{-FORM} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \star : \mathbf{1}}\ \mathbf{1}\text{-INTRO}$$

Fix an elimination type $\Gamma, x{:}\mathbf{1} \vdash C[x] : \mathcal{U}_0$ with witness $\Gamma \vdash c : C[\star]$. Elimination and associated computational behaviour are given by:

$$\frac{\dots \qquad \Gamma \vdash s : \mathbf{1}}{\Gamma \vdash \mathsf{ind}_{\mathbf{1}}(C, c, s) : C[s]}\ \mathbf{1}\text{-ELIM} \qquad \frac{\dots}{\Gamma \vdash \mathsf{ind}_{\mathbf{1}}(C, c, \star) \equiv c : C[\star]}\ \mathbf{1}\text{-COMP}$$

Notation The type of Booleans $\mathbf{2} : \mathcal{U}_0$ is defined as $\mathbf{2} :\equiv \mathbf{1} + \mathbf{1}$. Its constructors are given by $0_{\mathbf{2}} :\equiv \mathrm{inl}(\star)$ and $1_{\mathbf{2}} :\equiv \mathrm{inr}(\star)$.

3.2.8 Empty Type

Formation and elimination are as follows:

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbf{0} : \mathcal{U}_i}\ \mathbf{0}\text{-FORM} \qquad \frac{\Gamma, x{:}\mathbf{0} \vdash C[x] : \mathcal{U}_i \qquad \Gamma \vdash s : \mathbf{0}}{\Gamma \vdash \mathsf{ind}_{\mathbf{0}}(C, s) : C[s]}\ \mathbf{0}\text{-ELIM}$$

Note that there are no introduction, hence also no computation rules.

Notation As usual, $\neg A$ will abbreviate $A \to \mathbf{0}$.

Natural Numbers

Formation and introduction are as follows:

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbb{N} : \mathcal{U}_i}\ \mathbb{N}\text{-FORM} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash 0 : \mathbb{N}}\ \mathbb{N}\text{-INTRO}_1 \qquad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{succ}(n) : \mathbb{N}}\ \mathbb{N}\text{-INTRO}_2$$

Fix an elimination type $\Gamma, x{:}\mathbb{N} \vdash C[x] : \mathcal{U}_i$ with witnesses

$$\Gamma \vdash c_0 : C[0],$$
$$\Gamma, x{:}\mathbb{N}, y{:}C[x] \vdash c_s[x, y] : C[\mathsf{succ}(x)]$$

and shorten $\mathsf{ind}_{\mathbb{N}}(C, c_0, c_s, \cdot)$ by writing $\mathsf{ind}(\cdot)$. Elimination and associated computational behaviour are given by:

$$\frac{\ldots \qquad \Gamma \vdash s : \mathbb{N}}{\Gamma \vdash \mathsf{ind}(s) : C[s]}\ \mathbb{N}\text{-ELIM} \qquad \frac{\ldots}{\Gamma \vdash \mathsf{ind}(0) \equiv c_0 : C[0]}\ \mathbb{N}\text{-COMP}_1$$

$$\frac{\ldots \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathsf{ind}(\mathsf{succ}(n)) \equiv c_s[n, \mathsf{ind}(n)] : C[\mathsf{succ}(n)]}\ \mathbb{N}\text{-COMP}_2$$

3.2.9 Identity Types

We present the Paulin-Mohring version of equality [47]. For this subsection, fix judgements $\Gamma \vdash A : \mathcal{U}_i$ and $\Gamma \vdash a : A$. Formation and introduction rules are as follows:

$$\frac{\ldots \qquad \Gamma \vdash b : A}{\Gamma \vdash a =_A b : \mathcal{U}_i}\ =\text{-FORM} \qquad \frac{\ldots}{\Gamma \vdash \mathsf{refl}_a : a =_A a}\ =\text{-INTRO}$$

Fix an elimination type $\Gamma, y{:}A, q{:}a =_A y \vdash C[y, q] : \mathcal{U}_i$ with witness

$$\Gamma, z{:}A \vdash c[z] : C[z, \mathsf{refl}_z]$$

and shorten $\mathsf{ind}_{=_{A,a}}(C, c, \cdot, \cdot)$ by writing $\mathsf{ind}(\cdot, \cdot)$. Elimination, also called *path induction*, and associated computational behaviour are given by:

$$\frac{\ldots \qquad \Gamma \vdash b : A \qquad \Gamma \vdash p : a =_A b}{\Gamma \vdash \mathsf{ind}(b, p) : C[b, p]}\ =\text{-ELIM} \qquad \frac{\ldots}{\Gamma \vdash \mathsf{ind}(a, \mathsf{refl}_a) \equiv c[a] : C[a, \mathsf{refl}_a]}\ =\text{-COMP}$$

Notation We will omit the typing subscript A in $a =_A b$ if it is easily derivable. As usual, $a \neq_A b$ will stand for $\neg(a =_A b)$, i.e. $a =_A b \to \mathbf{0}$.

3.3 Preliminaries

We will briefly review basic ideas, constructions, facts, and terminology from homotopy type theory. Detailed proofs and further discussion can be found in our main reference [56].

3.3.1 Types as ω-groupoids

Any type A can be regarded as a (weak) ω-groupoid where the 0-cells are the elements of A and the 1-cells are the equalities, or *paths*, between them: we have a neutral element $\mathsf{refl}_x : x = x$ for $x : A$ and can define composition $p \cdot q : x = z$ for $p : x = y$ and $q : y = z$ as well as reversal $p^{-1} : y = x$ for $p : x = y$. These operations fulfill the expected laws of neutrality, associativity,

and inversion. The witnesses of these laws, families of 2-paths, again cohere in a meaningful way with each other and the 1-operations. [5]

Functions between types preserve this ω-groupoidal structure, i.e. act as (weak) ω-functors. That is, a function $f : A \to B$ induces a function $\mathsf{ap}_f : x =_A y \to f(x) =_B f(y)$ for $x, y : A$ that is ω-functorial in f and its path space argument. Given $p : x =_A y$, we will usually write $f(p)$ instead of $\mathsf{ap}_f(p)$ as is usual in category theory.

3.3.2 Truncation Levels

A type A is *contractible*, or a (-2)-type, if it has an inhabitant a and any other element x has a designated path from a, i.e. $\prod_{(x:A)} a = x$. A type A is an $(n+1)$-type for $n \geq -2$ if the path space $x =_A y$ is an n-type for all $x, y : A$. [6]

The (-1)-types are commonly known as *propositions*. They can also be characterized as those types that have a choice for a path between any two elements. [7] Thus, an inhabited proposition is contractible. The 0-types are known as *sets*. In the context of verifying A is an n-type,

- if $n \geq -1$, we can assume A is inhabited, [8]
- if $n \geq 0$, we only need to show $x =_A x$ is an $(n-1)$-type for $x : A$. [9]

The statement of A being an n-type is monotonic in n, i.e. for $n_1 \leq n_2$, if A is an n_1-type, then also an n_2-type. [10] Importantly, it is propositional itself. [11]

3.3.3 Equivalences

The *fiber* of a function $f : A \to B$ over a point $y : B$ is the type $\sum_{(x:A)} f(x) = y$. If all fibers of f are contractible, then f is called an *equivalence*. This predicate $\mathsf{isequiv}(f)$ is propositional. An equivalent definition is in terms of *coherent isomorphism*, consisting of a pointwise inverse $g : B \to A$ with $\epsilon : \prod_{(x:A)} g(f(x)) = x$ and $\eta : \prod_{(y:B)} f(g(y)) = y$ that are coherent by $\prod_{(x:A)} f(\epsilon(x)) = \eta(f(x))$. Any isomorphism can be made coherent by changing either ϵ or η.

We write $A \simeq B :\equiv \sum_{(f:A\to B)} \mathsf{isequiv}(f)$ for the type of equivalences between A and B. Equivalences, like equalities, are closed under identity, composition, and inverse. Given $e : A \simeq B$, we will also use e to designate the underlying function from A to B. Equality of equivalences is equivalent to equality of the underlying functions. [12] An equivalence $f : A \to B$ induces equivalences ap_f on the path spaces of A and B.

All type formers preserve equivalence. For example, given $A_j : \mathcal{U}_i$ and $B_j : A_j \to \mathcal{U}_i$ for $j = 1, 2$ with an equivalence $u : A_1 \simeq A_2$ and equivalences $B_1(a_1) \simeq B_2(u(a_1))$ for $a_1 : A$, then $\Pi_{A_1} B_1 \simeq \Pi_{A_2} B_2$ and $\Sigma_{A_1} B_1 \simeq \Sigma_{A_2} B_2$. These type-level compositional equivalences are best understood in terms of coherent isomorphisms.

The notion of being an n-type is invariant under equivalence.

---

[5] Even though we can state these higher coherence laws for any bounded level of dimension, i.e. define an n-groupoid for any n, actually giving a definition of an ω-groupoid on types inside homotopy type theory seems to be a profound open problem.

[6] By convention, the notion of n-type ranges over $n \geq -2$. We regard this as a simple notional convenience with a canonical isomorphism $\mathbb{N}_{-2} \simeq \mathbb{N}$.

[7] This instructive exercise forms the core argument of Hedberg's Theorem [24].

[8] Just note that we are given elements $x, y : A$ for which we must show something (that $x = y$ is an $(n-1)$-type).

[9] Given $x, y : A$, do path induction on the element of $x =_A y$ coming from the previous list item.

[10] This follows from contractible types being propositional with the above characterization of propositions.

[11] A path between two witnesses of contractibility of A is given by contractibility of A and the structural equality of (=-Σ) and (=-Π) below. The general notion of being an n-type inherits propositionality from the case $n \equiv -2$ as explained under (=-Π).

[12] This follows from $\mathsf{isequiv}(\cdot)$ being propositional and the structural equality (=-Σ) on Σ-types below.

3.3.4 Univalence

Given a family $P : A \to U_i$ and a path $p : a_1 =_A a_2$, there is a *transport* or *substitution* operation $\mathsf{transport}^P(p, \cdot) :\equiv p_* : P(a_1) \to P(a_2)$ reducing to the identity for a reflexivity argument. We will use the shorter second form if the type family P is clear from the context. It is (pointwise) ω-functorial in p, and in particular ${p^{-1}}_*(\cdot)$ is a pointwise inverse to $p_*(\cdot)$. It thus induces an equivalence $P(a_1) \simeq P(a_2)$.

For $A = U_i$ and $P(X) = X$, the above transportation equivalence yields a function $\mathsf{idtoeqv} : X =_{U_i} Y \to X \simeq Y$ coercing equalities on types to equivalences. The *Univalence Axiom* now states that this function is an equivalence. [13] When reasoning about type equivalence, we will henceforth use equality notation.

3.3.5 Structural Equality via Univalence

The Univalence Axiom enables us to identify equality on a given type former with the correct structural notion of equality of the type former in question. This core tool will often be used as primitive steps in our manipulations of types.

It is important to keep in mind not only the type equalities presented in this subsection, but also the action of the underlying isomorphisms. They mediate ω-functorially between the natural ω-groupoidal structure on a given type and a synthetic one built out of the type structure. For instance, a reflexivity proof will correspond to a family of reflexivity proofs via (=-Π). This will be made use of when these equations are cited later on.

Universes $\mathcal{U}_i$ Given $A, B : \mathcal{U}_i$, univalence makes equality $A =_{\mathcal{U}_i} B$ on a universe equivalent to equivalence $A \simeq B$ via $\mathsf{idtoeqv}$:

$$(A =_{\mathcal{U}_i} B) \;=\; (A \simeq B). \tag{=-$\mathcal{U}$}$$

Writing $\mathcal{U}_i^n :\equiv \sum_{(X:\mathcal{U}_i)} \mathsf{is}\text{-}n\text{-}\mathsf{type}(X)$ for the subuniverse of n-types, equality on $\mathcal{U}_i^n$ coincides with equality on its embedding in $\mathcal{U}_i$. If one of A or B is an n-type, then so is $A \simeq B$, i.e. $A = B$. [14] It follows that $\mathcal{U}_i^n$ is an $(n+1)$-type. The central consequence of this work is that $\mathcal{U}_n^n$ is *strictly* an $(n+1)$-type, i.e. not an n-type.

Unit Type 1 Equality on the unit type is equivalent to the unit type itself. This situation can be viewed as a degenerate instance of the general fact that equality on k-tuples is equivalent to a k-tuple of equalities. Given $s, t : \mathbf{1}$, then

$$(s =_{\mathbf{1}} t) \;=\; \mathbf{1}. \tag{=-$\mathbf{1}$}$$

Note that the unit type is contractible. Any contractible type is equivalent, and by univalence hence equal, to the unit type.

Σ-Types The obvious structural notion of equality on $A \times B$ is a pair of equalities, one on A and the other on B. Given $(a_1, b_1), (a_2, b_2) : A \times B$, we indeed have an equivalence between $(a_1, b_1) = (a_2, b_2)$ and $(a_1 = a_2) \times (b_1 = b_2)$.

In case of a family $B : A \to U_i$ and with $(a_1, b_1), (a_2, b_2) : \Sigma_A B$, we cannot directly compare b_1 and b_2 as they live in different types. However, given a path of type $a_1 = a_2$, transportation induces an equivalence $B(a_1) \simeq B(a_2)$ enabling us to state equality on the second components.

[13] The Univalence Axiom is the only axiom assumed in this development (in addition to the type theory described in the previous section).

[14] This assertion and the one before it follow from the paragraphs below on Π- and Σ-types and the unit type $\mathbf{1}$: an equality on dependent pairs with propositional second component type is equivalent to equality on the first component.

Equality on dependent pairs on $\Sigma_A B$ is then equivalent to the structural notion of dependent pairs of equality on A and $B(\cdot)$:

$$((a_1, b_1) =_{\Sigma_A B} (a_2, b_2)) \;=\; \textstyle\sum_{(p:a_1 =_A a_2)} (p_*(b_1) =_{B(a_2)} b_2). \tag{=-Σ}$$

The reverse direction of this equivalence is denoted $\mathsf{pair}^=$, defined by path induction on the first and second component.

Again, induction on n shows that $\Sigma_A B$ is an n-type if A and $B(a)$, for all $a : A$, are n-types.

Π-Types Fix $f_1, f_2 : \Pi_A B$. Equality of functions certainly implies pointwise equality. For instance, we might define $\mathsf{happly} : f = g \to \Pi_{a:A}(f_1(a) = f_2(a))$ via $\mathsf{happly}(p)(a) = \mathsf{ap}_{\lambda f.\, f(a)}(p)$. Univalence now implies that happly is an equivalence, an assertion commonly known as *function extensionality*. This makes equality on $\Pi_A B$ equivalent to the structural notion of pointwise equality on $B(a)$ for $a : A$:

$$(f_1 =_{\Pi_A B} f_2) \;=\; \textstyle\prod_{(a:A)} (f_1(a) =_{B(a)} f_2(a)). \tag{=-Π}$$

By induction on n, we see that $\Pi_A B$ is an n-type if $B(a)$ is an n-type for all $a : A$.

Naturals $\mathbb{N}$ (Parametric) inductive types are modelled by so-called (parametric) W-types that compute the least-fixed point of an (indexed) functor on types. In general, equality on (parametric) W-types can be shown equivalent to a parametric W-type of equalities. [15]

However, since our type theory does not include facilities for any inductive type besides $\mathbb{N}$, we will not delve into this direction. Still, along the lines of the reasoning alluded to above it may be seen that $\mathbb{N}$ is a set. [16]

3.3.6 Computing Transportation

If we know the shape of $P : A \to \mathcal{U}_i$, we expect to be able to decompose a transport $p_*(\cdot) : P(a_1) \to P(a_2)$ on P where $p : a_1 =_A a_2$ into transports on its components. The proof of the below reduction principles consists mainly of an application of path induction on p.

Constant Types Let $P(x) :\equiv W$ with $W : \mathcal{U}_i$, i.e. let P have no dependency on x. The transportation action is then trivial: given $w : W$, we have

$$\mathsf{transport}^P(p, w) = w. \tag{$*$-const}$$

Σ-Types Let first $P(x) :\equiv U(x) \times V(x)$ with $U, V : A \to U_i$. Transportation on a product is componentwise transportation: given $u : U(a_1)$ and $v : V(a_1)$, we have

$$\mathsf{transport}^P(p, (u, v)) = (\mathsf{transport}^U(p, u), \mathsf{transport}^V(p, v)). \tag{$*$-$\times$}$$

Let us now consider the dependent case $P(x) :\equiv \sum_{(u:U(x))} V(x, u)$ with $U : A \to U_i$ and $V : \Sigma_A U \to U_i$. Given $u : U(a_1)$ and $v : V(a_1, u)$, then

$$\mathsf{transport}^P(p, (u, v)) = (\mathsf{transport}^U(p, u), \mathsf{transport}^V(\mathsf{pair}^=(p, \mathsf{refl}_{p_*(u)}), v)). \tag{$*$-Σ}$$

[15] The same is true for (parametric) M-types modelling (parametric) coinductive types.

[16] Hedberg's Theorem [24], occasionally cited in regards to this assertion, does not have much to do with this at all and is more a confused instance of backwards reasoning: showing that $\mathbb{N}$ has decidable equality is in fact just a selective reading of the above recursive presentation of equality on $\mathbb{N}$ in structural form, which is already a proposition by trivial induction.

Π-Types Let first $P(x) :\equiv U(x) \to V(x)$ with $U, V : A \to U_i$. Transportation on a function space is contravariant transportation on the domain and covariant transportation on the codomain: given $f : U(a_1) \to V(a_1)$, then

$$\mathsf{transport}^P(p, f) = \mathsf{transport}^V(p, \cdot) \circ f \circ \mathsf{transport}^U(p^{-1}, \cdot). \qquad (*\text{-}\to)$$

Again, let us consider the dependent case $P(x) :\equiv \prod_{(u:U(x))} V(x, u)$ with $U : A \to U_i$ and $V : \Sigma_A U \to U_i$. Given $f : \prod_{(u:U(x))} V(a_1, u)$, then

$$\mathsf{transport}^P(p, f) = \lambda u.\, \mathsf{transport}^V(\mathsf{pair}^{=}(p^{-1}, \mathsf{refl}_{{p^{-1}}_*(u)})^{-1}, f(\mathsf{transport}^U(p^{-1}, u))). \qquad (*\text{-}\Pi)$$

=-Types Let $P(x) :\equiv u(x) =_B v(x)$ with $u, v : A \to B$. [17] This case is related in spirit to the non-dependent function space case: given $q : u(a_1) = v(a_1)$, then

$$\mathsf{transport}^P(p, q) = u(p)^{-1} \cdot q \cdot v(p). \qquad (*\text{-}=)$$

3.4 Some Useful Type Isomorphisms

We will review straightforward generalizations of structural type isomorphisms in cartesian-closed categories to dependent types. The witness maps for the below equivalences are canonical. We will omit them.

The unit type is a left unit for dependent pairs and functions: for any family $B : \mathbf{1} \to \mathcal{U}_i$, we have

$$\textstyle\sum_{(a:\mathbf{1})} B(a) \;\simeq\; B(\star), \qquad (\Sigma_1\text{-}\mathbf{1})$$
$$\textstyle\prod_{(a:\mathbf{1})} B(a) \;\simeq\; B(\star). \qquad (\Pi_1\text{-}\mathbf{1})$$

The unit type is a right unit for dependent pairs and a right annihilator for dependent functions: for any type $A : \mathcal{U}_i$, we have

$$\textstyle\sum_{(a:A)} \mathbf{1} \;\simeq\; A, \qquad (\Sigma_2\text{-}\mathbf{1})$$
$$\textstyle\prod_{(a:A)} \mathbf{1} \;\simeq\; \mathbf{1}. \qquad (\Pi_2\text{-}\mathbf{1})$$

For the next three laws, fix a type $A : \mathcal{U}_i$, a family $B : A \to \mathcal{U}_i$ and a another family $C : \Sigma_A B \to \mathcal{U}_i$. Dependent pairs are associative:

$$\textstyle\Sigma_{\Sigma_A B} C \;\simeq\; \sum_{(a:A)} \sum_{(b:B(a))} C(a, b). \qquad (\Sigma\text{-}\Sigma)$$

Currying is an equivalence:

$$\textstyle\Pi_{\Sigma_A B} C \;\simeq\; \prod_{(a:A)} \prod_{(b:B(a))} C(a, b). \qquad (\Pi_1\text{-}\Sigma)$$

Dependent pairs distribute under dependent functions: [18]

$$\textstyle\prod_{(a:A)} \sum_{(b:B(a))} C(a, b) \;\simeq\; \sum_{(f:\Pi_A B)} \prod_{(a:A)} C(a, f(a)). \qquad (\Pi_2\text{-}\Sigma)$$

[17] There is a more complicated, though equally natural, version where the underlying type B may depend itself on a. However, we do not need to go that far in this development.

[18] also known controversially as the *type-theoretic axiom of choice*

3.5 Universe $\mathcal{U}_0$ is not a set

The result presented in this section has been well known since the beginnings of homotopy type theory. It is a direct consequence of the fact that univalence allows the construction of a non-trivial equality $\mathbf{2} =_{\mathcal{U}_0} \mathbf{2}$.

The Booleans $\mathbf{2}$ are the simplest example of a type with a non-trivial automorphism, here given by negation $\mathsf{swap} : \mathbf{2} \simeq \mathbf{2}$. Indeed, $\mathsf{swap} = \mathsf{id}$ would imply $1_{\mathbf{2}} \equiv \mathsf{swap}(0_{\mathbf{2}}) = \mathsf{id}(0_{\mathbf{2}}) \equiv 0_{\mathbf{2}}$. Defining a family $P : \mathbf{2} \to \mathcal{U}_0$ by $P(0_{\mathbf{2}}) :\equiv \mathbf{0}$ and $P(1_{\mathbf{2}}) :\equiv \mathbf{1}$, we could use our proof $p : 1_{\mathbf{2}} = 0_{\mathbf{2}}$ to transport $\star : \mathbf{1}$ to $p_*(\star) : \mathbf{0}$, showing $\mathsf{swap} \neq \mathsf{id}$. By definition of (-1)-types, this implies $\mathbf{2} \simeq \mathbf{2}$ is not propositional. Univalence then shows the same holds for $\mathbf{2} =_{\mathcal{U}_0} \mathbf{2}$. Hence, $\mathcal{U}_0$ cannot be a set.

The obvious generalization of $\mathcal{U}_n$ not being an n-type was long thought to be similarly trivial. On detailed examination, this turned out to be a naive assumption. While several people were able to give different constructions for the case $n :\equiv 1$, the general case remained unsolved and was indeed featured on the open problem list of the Special Year on Univalent Foundations. In the remainder of this chapter, we will present our approach to this problem.

3.6 Pointed types

A *pointed type* is simply a type A with a chosen inhabitant $a : A$. We write $\mathcal{U}_i^\bullet :\equiv \sum_{(A:\mathcal{U}_i)} A$ for the so-called universe of pointed types. A pointed function $(A, a) \to^\bullet (B, b)$ between pointed types is a function $f : A \to B$ together with a proof $f(a) = b$ that the basepoint is preserved. A pointed equivalence $(A, a) \simeq^\bullet (B, b)$ is a pointed function where the function part is also an equivalence. A straightforward[19] calculation

$$\begin{array}{lll} [\text{by } (=\text{-}\Sigma)] \quad (A,a) =_{\mathcal{U}_i^\bullet} (B,b) & = & \sum_{(p:A=B)} p_*(a) = b \\ [\text{by } (=\text{-}\mathcal{U})] & = & \sum_{(e:A\simeq B)} e(a) = b \qquad (=\text{-}\mathcal{U}^\bullet) \\ & \equiv & (A,a) \simeq^\bullet (B,b) \end{array}$$

reveals that pointed equivalence is indeed the correct notion of equality on $\mathcal{U}_i^\bullet$, giving us a version of univalence for pointed types.

Pointed types are significant in algebraic topology as they conveniently provide a basepoint for looping paths (with respect to which, for example, homotopy groups can be computed). This motivates the introduction of the *loop space* operator

$$\begin{array}{rcl} \Omega & : & \mathcal{U}_i^\bullet \to \mathcal{U}_i^\bullet, \\ \Omega(A,a) & :\equiv & (a = a, \mathsf{refl}_a). \end{array}$$

Note that, both endpoints of the path being identical, the identity path naturally serves as a new basepoint.

Given a pointed type $(A, a) : \mathcal{U}_i^\bullet$, a *pointed predicate* over (A, a) consists of a family $B : A \to \mathcal{U}_i$ with a chosen point $b : B(a)$. We write $\mathsf{Pred}_i^\bullet(A, a) :\equiv \sum_{(B:A\to\mathcal{U}_i)} B(a)$ for the type of pointed predicates over (A, a). Note that pointed types can be recovered as pointed predicates over the pointed unit type $\mathbf{1}^\bullet$. We can generalize the loop space operator to pointed predicates:

$$\begin{array}{rcl} \tilde{\Omega}_{(A,a)} & : & \mathsf{Pred}_i^\bullet(A,a) \to \mathsf{Pred}_i^\bullet(\Omega(A,a)), \\ \tilde{\Omega}_{(A,a)}(B,b) & :\equiv & (\lambda(p : a =_A a).\, p_*(b) =_{B(a)} b, \mathsf{refl}_b). \end{array}$$

The meaning of the transport will become apparent when defining pointed dependent pairs. Again, under the equalities $\mathsf{Pred}_i^\bullet(\mathbf{1}^\bullet) = \mathcal{U}_i^\bullet$ and $\Omega(\mathbf{1}^\bullet) = \mathbf{1}^\bullet$, we have $\tilde{\Omega}_{\mathbf{1}^\bullet} = \Omega$.

For readability, we will sometimes use a pointed type or predicate to stand in for its underlying type or predicate, respectively, with the obvious coercion left implicit. For example, we

[19] in the sense of there only being one sensible way to proceed

might call a pointed type an n-type when actually talking about its underlying type. Note that for pointed types, being contractible and propositional is equivalent.

Let us try to generalize some of our type formers to pointed types.

Pointed unit type $\mathbf{1}^\bullet$ The unit type $\mathbf{1}$ comes with a canonical inhabitant $\star$. We thus define $\mathbf{1}^\bullet :\equiv (\mathbf{1}, \star)$. Trivially,

$$\Omega(\mathbf{1}^\bullet) = \mathbf{1}^\bullet. \tag{Ω-$\mathbf{1}^\bullet$}$$

$\Pi^\bullet$-types Given $A : \mathcal{U}_i$, a family $\Phi : A \to \mathcal{U}_i^\bullet$ of pointed types may be presented as $\Phi = \langle B, b\rangle$ with a type family $B : A \to \mathcal{U}_i$ and a section $b : \Pi_A B$ of basepoints. We may thus define a type former for pointed dependent products:

$$\begin{aligned} \Pi^\bullet &: \prod_{(A:\mathcal{U}_i)} (A \to \mathcal{U}_i^\bullet) \to \mathcal{U}_i^\bullet \\ \Pi^\bullet_A \Phi &:\equiv (\Pi_A B, b). \end{aligned}$$

We will also write $\Pi^\bullet_{a:A}\Phi(a)$ for $\Pi^\bullet_A \Phi$.

In line with the above paradigm of equality on some type structure as a same-shape structure of equalities, the loop space operator interacts nicely with our new type former $\Pi^\bullet$:

$$\begin{aligned} \Omega(\Pi^\bullet_A \Phi) &\equiv \Omega(\Pi_A B, b) \\ &\equiv (b = b, \mathsf{refl}_{\mathsf{pr}_2 \circ \Phi}) \\ [\text{by } (=\text{-}\Pi)] \qquad &= (\textstyle\prod_{(a:A)} b(a) = b(a), \lambda(a:A).\, \mathsf{refl}_{b(a)}) \\ &\equiv \Pi^\bullet_{a:A}(b(a) = b(a), \mathsf{refl}_{b(a)}) \\ &\equiv \Pi^\bullet_{a:A}(\Omega(\Phi(a))). \end{aligned} \tag{Ω-$\Pi^\bullet$}$$

$\Sigma^\bullet$-types We may define a type former for pointed dependent pairs

$$\begin{aligned} \Sigma^\bullet &: \prod_{((A,a):\mathcal{U}_i^\bullet)} \mathsf{Pred}_i^\bullet(A, a) \to \mathcal{U}_i^\bullet, \\ \Sigma^\bullet_{(A,a)}(B, b) &:\equiv (\Sigma_A B, (a, b)). \end{aligned}$$

Let us look at the interaction of the loop space operator with the type former $\Sigma^\bullet$:

$$\begin{aligned} \Omega(\Sigma^\bullet_{(A,a)}(B, b)) &\equiv \Omega(\Sigma_A B, (a, b)) \\ &\equiv ((a, b) = (a, b), \mathsf{refl}_{(a,b)}) \\ [\text{by } (=\text{-}\Sigma)] \qquad &= (\textstyle\sum_{(p:a=a)} p_*(b) = b, (\mathsf{refl}_a, \mathsf{refl}_b)) \\ &\equiv \Sigma^\bullet_{(a=a,\mathsf{refl}_a)}(\lambda(p : a = a).\, p_*(b) = b, \mathsf{refl}_b) \\ &\equiv \Sigma^\bullet_{\Omega(A,a)}(\tilde{\Omega}_{(A,a)}(B, b)). \end{aligned} \tag{Ω-$\Sigma^\bullet$}$$

Here, we had to make use of our loop space indexed operator $\tilde{\Omega}$ on pointed predicates.

Universe $\mathcal{U}$ with a point Fix $A : \mathcal{U}_i$. Let us calculate

$$\begin{aligned} \Omega(\mathcal{U}_i, A) &\equiv (A = A, \mathsf{refl}_A) \\ [\text{by } (=\text{-}\mathcal{U})] \qquad &= (A \simeq A, \mathsf{id}_A) \\ &\equiv (\textstyle\sum_{(f:A\to A)} \mathsf{isequiv}(f), (\mathsf{id}_A, \mathsf{idisequiv}_A)) \\ &\equiv \Sigma^\bullet_{(A\to A, \mathsf{id}_A)}(\mathsf{isequiv}, \mathsf{idisequiv}_A) \\ &\equiv \Sigma^\bullet_{\Pi^\bullet_{a:A}(A,a)}(\mathsf{isequiv}, \mathsf{idisequiv}_A). \end{aligned} \tag{Ω-$\mathcal{U}^\bullet$}$$

Since the pointed predicate $(\mathsf{isequiv}, \mathsf{idisequiv}_A)$ is a propositional family, it will vanish under the next loop space iteration:

$$\begin{array}{lrcll} \text{[by } (\Omega\text{-}\Sigma^\bullet)\text{]} & \Omega^2(\mathcal{U}_i, A) & = & \Sigma^\bullet_{\Omega(\Pi^\bullet_{a:A}(A,a))}\tilde{\Omega}(\mathsf{isequiv}, \mathsf{idisequiv}_A) & \\ \text{[by contractibility and } (\Sigma_2\text{-}\mathbf{1})\text{]} & & = & \Omega(\Pi^\bullet_{a:A}(A,a)) & (\Omega^2\text{-}\mathcal{U}^\bullet) \\ \text{[by } (\Omega\text{-}\Pi^\bullet)\text{]} & & = & \Pi^\bullet_{a:A}\Omega(A,a). & \end{array}$$

For intuition, it might be helpful at this point to recall that $\Omega^2(\mathcal{U}_i, A) \equiv (\mathsf{refl}_A =_{A=_{\mathcal{U}_i} A} \mathsf{refl}_A, \mathsf{refl}_{\mathsf{refl}_A})$. Particularly at higher iterations, we prefer the abstract loop space notation over the explicit expansion, which becomes hard to operate on quite early. As a further benefit, loop spaces as abstract operations on pointed types force us to always be rigorous with respect to how our type transformations affect the chosen inhabitant.

We call a type A *transitive* [20] if any $x, y : A$ have an equivalence $e : A \simeq A$ such that $e(x) = y$. By $(=\text{-}\mathcal{U}^\bullet)$, a pointed type (A, a) is transitive if and only $(A, a) = (A, x)$. [21] A transitive pointed type (A, a) being an n-type with $n \geq 0$ is equivalent to $\Omega(A, x)$ being an $(n-1)$-type for all $x : A$ by our preliminaries on truncation levels, and by transitivity simply equivalent to $\Omega(A, a)$ being an $(n-1)$-type.

An important instance, the loop space $\Omega(A, a)$ of a pointed type $(A, a) : \mathcal{U}_i^\bullet$ is transitive: for any $x : A$ and $p : a = x$, we have

$$(a = a, \mathsf{refl}_a) =_{\mathcal{U}_i^\bullet} (a = x, p)$$

by path induction on x and p. Specializing x to a, we get the desired conclusion. [22] Using this to iterate the previous argument, a transitive pointed type (A, a) is an n-type with $n \geq -1$ exactly if $\Omega^{n+1}(A, a)$ is a proposition, i.e. contractible. This implies

Lemma 3.1. *Given $n \geq 0$, a type A is an n-type exactly if $\Omega^{n+1}(A, a)$ is contractible for all $a : A$.*

Proof. Simply recall that A being an n-type is equivalent to $\Omega(A, a)$ being an $(n-1)$-type for all $a : A$ and use the remark preceeding the lemma. □

3.7 Universe $\mathcal{U}_1$ is not a 1-type

Now we have the necessary machinery to elegantly handle the next step of our problem. By Lemma 3.1, constructively disproving $\mathcal{U}_1$ a 1-type means finding $A : \mathcal{U}_1$ such that $\Omega^2(\mathcal{U}_1, A)$ is not contractible. Applying $(\Omega^2\text{-}\mathcal{U}^\bullet)$, it suffices to find a non-trivial element of $\Pi^\bullet_{a:A}\Omega(A, a)$.

Having already found that $\mathcal{U}_0$ exhibits non-trivial higher homotopies, it is tempting to use this base case by inserting $A :\equiv \mathcal{U}_0$. Under univalence $(=\text{-}\mathcal{U})$, we know $\prod_{(X:\mathcal{U}_0)} X = X$ corresponds to $\prod_{(X:\mathcal{U}_0)} X \simeq X$. On a closed universe, with an induction principle on the form of types available, we could endevaour to find such a non-trivial family of auto-equivalences. In our situation, with an open universe, it seems hopeless. In fact, with a hypothetical form of parametricity in the univalent setting, [23] we could metatheoretically show that the only derivable example of such a family of auto-equivalences is the identities.

[20] We came up with this terminology during our work on the article [36] this chapter is based on, in reference to transitive actions. Subsequently, other people have chosen to use the term *homogeneous*.

[21] Axiom K allows the formulation of McBride's *heterogeneous equality* framework [43] where $(A, a) =_{\mathcal{U}_i^\bullet} (A, x)$ always implies $a = x$. In that setting, the above statement would lose any meaning, serving to illustrate the difference between UIP and univalence.

[22] This is an illuminating example of a common technique in homotopy type theory: generalizing path endpoints by introducing separate point quantifiers where possible, we can often create surprising opportunities for path induction.

[23] Terms in type theory have certain naturality properties. For example, polymorphic operations, i.e. operations defined over a parameter that is an element of a universe, cannot base their behaviour on an inspection of intensional properties of their argument like its structure. Parametricity is the name given to any metatheorical theorem that makes these properties explicit. In a sense, it is the metatheorical analysis of the term model, i.e.

We are hence lead to a more refined approach. Consider the type

$$A :\equiv \Sigma_{\mathcal{U}_0}\Omega(\mathcal{U}_0,\cdot) \equiv \textstyle\sum_{(X:\mathcal{U}_0)}\Omega(\mathcal{U}_0,X).$$

Intuitively, A is the type of images of 1-spheres, or *circles*, in $\mathcal{U}_0$, with such an image given by a basepoint and a loop. Elements of $\Pi^\bullet_{a:A}\Omega(A,a)$ correspond to uniform transformations of such an image into itself. Pictorially speaking, a particular non-trivial such transformation is given by rotating the given image of the circle along itself, and in the topological model, this is exactly what the following construction corresponds to. For intuition, let us do the following calculation for the underlying type only, disregarding pointedness.

$$\begin{aligned}
\Omega^2(\mathcal{U}_1,A) &= \textstyle\prod_{((X,p):A)}\Omega(A,(X,p)) \\
&= \textstyle\prod_{((X,p):A)}(X,p) = (X,p) \\
\text{[by } (\Pi_1\text{-}\Sigma) \text{ and } (=\text{-}\Sigma)] \quad &= \textstyle\prod_{(X:\mathcal{U}_0)}\prod_{(p:X=X)}\sum_{(q:X=X)}\mathsf{transport}^{\lambda X.\, X=X}(p,q) = p \\
\text{[by } (*\text{-}=)] \quad &= \textstyle\prod_{(X:\mathcal{U}_0)}\prod_{(p:X=X)}\sum_{(q:X=X)} q^{-1}\cdot p\cdot q = p.
\end{aligned}$$

We are thus asked to find, for any type $X:\mathcal{U}_0$ and loop $p : X = X$, a loop q of the same type *commuting* with p. Even though, in contrast to higher homotopy groups, the fundamental group is not in general commutative, such examples are readily found in the powers of p, for example

$$\begin{aligned}
f_0 &:\equiv \lambda X.\,\lambda p.\,(\mathsf{refl}_X, w_0[X,p]), \\
f_1 &:\equiv \lambda X.\,\lambda p.\,(p, w_1[X,p]),
\end{aligned}$$

where $w_0[X,p] : {\mathsf{refl}_X}^{-1}\cdot p\cdot \mathsf{refl}_X = p$ and $w_1[X,p] : p^{-1}\cdot p\cdot p = p$ are left implicit.

Note that f_0 corresponds to the basepoint of $\Omega^2(\mathcal{U}_1,A)$ under the above series of transformations. This separate transformation traversal can be avoided by carrying out the above steps within our framework for loop spaces of pointed typed:

$$\begin{aligned}
\Omega^2(\mathcal{U}_1,A) &= \Pi^\bullet_{(X,p):A}\Omega(A,(X,p)) \\
&\equiv \Pi^\bullet_{(X,p):A}\Omega(\Sigma^\bullet_{(\mathcal{U}_0,X)}(\Omega(\mathcal{U}_0,\cdot),p)) \\
\text{[by } (\Omega\text{-}\Sigma^\bullet)] \quad &= \Pi^\bullet_{(X,p):A}\Sigma^\bullet_{\Omega(\mathcal{U}_0,X)}\tilde{\Omega}(\Omega(\mathcal{U}_0,\cdot),p) \\
\text{[by } (\Pi_2\text{-}\Sigma)] \quad &= \Pi^\bullet_{X:\mathcal{U}_0}\Pi^\bullet_{p:\Omega(\mathcal{U}_0,X)}\Sigma^\bullet_{\Omega(\mathcal{U}_0,X)}\tilde{\Omega}(\Omega(\mathcal{U}_0,\cdot),p)
\end{aligned}$$

where

$$\begin{aligned}
\tilde{\Omega}_{\Omega(\mathcal{U}_0,X)}(\Omega(\mathcal{U}_0,\cdot),p) &\equiv (\lambda(q:X=X).\,\mathsf{transport}^{\lambda X.\, X=X}(p,q) = p, \mathsf{refl}_p) \\
\text{[by } (*\text{-}=)] \quad &= (\lambda q.\, q^{-1}\cdot p\cdot q = p, w_0[X,p]).
\end{aligned}$$

While at first glance this mainly seems to introduce more obscurity, in particular the loop space indexed operator $\tilde{\Omega}$ on pointed predicates, it actually makes for a more modular and concise formal development, avoiding redundant clutter. This will become ever more obvious in the case $n > 1$.

Continuing where we left off, assuming $f_0 = f_1$, we have $\sum_{(r:\mathsf{refl}_X=p)} r_*(w_0[X,p]) = w_1[X,p]$ for all X and p after applications of $(=\text{-}\Pi)$ and $(=\text{-}\Sigma)$. In particular, we have $\mathsf{refl}_X = p$, showing $X = X$ contractible for all $X:\mathcal{U}_0$. This makes $\mathcal{U}_0$ a set, which we already know to be false. Hence, f_1 must be a non-trivial element, making $\Omega^2(\mathcal{U}_1,A)$ non-contractible, and thus $\mathcal{U}_1$ a non-1-type.

---

the syntactical form of closed inhabitants of a given type.

For a given internalization of naturality properties, parametricity is also the name given to an axiom stating internally that all operations, not only those syntactibly definable, satisfy these properties. To a certain extent, univalence is such an axiom. However, it deals only with naturality of isomorphisms, not all functions, between types and structures. The study of how to extend identity types and univalence to cover this directedness is part of the subject of directed type theory.

3.8 Universe $\mathcal{U}_n$ is not an n-type

3.8.1 A failed approach

Inspired by our success in the case of $\mathcal{U}_1$, we will now try to attack the general case $n \geq 1$. Again, by Lemma 3.1, constructively disproving $\mathcal{U}_n$ an n-type means finding $A : \mathcal{U}_n$ such that $\Omega^{n+1}(\mathcal{U}_n, A)$ is not contractible, and with (Ω^2-$\mathcal{U}^\bullet$) it suffices to find a non-trivial element of $\Pi^\bullet_{a:A}\Omega^n(A, a)$.

Generalizing from the previous section, we now make the educated guess of

$$A :\equiv \Sigma_{\mathcal{U}_{n-1}}\Omega^n(\mathcal{U}_{n-1}, \cdot) \equiv \textstyle\sum_{(X:\mathcal{U}_{n-1})}\Omega^n(\mathcal{U}_{n-1}, X).$$

Intuitively, A is the type of images of n-spheres in $\mathcal{U}_n$, consisting of a basepoint and a higher loop. Describing what an element of $\Pi^\bullet_{a:A}\Omega^n(A, a)$ corresponds to in this picture is becoming rather difficult already for $n :\equiv 2$, but we hope to have included enough homotopical complexity on level n such that our calculation in pointed types will result in something useful.

The first steps in our calculation mirror the ones of the previous section:

$$\begin{aligned}
\Omega^{n+1}(\mathcal{U}_n, A) &= \Pi^\bullet_{(X,p):A}\Omega^n(A, (X, p)) \\
&\equiv \Pi^\bullet_{(X,p):A}\Omega^n(\Sigma^\bullet_{(\mathcal{U}_{n-1},X)}(\Omega^n(\mathcal{U}_{n-1}, \cdot), p)) \\
\text{[by } (\Omega\text{-}\Sigma^\bullet)] \quad &= \Pi^\bullet_{(X,p):A}\Sigma^\bullet_{\Omega^n(\mathcal{U}_{n-1},X)}\tilde{\Omega}^n(\Omega^n(\mathcal{U}_{n-1}, \cdot), p) \\
\text{[by } (\Pi_2\text{-}\Sigma)] \quad &= \Pi^\bullet_{X:\mathcal{U}_{n-1}}\Pi^\bullet_{p:\Omega^n(\mathcal{U}_{n-1},X)}\Sigma^\bullet_{\Omega^n(\mathcal{U}_{n-1},X)}\tilde{\Omega}^n(\Omega^n(\mathcal{U}_{n-1}, \cdot), p).
\end{aligned}$$

But what does $\tilde{\Omega}^n(\Omega^n(\mathcal{U}_{n-1}, \cdot), p)$ compute to (recall the definition of $\tilde{\Omega}$ from the section on pointed types)?

3.8.2 The Remedy

There is a way to entirely avoid higher homotopical complexity, obviating the need to compute $\tilde{\Omega}^n(\Omega^n(\mathcal{U}_{n-1}, \cdot), p)$. Instead of looking at images of n-spheres in $\mathcal{U}_{n-1}$, we will look at such images in $\mathcal{U}^{n-1}_{n-1}$:

$$A \;:\equiv\; \textstyle\sum_{(Y:\mathcal{U}^{n-1}_{n-1})}\Omega^n(\mathcal{U}^{n-1}_{n-1}, Y).$$

Note that A inherits its status as an n-type from $\mathcal{U}^{n-1}_{n-1}$.

Let us once again calculate:

$$\begin{aligned}
\Omega^{n+1}(\mathcal{U}_n, A) &= \Pi^\bullet_{(Y,q):A}\Omega^n(A, (Y, q)) \\
&\equiv \Pi^\bullet_{(Y,q):A}\Omega^n(\Sigma^\bullet_{(\mathcal{U}^{n-1}_{n-1},Y)}(\Omega^n(\mathcal{U}^{n-1}_{n-1}, \cdot), q)) \\
\text{[by } (\Omega\text{-}\Sigma^\bullet)] \quad &= \Pi^\bullet_{(Y,q):A}\Sigma^\bullet_{\Omega^n(\mathcal{U}^{n-1}_{n-1},Y)}\tilde{\Omega}^n(\Omega^n(\mathcal{U}^{n-1}_{n-1}, \cdot), q)
\end{aligned}$$

Recalling $\mathcal{U}^{n-1}_{n-1}$ is an n-type, we see $\Omega^n(\mathcal{U}^{n-1}_{n-1}, \cdot)$ is a family of sets. Since we are in the case $n \geq 2$, we have $\tilde{\Omega}^2(\ldots)$ contractible:

$$\begin{aligned}
\text{[by } (\Sigma_2\text{-}\mathbf{1})] \quad (\ldots) &= \Pi^\bullet_{(Y,q):A}\Omega^n(\mathcal{U}^{n-1}_{n-1}, Y) \\
&= \Pi^\bullet_{Y:A}\Pi^\bullet_{q:\Omega^n(\mathcal{U}^{n-1}_{n-1},Y)}\Omega^n(\mathcal{U}^{n-1}_{n-1}, Y).
\end{aligned}$$

The remainder of the argument proceeds similar to the case of $n \equiv 1$. Consider the elements

$$\begin{aligned}
f_0(Y, q) &:\equiv \mathsf{refl}, \\
f_1(Y, q) &:\equiv q,
\end{aligned}$$

of the above pointed type, the former just being basepoint. Assuming $f_0 = f_1$, we have $\mathsf{refl} = q$ for all $Y : \mathcal{U}^{n-1}_{n-1}$ and $q : \Omega^n(\mathcal{U}^{n-1}_{n-1}, Y)$. By Lemma 3.1, this amount to $\mathcal{U}^{n-1}_{n-1}$ being an $(n-1)$-type. However, as seen below, we may assume by induction that $\mathcal{U}^{n-1}_{n-1}$ is strictly an n-type.

This proves $f_0 \neq f_1$ and, again by Lemma 3.1, shows $\mathcal{U}_n$ a non-n-type. Furthermore, since A is actually an n-type (this holds true in the section on $n \equiv 1$ as well), this also shows $\mathcal{U}_n^n$ a non-n-type, making it a strict $(n+1)$-type.

3.9 Further Work

It is possible to gain a better understanding of why our first approach failed. For this, we need to develop some homotopy theory, requiring the introduction of so-called higher inductive types. Using a practical theory of homotopy coherent diagrams and manipulating such diagrams while controlling for their homotopy colimit, we may define the homotopy type theoretic Whitehead product. Together with a parametricity argument to contrain the possible closed terms of type $\Pi^{\bullet}_{a:A}\Omega^n(A,a)$ with $A :\equiv \sum_{(X:\mathcal{U}_{n-1})} \Omega^n(\mathcal{U}_{n-1}, X)$, known results [38] on the Whitehead product with the identity map on spheres show that the approach constructing non-trivial higher loops in unrestricted universes cannot possibly work for even n, in particular $n = 2$.

However, all of this, in particular the theory of homotopy coherent diagrams, depends on a workable generalization of the ω-groupoidal structure of types to an $(\omega, 1)$-categorical structure containing the original ω-groupoidal structure as the ω-subcategory of invertible morphisms. This structure is still preserved internally, but only by those operations that properly take into account variance.

As an example, consider the type of arrows in a universe $\mathcal{U}$ given by

$$W :\equiv \sum_{(X:\mathcal{U})}\sum_{(Y:\mathcal{U})} X \to Y.$$

Univalence identifies the identity type on W with its structural equality: an equality between elements (X_0, Y_0, f_0) and (X_1, Y_1, f_1) is given by triples (u, v, p) where $u : X_0 \simeq X_1$ and $v : Y_0 \simeq Y_1$ establish equivalence of source and target while

$$p_x : v(f_0(u^{-1}(x))) = x \tag{3.1}$$

for $x : X_1$ witnesses coherence of the morphisms. This coincides with the usual notion of isomorphism for arrows. Now consider the notion of morphisms for arrows: a morphism from (X_0, Y_0, f_0) to (X_1, Y_1, f_1) should consist of a triple (u, v, p) where $u : X_0 \to X_1$ and $v : Y_0 \to Y_1$ such that

$$v(f_0(x)) = f_1(u(x)) \tag{3.2}$$

for $x : X_0$, the only essential difference being the use of functions u and v instead of equivalences.

The need for rearranging the inverted occurrence of u when going from (3.1) to (3.2) is not in any way ad-hoc, but reveals the need for a proper treatment of variance: the use of u in translating between the domains of f_0 and f_1 is contravariant.

Can we find a framework for type theory that endows every type with a notion of *directed* identity types that represent such a notion of morphisms similarly to how identity types represent isomorphism of structures? This is the topic of *directed type theory*, a subject still under development. A reworked version of the univalence axiom in this setting is expected to have connections with parametricity: directed identity types on universes are just function spaces.

In practical usage, the directed interpretation should allow the automatic and synthetic generation of correct (higher) categorical structures for given type derivations, obviating the need for explicitly defining these notions and then having to show coherence and preservation properties. As an example for the potential use of this, consider the problem of formally showing that the cartesian square of the 1-sphere is isomorphic to the torus, a formally completely expanded proof of which has recently been given by Kristina Sojakova. [24] With the proper directed categorical notions, we speculate this result should fall out of manipulations of algebras of loops.

[24] See `http://ncatlab.org/homotopytypetheory/files/torus.pdf`.

3.10 Related Work

Our result falls cleanly into the recently emerged area of homotopy type theory [56]. It is very specific and answers one detail question left open at the Special Year on Univalent Foundations regarding non-trivial higher homotopies in higher univalent universes. Several other groups of people independently made different candidate proposals for attacking this problem in unpublished communication.

Coquand suggested to use the type

$$A :\equiv \textstyle\sum_{(X:\mathcal{U}_0^0)} \sum_{(f:X=X)} f^2 = \mathsf{id} : \mathcal{U}_1$$

of $\mathbb{Z}/(2)$-sets. This is remarkably similar to our own suggestion for $\mathcal{U}_1$, differing only in having an extra dependent sum component. Note that our suggestion corresponds to $\mathbb{Z}$-sets. The proof of $\mathcal{U}_1$ not having truncation level one works out identically in both cases. However, it is not clear how to generalize Coquand's suggestion to higher universe/truncation levels.

A different approach was taken by Finster and Lumsdaine, who suggested to use the type

$$B_0 :\equiv \textstyle\sum_{(X:\mathcal{U}_0^0)} \|X = \mathbf{2}\|_{-1} : \mathcal{U}_1$$

of sets merely isomorphic to the Booleans. It should be noted that this makes use of propositional truncation, a language feature of homotopy type theory we have not needed to introduce in our brief treatment of the syntactic theory. Allowing truncation, a special kind of higher inductive type, has the potential of making the problem much easier. It can indeed be seen that defining

$$B_{n+1} :\equiv \textstyle\sum_{(X:\mathcal{U}_{n+1}^{n+1})} \|X = B_n\|_{-1} : \mathcal{U}_{n+2}$$

yields a family of types $B_n : \mathcal{U}_{n+1}^{n+1}$ with $n : \mathbb{N}$ such that $B_n = B_n$ is not an n-type.

Bibliography

[1] Michael Abbott, Thorsten Altenkirch, and Neil Ghani. Containers: Constructing strictly positive types. *Theoretical Computer Science*, 342(1):3–27, 2005.

[2] Michael Abott, Thorsten Altenkirch, Neil Ghani, and Conor McBride. Derivatives of containers. In *Typed Lambda Calculi and Applications, TLCA*, 2003.

[3] Michael Abott, Thorsten Altenkirch, Neil Ghani, and Conor McBride. Constructing polymorphic programs with quotient types. In *7th International Conference on Mathematics of Program Construction (MPC 2004)*, 2004.

[4] Thorsten Altenkirch. Isomorphisms on inductive types (talk), 2005. URL `http://www.cs.nott.ac.uk/~txa/talks/isos05.pdf`.

[5] Roland Backhouse, Marcel Bijsterveld, Rik van Geldrop, and Jaap Van Der Woude. Category theory as coherently constructive lattice theory, 1998. Working document.

[6] Roland Backhouse, Wei Chen, and JoãoF. Ferreira. The algorithmics of solitaire-like games. In Claude Bolduc, Jules Desharnais, and Béchir Ktari, editors, *Mathematics of Program Construction*, volume 6120 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2010.

[7] Marc Bezem, Thierry Coquand, and Simon Huber. A model of type theory in cubical sets. *Preprint*, 2013.

[8] Andreas Blass. Seven trees in one. *J. Pure Appl. Algebra*, 103:1–21, 1995.

[9] Corrado Böhm and Giuseppe Jacopini. Flow diagrams, Turing machines and languages with only two formation rules. *Commun. ACM*, 9(5):366–371, 1966.

[10] Wilfried Buchholz. A term calculus for (co-)recursive definitions on streamlike data structures. *Ann. Pure Appl. Logic*, 136(1-2):75–90, 2005.

[11] John H. Conway. Fractran: A simple universal programming language for arithmetic. In T. M. Cover and B. Gopinath, editors, *Open Problems in Communication and Computation*, chapter 2, pages 4–26. Springer, 1987.

[12] Edgar W. Dijkstra. On the producitivity of recursive definitions. EWD749, 1980.

[13] David Eisenbud. *Commutative algebra with a view toward algebraic geometry*, volume 150 of *Graduate Texts in Mathematics*. Springer, 1995.

[14] Jörg Endrullis, Clemens Grabmayer, and Dimitri Hendriks. Data-oblivious stream productivity. In *Proc. 15th Int. Conf. on LPAR*, LPAR '08, pages 79–96. Springer, 2008.

[15] Jörg Endrullis, Herman Geuvers, Jacob G. Simonses, and Hans Zantema. Levels of undecidability in rewriting. *Inf. Comput.*, 209(2):227–245, 2011.

[16] Jörg Endrullis, Clemens Grabmayer, Dimitri Hendriks, Ariya Isihara, and Jan Willem Klop. Productivity of stream definitions. In *Proc. FCT 2007*, volume 4639 of *LNCS*, pages 274–287. Springer, 2007.

[17] Jörg Endrullis, Clemens Grabmayer, and Dimitri Hendriks. Complexity of Fractran and productivity. In *CADE*, pages 371–387, 2009.

[18] Marcelo Fiore. Isomorphisms of generic recursive polynomial types. In *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '04, pages 77–88. ACM, 2004.

[19] Philippe Flajolet. Analytic models and ambiguity of context-free languages. *Theoretical Computer Science*, 49(2–3):283 – 309, 1987.

[20] Maarten M. Fokkinga. Monadic maps and folds for arbitrary datatypes. Memoranda Informatica 94-28, 1994.

[21] Gerhard Gierz, Karl Heinrich Hofmann, Klaus Keimel, Jimmie D. Lawson, Michael Mislove, and Dana S. Scott. *Continuous Lattices and Domains*, volume 93 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2003.

[22] Håkon R. Gylterud. Symmetric containers. Master's thesis, University of Oslo, 2011. URL `http://www.duo.uio.no/publ/matematikk/2011/144617/thesisgylterud.pdf`.

[23] David Harel. On folk theorems. *SIGACT News*, 12:68–80, 1980.

[24] Michael Hedberg. A coherence theorem for Martin-Löf's type theory. *J. Funct. Program.*, 8(4):413–436, 1998.

[25] Claudio Hermida and Bart Jacobs. Structural induction and coinduction in a fibrational setting. *Information and Computation*, 145(2):107–152, 1998.

[26] Andreas M. Hinz. The Tower of Hanoi. *Enseign. Math.*, 35(2):289–321, 1989.

[27] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In Giovanni Sambin and Jan M. Smith, editors, *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 83–111. Oxford University Press, 1998.

[28] Thiet-Dung Huynh. The complexity of semilinear sets. In Jaco Bakker and Jan Leeuwen, editors, *Automata, Languages and Programming*, volume 85 of *Lecture Notes in Computer Science*, pages 324–337. Springer, 1980.

[29] Ryuichi Ito. Every semilinear set is a finite union of disjoint linear sets. *J. Comput. Syst. Sci.*, 3(2):221–231, 1969.

[30] Mauro Jaskelioff and Ondrej Rypacek. An investigation of the laws of traversals. In James Chapman and Paul Blain Levy, editors, *Proceedings of the Fourth Workshop on Mathematically Structured Functional Programming*, volume 76 of *EPTCS*, pages 40–49, 2012.

[31] C. Barry Jay. A semantics for shape. *Science of Computer Programming*, 25(2–3):251–283, 1995. Selected Papers of ESOP'94.

[32] André Joyal. Une théorie combinatoire des séries formelles. *Advances in Mathematics*, 42 (1):1–82, 1981.

[33] Chris Kapulkin, Peter LeFanu Lumsdaine, and Vladimir Voevodsky. Univalence in simplicial sets. 2012. ArXiv e-print 1203.2553.

[34] Stephen C. Kleene. Recursive predicates and quantifiers. *Trans. AMS*, 53(1):41–73, 1943.

[35] Donald E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, third edition, 1997.

[36] Nicolai Kraus and Christian Sattler. On the Hierarchy of Univalent Universes: U(n) is not n-Truncated. 2013. ArXiv e-print 1311.4002.

[37] Nicolai Kraus and Christian Sattler. On the hierarchy of univalent universes: U(n) is not n-truncated, 2013. URL `http://github.com/sattlerc/HoTT-Agda/tree/universe-article`. Agda Formalization.

[38] Leif Kristensen and Ib Madsen. Note on whitehead products in spheres. *Mathematica Scandinavica*, 21:301–314, 1967.

[39] Joseph B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory, Series A*, 13(3):297 – 305, 1972.

[40] Stuart A. Kurtz and Janos Simon. The undecidability of the generalized Collatz problem. In *TAMS*, volume 4484 of *LNCS*, pages 542–553. Springer, 2007.

[41] Jeffery C. Lagarias. *The Ultimate Challenge: The* $3x + 1$ *Problem*. AMS, 2010.

[42] Per Martin-Löf. An intuitionistic theory of types. In Giovanni Sambin and Jan M. Smith, editors, *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 127–172. Oxford University Press, 1998.

[43] Conor McBride. Elimination with a motive. In Paul Callaghan, Zhaohui Luo, James McKinna, and Robert Pollack, editors, *Types for Proofs and Programs (Proceedings of the International Workshop, TYPES'00)*, volume 2277 of *LNCS*. Springer-Verlag, 2002.

[44] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

[45] Eugenio Moggi, Gianna Belle, and C. Barry Jay. Monads, shapely functors and traversals. *Electronic Notes in Theoretical Computer Science*, 29:187–208, 1999.

[46] Rohit J. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, 1966.

[47] Christine Paulin-Mohring. Inductive Definitions in the System Coq - Rules and Properties. In Marc Bezem and Jan Friso Groote, editors, *Proceedings of the conference Typed Lambda Calculi and Applications*, number 664 in Lecture Notes in Computer Science, 1993.

[48] Eleonora Perkowska. Theorem on the normal form of a program. *Bull. Acad. Pol. Sci., Ser. Sci. Math. Astr. Phys.*, 22(4):439–442, 1974.

[49] Dayanand S. Rajan. The equations $D^kY = X^n$ in combinatorial species. *Discrete Mathematics*, 118(1–3):197–206, 1993.

[50] Grigore Roşu. Equality of streams is a Π^0_2-complete problem. In *ICFP*. ACM, 2006.

[51] Jan M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theor. Comp. Sci.*, 308(1-3):1–53, 2003.

[52] Christian Sattler and Florent Balestrieri. Turing-completeness of polymorphic stream equation systems. In Ashish Tiwari, editor, *RTA*, volume 15 of *LIPIcs*, pages 256–271, 2012.

[53] Jakob Grue Simonsen. The Π^0_2-completeness of most of the properties of rewriting systems you care about (and productivity). In *Proc. 20th Int. Conf. on RTA*, RTA '09, pages 335–349. Springer, 2009.

[54] Robert I. Soare. *Recursively Enumerable Sets and Degrees*. Perspectives in Mathematical Logic. Springer, 1987.

[55] Thomas Streicher. Investigations into intensional type theory, 1993. Habilitationsschrift, Ludwig-Maximilians-Universität München.

[56] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `http://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.

[57] Alfred J. van der Poorten. Power series representing algebraic functions. In Sinnou David, editor, *Séminaire de Théorie des Nombres, Paris (1990-1991)*, volume 108 of *Progress in Math.*, pages 241–262. Birkhäuser, 1993.

[58] Vladimir Voevodsky. A very short note on the homotopy λ-calculus. `http://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/Hlambda_short_current.pdf`, 2006.

[59] Hans Zantema. Well-definedness of streams by transformation and termination. *LMCS*, 6 (3), 2010. paper 21.